API User's Guide

Java API

Java Application Programming Interface

User's Guide

How the API Is Organized

There are three levels to the API:

- All Packages
- All Classes (within a package)
- This Class (selected class).

Level 1 - All Packages

This level of the API provides links to the packages in the 1.0 release.

Level 2 - This Package

This level provides links to the classes and interfaces in a given package. There are three catogories in the listing:

- Interfaces
- Classes
- Exceptions

Level 3 - This Class/Interface

This level begins with an index, followed by the detailed API. There are three categories at the class level.

- Variables
- Constructors
- Methods

A category is omitted when a class has no applicable entries.

Within these categories there is additional color coding as follows:
Instance Variables
Static Variables
Constructors
Instance Methods
Static Methods

How to Locate Items

- To Browse A Package
  Select a package from the list of All Packages. This list is the home page for the the Java API.
- To Locate a Class
  Use the searchable index tool.
  Or, select its package.
  Select the class from the alphabetical index.
- To Browse a Class
  Use the Next/Previous anchors to browse alphabetically.
  Or, traverse the links within the class.
- To Locate a Method
  Use the searchable index tool.
  Or, scroll through the alphabetical class index to locate a method.

A Closer Look at the Class-Level API

Take a look at class String in the package java.lang. The navigational anchors are at the top. This is followed by the fully qualified class name and a representation of its position in the class hierarchy.

The next entries are links to the superclass and the interfaces, if any. This is followed by a description of the class, taken from the class comment. Notice how the programmer has embedded some code samples using html tags.

The author also chose to include a See Also entry to another class. Following the class-level entries for See Also, Version, and Author, the index begins.

The Index

Each class/interface begins with an index of its variables, constructors and methods, sorted alphabetically. The entry consists of the declaration and short description. The description is the first sentence of the doc comment for that item. The index e ntries are linked to their corresponding entries in the application programming inteface which immediately follows.

The Detailed API

The index is followed by the complete API for each entry. Within the three categories: Variables, Constructors, and Methods, the entries are presented in the order they appear in the source. This is done to preserve the logical groupings established by the programmer.

Where Are All the Links in the API?

- At the top of each class/interface there are navigational anchors to the other levels and to Previous and Next (class or interface).
- There are links in the class type of every method and variable definition.
- At the top of each class/interface there is a drawing of the tree structure down to the current class/interface, in which each superclass is a link.
- Every method contains a list of exceptions that it may throw. These are linked to the appropriate class.
- The superclass and interface references at the beginning of the class are links.
- Every See Also is a link.
- When a method overrides a method in the superclass, the API has the entry "Overrides: foo in class bar." Both foo (the method name) and bar (the class name) are links.

Packages

API User's Guide

Package Index

Applet API Packages

java.applet
java.awt
java.awt.image
java.awt.peer
java.io
java.lang
java.net
java.util

Other Packages

sun.tools.debug

This documentation was ported to *MS Window's Help* by Bill Bercik.
Bill may be reached at: bill@dippybird.com

Stop by his web site and get the latest update to the *Information Portafilter* for Java at:
http://www.dippybird.com/java.html

Package java.applet

---

package java.applet

Interface Index

AppletContext
AppletStub
AudioClip

Class Index

Applet

Package java.awt

---

package java.awt

Interface Index

Class Index

Exception Index

Error Index

Package java.awt.image

---

package java.awt.image

Interface Index

ImageConsumer
ImageObserver
ImageProducer

Class Index

ColorModel
CropImageFilter
DirectColorModel
FilteredImageSource
ImageFilter
IndexColorModel
MemoryImageSource
PixelGrabber
RGBImageFilter

Package java.awt.peer

---

package java.awt.peer

Package java.io

---

package java.io

Interface Index

DataInput
DataOutput
FilenameFilter

Class Index

BufferedInputStream
BufferedOutputStream
ByteArrayInputStream
ByteArrayOutputStream
DataInputStream
DataOutputStream
File
FileDescriptor
FileInputStream
FileOutputStream
FilterInputStream
FilterOutputStream
InputStream
LineNumberInputStream
OutputStream
PipedInputStream
PipedOutputStream
PrintStream
PushbackInputStream
RandomAccessFile
SequenceInputStream
StreamTokenizer
StringBufferInputStream

Exception Index

EOFException
FileNotFoundException
IOException
InterruptedIOException
UTFDataFormatException

Package java.lang

---

package java.lang

Interface Index

Cloneable
Runnable

Class Index

Boolean
Character
Class
ClassLoader
Compiler
Double
Float
Integer
Long
Math
Number
Object
Process
Runtime
SecurityManager
String
StringBuffer
System
Thread
ThreadGroup
Throwable

Exception Index

ArithmeticException
ArrayIndexOutOfBoundsException
ArrayStoreException
ClassCastException
ClassNotFoundException
CloneNotSupportedException
Exception
IllegalAccessException
IllegalArgumentException

IllegalMonitorStateException
IllegalThreadStateException
IndexOutOfBoundsException
InstantiationException
InterruptedException
NegativeArraySizeException
NoSuchMethodException
NullPointerException
NumberFormatException
RuntimeException
SecurityException
StringIndexOutOfBoundsException

Error Index

AbstractMethodError
ClassCircularityError
ClassFormatError
Error
IllegalAccessError
IncompatibleClassChangeError
InstantiationError
InternalError
LinkageError
NoClassDefFoundError
NoSuchFieldError
NoSuchMethodError
OutOfMemoryError
StackOverflowError
ThreadDeath
UnknownError
UnsatisfiedLinkError
VerifyError
VirtualMachineError

Package java.net

---

package java.net

Interface Index

ContentHandlerFactory
SocketImplFactory
URLStreamHandlerFactory

Class Index

ContentHandler
DatagramPacket
DatagramSocket
InetAddress
ServerSocket
Socket
SocketImpl
URL
URLConnection
URLEncoder
URLStreamHandler

Class Index

MalformedURLException
ProtocolException
SocketException
UnknownHostException
UnknownServiceException

Package java.util

---

package java.util

Enumeration
Observer

BitSet
Date
Dictionary
Hashtable
Observable
Properties
Random
Stack
StringTokenizer
Vector

EmptyStackException
NoSuchElementException

Package sun.tools.debug

---

package sun.tools.debug

Class Index

DebuggerCallback

Class Index

RemoteArray
RemoteBoolean
RemoteByte
RemoteChar
RemoteClass
RemoteDebugger
RemoteDouble
RemoteField
RemoteFloat
RemoteInt
RemoteLong
RemoteObject
RemoteShort
RemoteStackFrame
RemoteStackVariable
RemoteString
RemoteThread
RemoteThreadGroup
RemoteValue
StackFrame

Index of all Fields and Methods

Index of all Fields and Methods

A

Adds the specified menu to the menu bar.
add(MenuItem). Method in class java.awt.Menu
Adds the specified item to this menu.
add(Point). Method in class java.awt.Rectangle
Adds a point to a rectangle.
add(Rectangle). Method in class java.awt.Rectangle
Adds a rectangle to a rectangle.
add(String). Method in class java.awt.Menu
Adds an item with with the specified label to this menu.
add(String, Component). Method in class java.awt.Container
Adds the specified component to this container.
addConsumer(ImageConsumer). Method in class java.awt.image.FilteredImageSource
Adds an ImageConsumer to the list of consumers interested in data for this image.
addConsumer(ImageConsumer). Method in interface java.awt.image.ImageProducer
This method is used to register an ImageConsumer with the ImageProducer for access to the image data during a later reconstruction of the Image.
addConsumer(ImageConsumer). Method in class java.awt.image.MemoryImageSource
Adds an ImageConsumer to the list of consumers interested in data for this image.
addElement(Object). Method in class java.util.Vector
Adds the specified object as the last element of the vector.
addHelpMenu(Menu). Method in interface java.awt.peer.MenuBarPeer
addImage(Image, int). Method in class java.awt. MediaTracker
Adds an image to the list of images being tracked.
addImage(Image, int, int, int). Method in class java.awt.MediaTracker
Adds a scaled image to the list of images being tracked.
addItem(MenuItem). Method in interface java.awt.peer.MenuPeer
addItem(String). Method in class java.awt.Choice
Adds an item to this Choice.
addItem(String). Method in class java.awt.List
Adds the specified item to the end of scrolling list.
addItem(String, int). Method in interface java.awt.peer.ChoicePeer
addItem(String, int). Method in class java.awt.List
Adds the specified item to the end of scrolling list.
addItem(String, int). Method in interface java.awt.peer.ListPeer
addLayoutComponent(String, Component). Method in class java.awt.BorderLayout
Adds the specified named component to the layout.
addLayoutComponent(String, Component). Method in class java.awt.CardLayout
Adds the specified component with the specified name to the layout.
addLayoutComponent(String, Component). Method in class java.awt.FlowLayout
Adds the specified component to the layout.
addLayoutComponent(String, Component). Method in class java.awt.GridBagLayout
Adds the specified component with the specified name to the layout.
addLayoutComponent(String, Component). Method in class java.awt.GridLayout
Adds the specified component with the specified name to the layout.
addLayoutComponent(String, Component). Method in interface java.awt. LayoutManager
Adds the specified component with the specified name to the layout.
addMenu(Menu). Method in interface java.awt.peer.MenuBarPeer
addNotify(). Method in class java.awt.Button
Creates the peer of the button.
addNotify(). Method in class java.awt.Canvas
Creates the peer of the canvas.
addNotify(). Method in class java.awt.Checkbox
Creates the peer of the Checkbox.
addNotify(). Method in class java.awt.CheckboxMenuItem
Creates the peer of the checkbox item.
addNotify(). Method in class java.awt.Choice

Creates the Choice's peer.
addNotify(). Method in class java.awt.Component
Notifies the Component to create a peer.
addNotify(). Method in class java.awt.Container
Notifies the container to create a peer.
addNotify(). Method in class java.awt.Dialog
Creates the frame's peer.
addNotify(). Method in class java.awt.FileDialog
Creates the frame's peer.
addNotify(). Method in class java.awt.Frame
Creates the Frame's peer.
addNotify(). Method in class java.awt.Label
Creates the peer for this label.
addNotify(). Method in class java.awt.List
Creates the peer for the list.
addNotify(). Method in class java.awt.Menu
Creates the menu's peer.
addNotify(). Method in class java.awt.MenuBar
Creates the menu bar's peer.
addNotify(). Method in class java.awt.MenuItem
Creates the menu item's peer.
addNotify(). Method in class java.awt.Panel
Creates the Panel's peer.
addNotify(). Method in class java.awt.Scrollbar
Creates the Scrollbar's peer.
addNotify(). Method in class java.awt.TextArea
Creates the TextArea's peer.
addNotify(). Method in class java.awt.TextField
Creates the TextField's peer.
addNotify(). Method in class java.awt.Window
Creates the Window's peer.
addObserver(Observer). Method in class java.util.Observable
Adds an observer to the observer list.
addPoint(int, int). Method in class java.awt.Polygon
Appends a point to a polygon.
address. Variable in class java.net.SocketImpl
The internet address where the socket will make a connection.
addSeparator(). Method in class java.awt.Menu
Adds a separator line, or a hypen, to the menu at the current position.
addSeparator(). Method in interface java.awt.peer.MenuPeer
AdjustForGravity(GridBagConstraints, Rectangle). Method in class java.awt.GridBagLayout
after(Date). Method in class java.util.Date
Checks whether this date comes after the specified date.
ALLBITS. Static variable in interface java.awt.image.ImageObserver
A static image which was previously drawn is now complete and can be drawn again in its final form.
allowsMultipleSelections(). Method in class java.awt.List
Returns true if this list allows multiple selections.
allowUserInteraction. Variable in class java.net.URLConnection
ALT_MASK. Static variable in class java.awt.Event
The alt modifier constant.
anchor. Variable in class java.awt.GridBagConstraints
and(BitSet). Method in class java.util.BitSet
Logically ANDs this bit set with the specified set of bits.
append(boolean). Method in class java.lang.StringBuffer
Appends a boolean to the end of this buffer.
append(char). Method in class java.lang.StringBuffer

Appends a character to the end of this buffer.
append(char[]). Method in class java.lang.StringBuffer
Appends an array of characters to the end of this buffer.
append(char[], int, int). Method in class java.lang.StringBuffer
Appends a part of an array of characters to the end of this buffer.
append(double). Method in class java.lang.StringBuffer
Appends a double to the end of this buffer.
append(float). Method in class java.lang.StringBuffer
Appends a float to the end of this buffer.
append(int). Method in class java.lang.StringBuffer
Appends an integer to the end of this buffer.
append(long). Method in class java.lang.StringBuffer
Appends a long to the end of this buffer.
append(Object). Method in class java.lang.StringBuffer
Appends an object to the end of this buffer.
append(String). Method in class java.lang.StringBuffer
Appends a String to the end of this buffer.
appendText(String). Method in class java.awt.TextArea
Appends the given text to the end.
Applet(). Constructor for class java.applet.Applet
appletResize(int, int). Method in interface java.applet.AppletStub
Called when the applet wants to be resized.
arg. Variable in class java.awt.Event
An arbitraty argument.
ArithmeticException(). Constructor for class java.lang.ArithmeticException
Constructs an ArithmeticException with no detail message.
ArithmeticException(String). Constructor for class java.lang. ArithmeticException
Constructs an ArithmeticException with the specified detail message.
ArrangeGrid(Container). Method in class java.awt.GridBagLayout
arraycopy(Object, int, Object, int, int). Static method in class java.lang.System
Copies an array from the source array, beginning at the specified position, to the specified position of the destination array.
ArrayIndexOutOfBoundsException(). Constructor for class java.lang.
ArrayIndexOutOfBoundsException
Constructs an ArrayIndexOutOfBoundsException with no detail message.
ArrayIndexOutOfBoundsException(int). Constructor for class java.lang.
ArrayIndexOutOfBoundsException
Constructs a new ArrayIndexOutOfBoundsException class initialized to the specific index.
ArrayIndexOutOfBoundsException(String). Constructor for class java.lang.
ArrayIndexOutOfBoundsException
Constructs an ArrayIndexOutOfBoundsException class with the specified detail message.
ArrayStoreException(). Constructor for class java.lang.ArrayStoreException
Constructs a ArrayStoreException with no detail message.
ArrayStoreException(String). Constructor for class java.lang. ArrayStoreException
Constructs a ArrayStoreException with the specified detail message.
arrayTypeName(int). Method in class sun.tools.debug.RemoteArray
Return the element type as a string.
asin(double). Static method in class java.lang.Math
Returns the arc sine of a, in the range of -Pi/2 through Pi/2.
atan(double). Static method in class java.lang.Math
Returns the arc tangent of a, in the range of -Pi/2 through Pi/2.
atan2(double, double). Static method in class java.lang.Math
Converts rectangular coordinates (a, b) to polar (r, theta).
available(). Method in class java.io.BufferedInputStream
Returns the number of bytes that can be read without blocking.
available(). Method in class java.io.ByteArrayInputStream

Returns the number of available bytes in the buffer.
available(). Method in class java.io.FileInputStream
Returns the number of bytes that can be read without blocking.
available(). Method in class java.io.FilterInputStream
Returns the number of bytes that can be read without blocking.
available(). Method in class java.io.InputStream
Returns the number of bytes that can be read without blocking.
available(). Method in class java.io.LineNumberInputStream
Returns the number of bytes that can be read without blocking.
available(). Method in class java.io.PushbackInputStream
Returns the number of bytes that can be read.
available(). Method in class java.net.SocketImpl
Returns the number of bytes that can be read without blocking.
available(). Method in class java.io.StringBufferInputStream
Returns the number of available bytes in the buffer.
AWTError(String). Constructor for class java.awt.AWTError
AWTException(String). Constructor for class java.awt. AWTException
Constructs an AWTException with the specified detail message.

B

before(Date).Method in class java.util.Date
Checks whether this date comes before the specified date.
bind(InetAddress, int). Method in class java.net.SocketImpl
Binds the socket to the specified port on the specified host.
BitSet(). Constructor for class java.util.BitSet
Creates an empty set.
BitSet(int). Constructor for class java.util.BitSet
Creates an empty set with the specified size.
black. Static variable in class java.awt.Color
The color black.
blue. Static variable in class java.awt.Color
The color blue.
BOLD. Static variable in class java.awt.Font
The bold style constant.
Boolean(boolean). Constructor for class java.lang.Boolean
Constructs a Boolean object initialized to the specified boolean value.
Boolean(String). Constructor for class java.lang.Boolean
Constructs a Boolean object initialized to the value specified by the String parameter.
booleanValue(). Method in class java.lang.Boolean
Returns the value of this Boolean object as a boolean.
BorderLayout(). Constructor for class java.awt.BorderLayout
Constructs a new BorderLayout.
BorderLayout(int, int). Constructor for class java.awt.BorderLayout
Constructs a BorderLayout with the specified gaps.
BOTH. Static variable in class java.awt.GridBagConstraints
bottom. Variable in class java.awt.Insets
The inset from the bottom.
bounds(). Method in class java.awt.Component
Returns the current bounds of this component.
breakpointEvent(RemoteThread). Method in interface sun.tools.debug. DebuggerCallback
A breakpoint has been hit in the specified thread.
brighter(). Method in class java.awt.Color

Returns a brighter version of this color.
buf. Variable in class java.io.BufferedInputStream
The buffer where data is stored.
buf. Variable in class java.io.BufferedOutputStream
The buffer where data is stored.
buf. Variable in class java.io.ByteArrayInputStream
The buffer where data is stored.
buf. Variable in class java.io.ByteArrayOutputStream
The buffer where data is stored.
buffer. Variable in class java.io.StringBufferInputStream
The buffer where data is stored.
BufferedInputStream(InputStream). Constructor for class java.io. BufferedInputStream
Creates a new buffered stream with a default buffer size.
BufferedInputStream(InputStream, int). Constructor for class java.io. BufferedInputStream
Creates a new buffered stream with the specified buffer size.
BufferedOutputStream(OutputStream). Constructor for class java.io. BufferedOutputStream
Creates a new buffered stream with a default buffer size.
BufferedOutputStream(OutputStream, int). Constructor for class java.io. BufferedOutputStream
Creates a new buffered stream with the specified buffer size.
Button(). Constructor for class java.awt.Button
Constructs a Button with no label.
Button(String). Constructor for class java.awt.Button
Constructs a Button with a string label.
ByteArrayInputStream(byte[]). Constructor for class java.io. ByteArrayInputStream
Creates an ByteArrayInputStream from the specified array of bytes.
ByteArrayInputStream(byte[], int, int). Constructor for class java.io. ByteArrayInputStream
Creates an ByteArrayInputStream from the specified array of bytes.
ByteArrayOutputStream(). Constructor for class java.io.ByteArrayOutputStream
Creates a new ByteArrayOutputStream.
ByteArrayOutputStream(int). Constructor for class java.io.ByteArrayOutputStream
Creates a new ByteArrayOutputStream with the specified initial size.
bytesTransferred. Variable in class java.io.InterruptedIOException
Reports how many bytes had been transferred as part of the IO operation before it was interrupted.
bytesWidth(byte[], int, int). Method in class java.awt.FontMetrics
Returns the width of the specified array of bytes in this Font.

C

canFilterIndexColorModel.Variable in class java.awt.image.RGBImageFilter
This boolean indicates whether or not it is acceptable to apply the color filtering of the filterRGB method to the color table entries of an IndexColorModel object in lieu of pixel by pixel filtering.
canRead(). Method in class java.io.File
Returns a boolean indicating whether or not a readable file exists.
Canvas(). Constructor for class java.awt.Canvas
canWrite(). Method in class java.io.File
Returns a boolean indicating whether or not a writable file exists.
capacity(). Method in class java.lang.StringBuffer
Returns the current capacity of the String buffer.
capacity(). Method in class java.util.Vector
Returns the current capacity of the vector.
capacityIncrement. Variable in class java.util.Vector
The size of the increment.
CardLayout(). Constructor for class java.awt.CardLayout

Creates a new card layout.
CardLayout(int, int). Constructor for class java.awt.CardLayout
Creates a card layout with the specified gaps.
catchExceptions(). Method in class sun.tools.debug.RemoteClass
Enter the debugger when an instance of this class is thrown.
ceil(double). Static method in class java.lang.Math
Returns the "ceiling" or smallest whole number greater than or equal to a.
CENTER. Static variable in class java.awt.FlowLayout
The right alignment variable.
CENTER. Static variable in class java.awt.GridBagConstraints
CENTER. Static variable in class java.awt.Label
The center alignment.
Character(char). Constructor for class java.lang.Character
Constructs a Character object with the specified value.
charAt(int). Method in class java.lang.String
Returns the character at the specified index.
charAt(int). Method in class java.lang.StringBuffer
Returns the character at the specified index.
charsWidth(char[], int, int). Method in class java.awt.FontMetrics
Returns the width of the specified character array in this Font.
charValue(). Method in class java.lang.Character
Returns the value of this Character object.
charWidth(char). Method in class java.awt.FontMetrics
Returns the width of the specified character in this Font.
charWidth(int). Method in class java.awt.FontMetrics
Returns the width of the specified character in this Font.
checkAccept(String, int). Method in class java.lang.SecurityManager
Checks to see if a socket connection to the specified port on the specified host has been accepted.
checkAccess(). Method in class java.lang.Thread
Checks whether the current Thread is allowed to modify this Thread.
checkAccess(). Method in class java.lang.ThreadGroup
Checks to see if the current Thread is allowed to modify this group.
checkAccess(Thread). Method in class java.lang.SecurityManager
Checks to see if the specified Thread is allowed to modify the Thread group.
checkAccess(ThreadGroup). Method in class java.lang.SecurityManager
Checks to see if the specified Thread group is allowed to modify this group.
checkAll(). Method in class java.awt.MediaTracker
Checks to see if all images have finished loading but does not start loading the images if they are not already loading.
checkAll(boolean). Method in class java.awt.MediaTracker
Checks to see if all images have finished loading.
Checkbox(). Constructor for class java.awt.Checkbox
Constructs a Checkbox with no label, no Checkbox group, and initialized to a false state.
Checkbox(String). Constructor for class java.awt.Checkbox
Constructs a Checkbox with the specified label, no Checkbox group, and initialized to a false state.
Checkbox(String, CheckboxGroup, boolean). Constructor for class java.awt. Checkbox
Constructs a Checkbox with the specified label, specified Checkbox group, and specified boolean state.
CheckboxGroup(). Constructor for class java.awt.CheckboxGroup
Creates a new CheckboxGroup.
CheckboxMenuItem(String). Constructor for class java.awt.CheckboxMenuItem
Creates the checkbox item with the specified label.
checkConnect(String, int). Method in class java.lang.SecurityManager
Checks to see if a socket has connected to the specified port on the the specified host.
checkConnect(String, int, Object). Method in class java.lang.SecurityManager
Checks to see if the current execution context and the indicated execution context are both allowed

to connect to the indicated host and port.
checkCreateClassLoader(). Method in class java.lang.SecurityManager
Checks to see if the ClassLoader has been created.
checkDelete(String). Method in class java.lang.SecurityManager
Checks to see if a file with the specified system dependent file name can be deleted.
checkError(). Method in class java.io.PrintStream
Flushes the print stream and returns whether or not there was an error on the output stream.
checkExec(String). Method in class java.lang.SecurityManager
Checks to see if the system command is executed by trusted code.
checkExit(int). Method in class java.lang.SecurityManager
Checks to see if the system has exited the virtual machine with an exit code.
checkID(int). Method in class java.awt.MediaTracker
Checks to see if all images tagged with the indicated ID have finished loading, but does not start loading the images if they are not already loading.
checkID(int, boolean). Method in class java.awt.MediaTracker
Checks to see if all images tagged with the indicated ID have finished loading.
checkImage(Image, ImageObserver). Method in class java.awt.Component
Returns the status of the construction of a screen representation of the specified image.
checkImage(Image, int, int, ImageObserver). Method in class java.awt.Component
Returns the status of the construction of a scaled screen representation of the specified image.
checkImage(Image, int, int, ImageObserver). Method in interface java.awt.peer. ComponentPeer
checkImage(Image, int, int, ImageObserver). Method in class java.awt.Toolkit
Returns the status of the construction of the indicated method at the indicated width and height for the default screen.
checkLink(String). Method in class java.lang.SecurityManager
Checks to see if the specified linked library exists.
checkListen(int). Method in class java.lang.SecurityManager
Checks to see if a server socket is listening to the specified local port that it is bounded to.
checkPackageAccess(String). Method in class java.lang.SecurityManager
Checks to see if an applet can access a package.
checkPackageDefinition(String). Method in class java.lang.SecurityManager
Checks to see if an applet can define classes in a package.
checkPropertiesAccess(). Method in class java.lang.SecurityManager
Checks to see who has access to the System properties.
checkPropertyAccess(String). Method in class java.lang.SecurityManager
Checks to see who has access to the System property named by *key*.
checkPropertyAccess(String, String). Method in class java.lang.SecurityManager
Checks to see who has access to the System property named by *key* and *def*.
checkRead(FileDescriptor). Method in class java.lang.SecurityManager
Checks to see if an input file with the specified file descriptor object gets created.
checkRead(String). Method in class java.lang.SecurityManager
Checks to see if an input file with the specified system dependent file name gets created.
checkRead(String, Object). Method in class java.lang.SecurityManager
Checks to see if the current context or the indicated context are both allowed to read the given file name.
checkSetFactory(). Method in class java.lang.SecurityManager
Checks to see if an applet can set a networking-related object factory.
checkTopLevelWindow(Object). Method in class java.lang.SecurityManager
Checks to see if top-level windows can be created by the caller.
checkWrite(FileDescriptor). Method in class java.lang.SecurityManager
Checks to see if an output file with the specified file descriptor object gets created.
checkWrite(String). Method in class java.lang.SecurityManager
Checks to see if an output file with the specified system dependent file name gets created.
Choice(). Constructor for class java.awt.Choice
Constructs a new Choice.
ClassCastException(). Constructor for class java.lang.ClassCastException

Constructs a ClassCastException with no detail message.
ClassCastException(String). Constructor for class java.lang.ClassCastException
Constructs a ClassCastException with the specified detail message.
ClassCircularityError(). Constructor for class java.lang.ClassCircularityError
Constructs a ClassCircularityError with no detail message.
ClassCircularityError(String). Constructor for class java.lang. ClassCircularityError
Constructs a ClassCircularityError with the specified detail message.
classDepth(String). Method in class java.lang.SecurityManager
Return the position of the stack frame containing the first occurrence of the named class.
ClassFormatError(). Constructor for class java.lang.ClassFormatError
Constructs a ClassFormatError with no detail message.
ClassFormatError(String). Constructor for class java.lang.ClassFormatError
Constructs a ClassFormatError with the specified detail message.
ClassLoader(). Constructor for class java.lang.ClassLoader
Constructs a new Class loader and initializes it.
classLoaderDepth(). Method in class java.lang.SecurityManager
ClassNotFoundException(). Constructor for class java.lang.ClassNotFoundException
Constructs a ClassNotFoundException with no detail message.
ClassNotFoundException(String). Constructor for class java.lang. ClassNotFoundException
Constructs a ClassNotFoundException with the specified detail message.
clear(). Method in class java.util.Hashtable
Clears the hash table so that it has no more elements in it.
clear(). Method in class java.awt.List
Clears the list.
clear(). Method in interface java.awt.peer.ListPeer
clear(int). Method in class java.util.BitSet
Clears a bit.
clearBreakpoint(int). Method in class sun.tools.debug.RemoteClass
Clear a breakpoint at a specific address in a class.
clearBreakpointLine(int). Method in class sun.tools.debug.RemoteClass
Clear a breakpoint at a specified line.
clearBreakpointMethod(RemoteField). Method in class sun.tools.debug.RemoteClass
Clear a breakpoint at the start of a specified method.
clearChanged(). Method in class java.util.Observable
Clears an observable change.
clearRect(int, int, int, int). Method in class java.awt.Graphics
Clears the specified rectangle by filling it with the current background color of the current drawing surface.
clickCount. Variable in class java.awt.Event
The number of consecutive clicks.
clipRect(int, int, int, int). Method in class java.awt.Graphics
Clips to a rectangle.
clone(). Method in class java.util.BitSet
Clones the BitSet.
clone(). Method in class java.awt.GridBagConstraints
Creates a clone of the object.
clone(). Method in class java.util.Hashtable
Creates a clone of the hashtable.
clone(). Method in class java.awt.image.ImageFilter
Clones this object.
clone(). Method in class java.awt.Insets
Creates a clone of the object.
clone(). Method in class java.lang.Object
Creates a clone of the object.
clone(). Method in class java.util.Vector
Clones this vector.

CloneNotSupportedException(). Constructor for class java.lang. CloneNotSupportedException
Constructs an CloneNotSupportedException with no detail message.
CloneNotSupportedException(String). Constructor for class java.lang. CloneNotSupportedException
Constructs an CloneNotSupportedException with the specified detail message.
close(). Method in class java.net.DatagramSocket
Close the datagram socket.
close(). Method in class java.io.FileInputStream
Closes the input stream.
close(). Method in class java.io.FileOutputStream
Closes the stream.
close(). Method in class java.io.FilterInputStream
Closes the input stream.
close(). Method in class java.io.FilterOutputStream
Closes the stream.
close(). Method in class java.io.InputStream
Closes the input stream.
close(). Method in class java.io.OutputStream
Closes the stream.
close(). Method in class java.io.PipedInputStream
Closes the input stream.
close(). Method in class java.io.PipedOutputStream
Closes the stream.
close(). Method in class java.io.PrintStream
Closes the stream.
close(). Method in class java.io.RandomAccessFile
Closes the file.
close(). Method in class sun.tools.debug.RemoteDebugger
Close the connection to the remote debugging agent.
close(). Method in class java.io.SequenceInputStream
Closes the input stream; flipping to the next stream, if an EOF is reached.
close(). Method in class java.net.ServerSocket
Closes the server socket.
close(). Method in class java.net.Socket
Closes the socket.
close(). Method in class java.net.SocketImpl
Closes the socket.
Color(float, float, float). Constructor for class java.awt.Color
Creates a color with the specified red, green, and blue values in the range (0.0 - 1.0).
Color(int). Constructor for class java.awt.Color
Creates a color with the specified combined RGB value consisting of the red component in bits 16-23, the green component in bits 8-15, and the blue component in bits 0-7.
Color(int, int, int). Constructor for class java.awt.Color
Creates a color with the specified red, green, and blue values in the range (0 - 255).
ColorModel(int). Constructor for class java.awt.image.ColorModel
Constructs a ColorModel which describes a pixel of the specified number of bits.
columnWeights. Variable in class java.awt.GridBagLayout
columnWidths. Variable in class java.awt.GridBagLayout
command(Object). Static method in class java.lang. Compiler
commentChar(int). Method in class java.io. StreamTokenizer
Specifies that this character starts a single line comment.
compareTo(String). Method in class java.lang.String
Compares this String to another specified String.
compileClass(Class). Static method in class java.lang.Compiler
compileClasses(String). Static method in class java.lang.Compiler
COMPLETE. Static variable in class java.awt. MediaTracker
Flag indicating the download of media completed successfully.

COMPLETESCANLINES. Static variable in interface java.awt.image.ImageConsumer
The pixels will be delivered in (multiples of) complete scanlines at a time.
comptable. Variable in class java.awt.GridBagLayout
concat(String). Method in class java.lang.String
Concatenates the specified string to the end of this String.
connect(). Method in class java.net.URLConnection
URLConnection objects go through two phases: first they are created, then they are connected.
connect(InetAddress, int). Method in class java.net.SocketImpl
Connects the socket to the specified address on the specified port.
connect(PipedInputStream). Method in class java.io.PipedOutputStream
Connect this output stream to a receiver.
connect(PipedOutputStream). Method in class java.io.PipedInputStream
Connects this input stream to a sender.
connect(String, int). Method in class java.net.SocketImpl
Connects the socket to the specified port on the specified host.
connected. Variable in class java.net.URLConnection
consumer. Variable in class java.awt.image.ImageFilter
The consumer of the particular image data stream for which this instance of the ImageFilter is filtering data.
cont(). Method in class sun.tools.debug.RemoteThread
Resume this thread from a breakpoint, unless it previously suspended.
contains(Object). Method in class java.util.Hashtable
Returns true if the specified object is an element of the hashtable.
contains(Object). Method in class java.util.Vector
Returns true if the specified object is a value of the collection.
containsKey(Object). Method in class java.util.Hashtable
Returns true if the collection contains an element for the key.
ContentHandler(). Constructor for class java.net.ContentHandler
controlDown(). Method in class java.awt.Event
Checks if the control key is down.
copyArea(int, int, int, int, int, int). Method in class java.awt.Graphics
Copies an area of the screen.
copyInto(Object[]). Method in class java.util.Vector
Copies the elements of this vector into the specified array.
copyValueOf(char[]). Static method in class java.lang.String
Returns a String that is equivalent to the specified character array.
copyValueOf(char[], int, int). Static method in class java.lang.String
Returns a String that is equivalent to the specified character array.
cos(double). Static method in class java.lang.Math
Returns the trigonometric cosine of an angle.
count. Variable in class java.io.BufferedInputStream
The number of bytes in the buffer.
count. Variable in class java.io.BufferedOutputStream
The number of bytes in the buffer.
count. Variable in class java.io.ByteArrayInputStream
The number of characters to use in the buffer.
count. Variable in class java.io.ByteArrayOutputStream
The number of bytes in the buffer.
count. Variable in class java.io.StringBufferInputStream
The number of characters to use in the buffer.
countComponents(). Method in class java.awt.Container
Returns the number of components in this panel.
countItems(). Method in class java.awt.Choice
Returns the number of items in this Choice.
countItems(). Method in class java.awt.List
Returns the number of items in the list.

countItems(). Method in class java.awt.Menu
Returns the number of elements in this menu.
countMenus(). Method in class java.awt.MenuBar
Counts the number of menus on the menu bar.
countObservers(). Method in class java.util.Observable
Counts the number of observers.
countStackFrames(). Method in class java.lang.Thread
Returns the number of stack frames in this Thread.
countTokens(). Method in class java.util.StringTokenizer
Returns the next number of tokens in the String using the current deliminter set.
create(). Method in class java.awt.Graphics
Creates a new Graphics Object that is a copy of the original Graphics Object.
create(boolean). Method in class java.net.SocketImpl
Creates a socket with a boolean that specifies whether this is a stream socket or a datagram socket.
create(int, int, int, int). Method in class java.awt.Graphics
Creates a new Graphics Object with the specified parameters, based on the original Graphics Object.
createButton(Button). Method in class java.awt.Toolkit
Uses the specified Peer interface to create a new Button.
createCanvas(Canvas). Method in class java.awt.Toolkit
Uses the specified Peer interface to create a new Canvas.
createCheckbox(Checkox). Method in class java.awt.Toolkit
Uses the specified Peer interface to create a new Checkbox.
createCheckboxMenuItem(CheckboxMenuItem). Method in class java.awt.Toolkit
Uses the specified Peer interface to create a new CheckboxMenuItem.
createChoice(Choice). Method in class java.awt.Toolkit
Uses the specified Peer interface to create a new Choice.
createContentHandler(String). Method in interface java.net. ContentHandlerFactory
Creates a new ContentHandler to read an object from a URLStreamHandler.
createDialog(Dialog). Method in class java.awt.Toolkit
Uses the specified Peer interface to create a new Dialog.
createFileDialog(FileDialog). Method in class java.awt.Toolkit
Uses the specified Peer interface to create a new FileDialog.
createFrame(Frame). Method in class java.awt.Toolkit
Uses the specified Peer interface to create a new Frame.
createImage(ImageProducer). Method in class java.awt.Component
Creates an image from the specified image producer.
createImage(ImageProducer). Method in interface java.awt.peer.ComponentPeer
createImage(ImageProducer). Method in class java.awt. Toolkit
Creates an image with the specified image producer.
createImage(int, int). Method in class java.awt.Component
Creates an off-screen drawable Image to be used for double buffering.
createImage(int, int). Method in interface java.awt.peer.ComponentPeer
createLabel(Label). Method in class java.awt.Toolkit
Uses the specified Peer interface to create a new Label.
createList(List). Method in class java.awt.Toolkit
Uses the specified Peer interface to create a new List.
createMenu(Menu). Method in class java.awt.Toolkit
Uses the specified Peer interface to create a new Menu.
createMenuBar(MenuBar). Method in class java.awt.Toolkit
Uses the specified Peer interface to create a new MenuBar.
createMenuItem(MenuItem). Method in class java.awt.Toolkit
Uses the specified Peer interface to create a new MenuItem.
createPanel(Panel). Method in class java.awt.Toolkit
Uses the specified Peer interface to create a new Panel.
createScrollbar(Scrollbar). Method in class java.awt.Toolkit
Uses the specified Peer interface to create a new Scrollbar.

createSocketImpl(). Method in interface java.net.SocketImplFactory
Creates a new SocketImpl instance.
createTextArea(TextArea). Method in class java.awt.Toolkit
Uses the specified Peer interface to create a new TextArea.
createTextField(TextField). Method in class java.awt.Toolkit
Uses the specified Peer interface to create a new TextField.
createURLStreamHandler(String). Method in interface java.net. URLStreamHandlerFactory
Creates a new URLStreamHandler instance with the specified protocol.
createWindow(Window). Method in class java.awt.Toolkit
Uses the specified Peer interface to create a new Window.
CropImageFilter(int, int, int, int). Constructor for class java.awt.image. CropImageFilter
Constructs a CropImageFilter that extracts the absolute rectangular region of pixels from its source Image as specified by the x, y, w, and h parameters.
CROSSHAIR_CURSOR. Static variable in class java.awt.Frame
CTRL_MASK. Static variable in class java.awt.Event
The control modifier constant.
currentClassLoader(). Method in class java.lang.SecurityManager
The current ClassLoader on the execution stack.
currentThread(). Static method in class java.lang.Thread
Returns a reference to the currently executing Thread object.
currentTimeMillis(). Static method in class java.lang.System
Returns the current time in milliseconds GMT since the epoch (00:00:00 UTC, January 1, 1970).
cyan. Static variable in class java.awt.Color
The color cyan.

D

darker().Method in class java.awt.Color
Returns a darker version of this color.
darkGray. Static variable in class java.awt.Color
The color dark gray.
DatagramPacket(byte[], int). Constructor for class java.net.DatagramPacket
This constructor is used to create a DatagramPacket object used for receiving datagrams.
DatagramPacket(byte[], int, InetAddress, int). Constructor for class java.net.DatagramPacket
This constructor is used construct the DatagramPacket to be sent.
DatagramSocket(). Constructor for class java.net.DatagramSocket
Creates a datagram socket
DatagramSocket(int). Constructor for class java.net.DatagramSocket
Creates a datagram socket
DataInputStream(InputStream). Constructor for class java.io.DataInputStream
Creates a new DataInputStream.
DataOutputStream(OutputStream). Constructor for class java.io.DataOutputStream
Creates a new DataOutputStream.
Date(). Constructor for class java.util.Date
Creates today's date/time.
Date(int, int, int). Constructor for class java.util.Date
Creates a date.
Date(int, int, int, int, int). Constructor for class java.util.Date
Creates a date.
Date(int, int, int, int, int, int). Constructor for class java.util.Date
Creates a date.
Date(long). Constructor for class java.util.Date
Creates a date.

Date(String). Constructor for class java.util.Date
Creates a date from a string according to the syntax accepted by parse().
DEFAULT_CURSOR. Static variable in class java.awt.Frame
defaultConstraints. Variable in class java.awt. GridBagLayout
defaults. Variable in class java.util.Properties
defineClass(byte[], int, int). Method in class java.lang.ClassLoader
Converts an array of bytes to an instance of class Class.
delete(). Method in class java.io.File
Deletes the specified file.
deleteObserver(Observer). Method in class java.util.Observable
Deletes an observer from the observer list.
deleteObservers(). Method in class java.util.Observable
Deletes observers from the observer list.
delItem(int). Method in class java.awt.List
Delete an item from the list.
delItem(int). Method in interface java.awt.peer.MenuPeer
delItems(int, int). Method in class java.awt.List
Delete multiple items from the list.
delItems(int, int). Method in interface java.awt.peer.ListPeer
deliverEvent(Event). Method in class java.awt.Component
Delivers an event to this component or one of its sub components.
deliverEvent(Event). Method in class java.awt.Container
Delivers an event.
delMenu(int). Method in interface java.awt.peer.MenuBarPeer
description(). Method in class sun.tools.debug. RemoteArray
Return a description of the array.
description(). Method in class sun.tools.debug.RemoteClass
Return a (somewhat verbose) description.
description(). Method in class sun.tools.debug.RemoteObject
Return a description of the object.
description(). Method in class sun.tools.debug.RemoteString
Return the string value, or "null"
description(). Method in class sun.tools.debug.RemoteValue
Return a description of the RemoteValue.
deselect(int). Method in class java.awt.List
Deselects the item at the specified index.
deselect(int). Method in interface java.awt.peer.ListPeer
destroy(). Method in class java.applet.Applet
Cleans up whatever resources are being held.
destroy(). Method in class java.lang.Process
Kills the subprocess.
destroy(). Method in class java.lang.Thread
Destroy a thread, without any cleanup, i.e.
destroy(). Method in class java.lang.ThreadGroup
Destroys a Thread group.
Dialog(Frame, boolean). Constructor for class java.awt.Dialog
Constructs an initially invisible Dialog.
Dialog(Frame, String, boolean). Constructor for class java.awt.Dialog
Constructs an initially invisible Dialog with a title.
Dictionary(). Constructor for class java.util.Dictionary
digit(char, int). Static method in class java.lang. Character
Returns the numeric value of the character digit using the specified radix.
Dimension(). Constructor for class java.awt.Dimension
Constructs a Dimension with a 0 width and 0 height.
Dimension(Dimension). Constructor for class java.awt.Dimension
Constructs a Dimension and initializes it to the specified value.

Dimension(int, int). Constructor for class java.awt.Dimension
Constructs a Dimension and initializes it to the specified width and specified height.
DirectColorModel(int, int, int, int). Constructor for class java.awt.image. DirectColorModel
Constructs a DirectColorModel from the given masks specifying which bits in the pixel contain the red, green and blue color components.
DirectColorModel(int, int, int, int, int). Constructor for class java.awt.image. DirectColorModel
Constructs a DirectColorModel from the given masks specifying which bits in the pixel contain the alhpa, red, green and blue color components.
disable(). Static method in class java.lang.Compiler
disable(). Method in class java.awt.Component
Disables a component.
disable(). Method in interface java.awt.peer.ComponentPeer
disable(). Method in class java.awt.MenuItem
Makes this menu item unselectable by the user.
disable(). Method in interface java.awt.peer.MenuItemPeer
dispose(). Method in interface java.awt.peer. ComponentPeer
dispose(). Method in class java.awt.Frame
Disposes of the Frame.
dispose(). Method in class java.awt.Graphics
Disposes of this graphics context.
dispose(). Method in interface java.awt.peer.MenuComponentPeer
dispose(). Method in class java.awt.Window
Disposes of the Window.
doInput. Variable in class java.net.URLConnection
doOutput. Variable in class java.net.URLConnection
Double(double). Constructor for class java.lang.Double
Constructs a Double wrapper for the specified double value.
Double(String). Constructor for class java.lang.Double
Constructs a Double object initialized to the value specified by the String parameter.
doubleToLongBits(double). Static method in class java.lang.Double
Returns the bit represention of a double-float value
doubleValue(). Method in class java.lang.Double
Returns the double value of this Double.
doubleValue(). Method in class java.lang.Float
Returns the double value of this Float.
doubleValue(). Method in class java.lang.Integer
Returns the value of this Integer as a double.
doubleValue(). Method in class java.lang.Long
Returns the value of this Long as a double.
doubleValue(). Method in class java.lang.Number
Returns the value of the number as a double.
DOWN. Static variable in class java.awt.Event
The down arrow key.
down(int). Method in class sun.tools.debug.RemoteThread
Change the current stackframe to be one or more frames lower (as in, toward the current program counter).
draw3DRect(int, int, int, int, boolean). Method in class java.awt.Graphics
Draws a highlighted 3-D rectangle.
drawArc(int, int, int, int, int, int). Method in class java.awt.Graphics
Draws an arc bounded by the specified rectangle from startAngle to endAngle.
drawBytes(byte[], int, int, int, int). Method in class java.awt.Graphics
Draws the specified bytes using the current font and color.
drawChars(char[], int, int, int, int). Method in class java.awt.Graphics
Draws the specified characters using the current font and color.
drawImage(Image, int, int, Color, ImageObserver). Method in class java.awt. Graphics
Draws the specified image at the specified coordinate (x, y), with the given solid background Color.

drawImage(Image, int, int, ImageObserver). Method in class java.awt.Graphics
Draws the specified image at the specified coordinate (x, y).
drawImage(Image, int, int, int, int, Color, ImageObserver). Method in class java.awt.Graphics
Draws the specified image inside the specified rectangle, with the given solid background Color.
drawImage(Image, int, int, int, int, ImageObserver). Method in class java.awt. Graphics
Draws the specified image inside the specified rectangle.
drawLine(int, int, int, int). Method in class java.awt.Graphics
Draws a line between the coordinates (x1,y1) and (x2,y2).
drawOval(int, int, int, int). Method in class java.awt.Graphics
Draws an oval inside the specified rectangle using the current color.
drawPolygon(int[], int[], int). Method in class java.awt.Graphics
Draws a polygon defined by an array of x points and y points.
drawPolygon(Polygon). Method in class java.awt.Graphics
Draws a polygon defined by the specified point.
drawRect(int, int, int, int). Method in class java.awt.Graphics
Draws the outline of the specified rectangle using the current color.
drawRoundRect(int, int, int, int, int, int). Method in class java.awt.Graphics
Draws an outlined rounded corner rectangle using the current color.
drawString(String, int, int). Method in class java.awt.Graphics
Draws the specified String using the current font and color.
DumpConstraints(GridBagConstraints). Method in class java.awt.GridBagLayout
Print the layout constraints.
DumpLayoutInfo(GridBagLayoutInfo). Method in class java.awt.GridBagLayout
Print the layout information.
dumpStack(). Method in class sun.tools.debug.RemoteThread
Dump the stack.
dumpStack(). Static method in class java.lang.Thread
A debugging procedure to print a stack trace for the current Thread.

E

E.Static variable in class java.lang.Math
The float representation of the value E.
E_RESIZE_CURSOR. Static variable in class java.awt.Frame
EAST. Static variable in class java.awt. GridBagConstraints
echoCharIsSet(). Method in class java.awt.TextField
Returns true if this TextField has a character set for echoing.
elementAt(int). Method in class java.util.Vector
Returns the element at the specified index.
elementCount. Variable in class java.util.Vector
The number of elements in the buffer.
elementData. Variable in class java.util.Vector
The buffer where elements are stored.
elements(). Method in class java.util.Dictionary
Returns an enumeration of the elements.
elements(). Method in class java.util.Hashtable
Returns an enumeration of the elements.
elements(). Method in class java.util.Vector
Returns an enumeration of the elements.
empty(). Method in class java.util.Stack
Returns true if the stack is empty.
EmptyStackException(). Constructor for class java.util.EmptyStackException
Constructs a new EmptyStackException with no detail message.

enable(). Static method in class java.lang.Compiler
enable(). Method in class java.awt.Component
Enables a component.
enable(). Method in interface java.awt.peer.ComponentPeer
enable(). Method in class java.awt.MenuItem
Makes this menu item selectable by the user.
enable(). Method in interface java.awt.peer.MenuItemPeer
enable(boolean). Method in class java.awt.Component
Conditionally enables a component.
enable(boolean). Method in class java.awt.MenuItem
Conditionally enables a component.
encode(String). Static method in class java.net.URLEncoder
Translates String into x-www-form-urlencoded format.
END. Static variable in class java.awt.Event
The end key.
endsWith(String). Method in class java.lang.String
Determines whether the String ends with some suffix.
ensureCapacity(int). Method in class java.lang.StringBuffer
Ensures that the capacity of the buffer is at least equal to the specified minimum.
ensureCapacity(int). Method in class java.util.Vector
Ensures that the vector has at least the specified capacity.
enumerate(Thread[]). Static method in class java.lang.Thread
Copies, into the specified array, references to every active Thread in this Thread's group.
enumerate(Thread[]). Method in class java.lang.ThreadGroup
Copies, into the specified array, references to every active Thread in this Thread group.
enumerate(Thread[], boolean). Method in class java.lang.ThreadGroup
Copies, into the specified array, references to every active Thread in this Thread group.
enumerate(ThreadGroup[]). Method in class java.lang.ThreadGroup
Copies, into the specified array, references to every active Thread group in this Thread group.
enumerate(ThreadGroup[], boolean). Method in class java.lang.ThreadGroup
Copies, into the specified array, references to every active Thread group in this Thread group.
EOFException(). Constructor for class java.io.EOFException
Constructs an EOFException with no detail message.
EOFException(String). Constructor for class java.io.EOFException
Constructs an EOFException with the specified detail message.
eolIsSignificant(boolean). Method in class java.io.StreamTokenizer
If the flag is true, end-of-lines are significant (TT_EOL will be returned by nexttoken).
equals(Object). Method in class java.util.BitSet
Compares this object against the specified object.
equals(Object). Method in class java.lang.Boolean
Compares this object against the specified object.
equals(Object). Method in class java.lang.Character
Compares this object against the specified object.
equals(Object). Method in class java.awt.Color
Compares this object against the specified object.
equals(Object). Method in class java.util.Date
Compares this object against the specified object.
equals(Object). Method in class java.lang.Double
Compares this object against the specified object.
equals(Object). Method in class java.io.File
Compares this object against the specified object.
equals(Object). Method in class java.lang.Float
Compares this object against some other object.
equals(Objet). Method in class java.awt.Font
Compares this object to the specifed object.
equals(Object). Method in class java.net.InetAddress

Compares this object against the specified object.
equals(Object). Method in class java.lang.Integer
Compares this object to the specified object.
equals(Object). Method in class java.lang.Long
Compares this object against the specified object.
equals(Object). Method in class java.lang.Object
Compares two Objects for equality.
equals(Object). Method in class java.awt.Point
Checks whether two pointers are equal.
equals(Object). Method in class java.awt.Rectangle
Checks whether two rectangles are equal.
equals(Object). Method in class java.lang.String
Compares this String to the specified object.
equals(Object). Method in class java.net.URL
Compares two URLs.
equalsIgnoreCase(String). Method in class java.lang.String
Compares this String to another object.
err. Static variable in class java.io.FileDescriptor
Handle to standard error.
err. Static variable in class java.lang.System
Standard error stream.
ERROR. Static variable in interface java.awt.image.ImageObserver
An image which was being tracked asynchronously has encountered an error.
Error(). Constructor for class java.lang.Error
Constructs an Error with no specified detail message.
Error(String). Constructor for class java.lang.Error
Constructs an Error with the specified detail message.
ERRORED. Static variable in class java.awt.MediaTracker
Flag indicating the download of some media encountered an error.
Event(Object, int, Object). Constructor for class java.awt.Event
Constructs an event with the specified target component, event type, and argument.
Event(Object, long, int, int, int, int, int). Constructor for class java.awt. Event
Constructs an event with the specified target component, time stamp, event type, x and y coordinates, keyboard key, state of the modifier keys and an argument set to null.
Event(Object, long, int, int, int, int, int, Object). Constructor for class java.awt.Event
Constructs an event with the specified target component, time stamp, event type, x and y coordinates, keyboard key, state of the modifier keys and argument.
evt. Variable in class java.awt.Event
The next event.
Exception(). Constructor for class java.lang.Exception
Constructs an Exception with no specified detail message.
Exception(String). Constructor for class java.lang.Exception
Constructs a Exception with the specified detail message.
exceptionEvent(RemoteThread, String). Method in interface sun.tools.debug. DebuggerCallback
An exception has occurred.
exec(String). Method in class java.lang.Runtime
Executes the system command specified in the parameter.
exec(String, String[]). Method in class java.lang.Runtime
Executes the system command specified in the parameter.
exec(String[]). Method in class java.lang.Runtime
Executes the system command specified by cmdarray[0] with arguments specified by the strings in the rest of the array.
exec(String[], String[]). Method in class java.lang.Runtime
Executes the system command specified by cmdarray[0] with arguments specified by the strings in the rest of the array.
exists(). Method in class java.io.File

Returns a boolean indicating whether or not a file exists.
exit(int). Method in class java.lang.Runtime
Exits the virtual machine with an exit code.
exit(int). Static method in class java.lang.System
Exits the virtual machine with an exit code.
exitValue(). Method in class java.lang.Process
Returns the exit value for the subprocess.
exp(double). Static method in class java.lang.Math
Returns the exponential number e(2.718...) raised to the power of a.

F

F1.Static variable in class java.awt.Event
The F1 function key.
F10. Static variable in class java.awt.Event
The F10 function key.
F11. Static variable in class java.awt.Event
The F11 function key.
F12. Static variable in class java.awt.Event
The F12 function key.
F2. Static variable in class java.awt.Event
The F2 function key.
F3. Static variable in class java.awt.Event
The F3 function key.
F4. Static variable in class java.awt.Event
The F4 function key.
F5. Static variable in class java.awt.Event
The F5 function key.
F6. Static variable in class java.awt.Event
The F6 function key.
F7. Static variable in class java.awt.Event
The F7 function key.
F8. Static variable in class java.awt.Event
The F8 function key.
F9. Static variable in class java.awt.Event
The F9 function key.
FALSE. Static variable in class java.lang.Boolean
Assigns this Boolean to be false.
fd. Variable in class java.net.SocketImpl
The file descriptor object
File(File, String). Constructor for class java.io.File
Creates a File object (given a directory File object).
File(String). Constructor for class java.io.File
Creates a File object.
File(String, String). Constructor for class java.io.File
Creates a File object from the specified directory.
FileDescriptor(). Constructor for class java.io.FileDescriptor
FileDialog(Frame, String). Constructor for class java.awt.FileDialog
Creates a file dialog for loading a file.
FileDialog(Frame, String, int). Constructor for class java.awt.FileDialog
Creates a file dialog with the specified title and mode.
FileInputStream(File). Constructor for class java.io.FileInputStream
Creates an input file from the specified File object.

FileInputStream(FileDescriptor). Constructor for class java.io.FileInputStream
FileInputStream(String). Constructor for class java.io.FileInputStream
Creates an input file with the specified system dependent file name.
FileNotFoundException(). Constructor for class java.io.FileNotFoundException
Constructs a FileNotFoundException with no detail message.
FileNotFoundException(String). Constructor for class java.io. FileNotFoundException
Constructs a FileNotFoundException with the specified detail message.
FileOutputStream(File). Constructor for class java.io.FileOutputStream
Creates an output file with the specified File object.
FileOutputStream(FileDescriptor). Constructor for class java.io. FileOutputStream
FileOutputStream(String). Constructor for class java.io.FileOutputStream
Creates an output file with the specified system dependent file name.
fill. Variable in class java.awt.GridBagConstraints
fill3DRect(int, int, int, int, boolean). Method in class java.awt.Graphics
Paints a highlighted 3-D rectangle using the current color.
fillArc(int, int, int, int, int, int). Method in class java.awt.Graphics
Fills an arc using the current color.
fillInStackTrace(). Method in class java.lang.Throwable
Fills in the excecution stack trace.
fillOval(int, int, int, int). Method in class java.awt.Graphics
Fills an oval inside the specified rectangle using the current color.
fillPolygon(int[], int[], int). Method in class java.awt.Graphics
Fills a polygon with the current color using an even-odd fill rule (otherwise known as an alternating rule).
fillPolygon(Polygon). Method in class java.awt.Graphics
Fills the specified polygon with the current color using an even-odd fill rule (otherwise known as an alternating rule).
fillRect(int, int, int, int). Method in class java.awt.Graphics
Fills the specified rectangle with the current color.
fillRoundRect(int, int, int, int, int, int). Method in class java.awt.Graphics
Draws a rounded rectangle filled in with the current color.
FilteredImageSource(ImageProducer, ImageFilter). Constructor for class java.awt.image.FilteredImageSource
Constructs an ImageProducer object from an existing ImageProducer and a filter object.
filterIndexColorModel(IndexColorModel). Method in class java.awt.image. RGBImageFilter
Filters an IndexColorModel object by running each entry in its color tables through the filterRGB function that RGBImageFilter subclasses must provide.
FilterInputStream(InputStream). Constructor for class java.io.FilterInputStream
Creates an input stream filter.
FilterOutputStream(OutputStream). Constructor for class java.io. FilterOutputStream
Creates an output stream filter.
filterRGB(int, int, int). Method in class java.awt.image.RGBImageFilter
Subclasses must specify a method to convert a single input pixel in the default RGB ColorModel to a single output pixel.
filterRGBPixels(int, int, int, int, int[], int, int). Method in class java.awt.image.RGBImageFilter
Filters a buffer of pixels in the default RGB ColorModel by passing them one by one through the filterRGB method.
finalize(). Method in class java.net.DatagramSocket
Code to perform when this object is garbage collected.
finalize(). Method in class java.io.FileInputStream
Closes the stream when garbage is collected.
finalize(). Method in class java.io.FileOutputStream
Closes the stream when garbage is collected.
finalize(). Method in class java.awt.Graphics
Disposes of this graphics context once it is no longer referenced.
finalize(). Method in class java.lang.Object

Code to perform when this object is garbage collected.
findClass(String). Method in class sun.tools.debug.RemoteDebugger
Find a specified class.
findSystemClass(String). Method in class java.lang.ClassLoader
Loads a system Class.
first(Container). Method in class java.awt.CardLayout
Flip to the first card.
firstElement(). Method in class java.util.Vector
Returns the first element of the sequence.
Float(double). Constructor for class java.lang.Float
Constructs a Float wrapper for the specified double value.
Float(float). Constructor for class java.lang.Float
Constructs a Float wrapper for the specified float value.
Float(String). Constructor for class java.lang.Float
Constructs a Float object initialized to the value specified by the String parameter.
floatToIntBits(float). Static method in class java.lang.Float
Returns the bit represention of a single-float value
floatValue(). Method in class java.lang.Double
Returns the float value of this Double.
floatValue(). Method in class java.lang.Float
Returns the float value of this Float object.
floatValue(). Method in class java.lang.Integer
Returns the value of this Integer as a float.
floatValue(). Method in class java.lang.Long
Returns the value of this Long as a float.
floatValue(). Method in class java.lang.Number
Returns the value of the number as a float.
floor(double). Static method in class java.lang.Math
Returns the "floor" or largest whole number less than or equal to a.
FlowLayout(). Constructor for class java.awt.FlowLayout
Constructs a new Flow Layout with a centered alignment.
FlowLayout(int). Constructor for class java.awt.FlowLayout
Constructs a new Flow Layout with the specified alignment.
FlowLayout(int, int, int). Constructor for class java.awt.FlowLayout
Constructs a new Flow Layout with the specified alignment and gap values.
flush(). Method in class java.io.BufferedOutputStream
Flushes the stream.
flush(). Method in class java.io.DataOutputStream
Flushes the stream.
flush(). Method in class java.io.FilterOutputStream
Flushes the stream.
flush(). Method in class java.awt.Image
Flushes all resources being used by this Image object.
flush(). Method in class java.io.OutputStream
Flushes the stream.
flush(). Method in class java.io.PrintStream
Flushes the stream.
font. Variable in class java.awt.FontMetrics
The actual font.
Font(String, int, int). Constructor for class java.awt.Font
Creates a new font with the specified name, style and point size.
FontMetrics(Font). Constructor for class java.awt.FontMetrics
Creates a new FontMetrics object with the specified font.
forDigit(int, int). Static method in class java.lang.Character
Returns the character value for the specified digit in the specified radix.
forName(String). Static method in class java.lang.Class

Returns the runtime Class descriptor for the specified Class.
Frame(). Constructor for class java.awt.Frame
Constructs a new Frame that is initially invisible.
Frame(String). Constructor for class java.awt.Frame
Constructs a new, initially invisible Frame with the specified title.
FRAMEBITS. Static variable in interface java.awt.image.ImageObserver
Another complete frame of a multi-frame image which was previously drawn is now available to be drawn again.
freeMemory(). Method in class sun.tools.debug.RemoteDebugger
Report the free memory available to the Java interpreter being debugged.
freeMemory(). Method in class java.lang.Runtime
Returns the number of free bytes in system memory.
fromHex(String). Static method in class sun.tools.debug.RemoteValue
Convert hexadecimal strings to ints.

G

gc().Method in class java.lang.Runtime
Runs the garbage collector.
gc(). Static method in class java.lang.System
Runs the garbage collector.
gc(RemoteObject[]). Method in class sun.tools.debug.RemoteDebugger
Free all objects referenced by the debugger.
get(). Method in class sun.tools.debug.RemoteBoolean
Return the boolean's value.
get(). Method in class sun.tools.debug.RemoteByte
Return the byte's value.
get(). Method in class sun.tools.debug.RemoteChar
Return the char's value.
get(). Method in class sun.tools.debug.RemoteDouble
Return the double's value.
get(). Method in class sun.tools.debug.RemoteFloat
Return the float's value.
get(). Method in class sun.tools.debug.RemoteInt
Return the int's value.
get(). Method in class sun.tools.debug.RemoteLong
Return the long's value.
get(). Method in class sun.tools.debug.RemoteShort
Return the short's value.
get(int). Method in class java.util.BitSet
Gets a bit.
get(Integer). Method in class sun.tools.debug.RemoteDebugger
Get an object from the remote object cache.
get(Object). Method in class java.util.Dictionary
Gets the object associated with the specified key in the Dictionary.
get(Object). Method in class java.util.Hashtable
Gets the object associated with the specified key in the hashtable.
getAbsolutePath(). Method in class java.io.File
Gets the absolute path of the file.
getAddress(). Method in class java.net.DatagramPacket
getAddress(). Method in class java.net.InetAddress
Returns the raw IP address in network byte order.
getAlignment(). Method in class java.awt.Label

Gets the current alignment of this label.
getAllByName(String). Static method in class java.net.InetAddress
Given a hostname, returns an array of all the corresponding InetAddresses.
getAllowUserInteraction(). Method in class java.net.URLConnection
getAlpha(int). Method in class java.awt.image. ColorModel
The subclass must provide a function which provides the alpha color compoment for the specified pixel.
getAlpha(int). Method in class java.awt.image.DirectColorModel
Return the alpha transparency value for the specified pixel in the range 0-255.
getAlpha(int). Method in class java.awt.image.IndexColorModel
Returns the alpha transparency value for the specified pixel in the range 0-255.
getAlphaMask(). Method in class java.awt.image.DirectColorModel
Returns the mask indicating which bits in a pixel contain the alpha transparency component.
getAlphas(byte[]). Method in class java.awt.image.IndexColorModel
Copies the array of alpha transparency values into the given array.
getApplet(String). Method in interface java.applet.AppletContext
Gets an applet by name.
getAppletContext(). Method in class java.applet.Applet
Gets a handle to the applet context.
getAppletContext(). Method in interface java.applet.AppletStub
Gets a handler to the applet's context.
getAppletInfo(). Method in class java.applet.Applet
Returns a string containing information about the author, version and copyright of the applet.
getApplets(). Method in interface java.applet.AppletContext
Enumerates the applets in this context.
getAscent(). Method in class java.awt.FontMetrics
Gets the font ascent.
getAudioClip(URL). Method in class java.applet.Applet
Gets an audio clip.
getAudioClip(URL). Method in interface java.applet.AppletContext
Gets an audio clip.
getAudioClip(URL, String). Method in class java.applet.Applet
Gets an audio clip.
getBackground(). Method in class java.awt.Component
Gets the background color.
getBlue(). Method in class java.awt.Color
Gets the blue component.
getBlue(int). Method in class java.awt.image.ColorModel
The subclass must provide a function which provides the blue color compoment for the specified pixel.
getBlue(int). Method in class java.awt.image.DirectColorModel
Returns the blue color compoment for the specified pixel in the range 0-255.
getBlue(int). Method in class java.awt.image.IndexColorModel
Returns the blue color compoment for the specified pixel in the range 0-255.
getBlueMask(). Method in class java.awt.image.DirectColorModel
Returns the mask indicating which bits in a pixel contain the blue color component.
getBlues(byte[]). Method in class java.awt.image.IndexColorModel
Copies the array of blue color components into the given array.
getBoolean(String). Static method in class java.lang.Boolean
Gets a Boolean from the properties.
getBoundingBox(). Method in class java.awt.Polygon
Determines the area spanned by this Polygon.
getByName(String). Static method in class java.net.InetAddress
Returns a network address for the indicated host.
getBytes(int, int, byte[], int). Method in class java.lang.String
Copies characters from this String into the specified byte array.

getChars(int, int, char[], int). Method in class java.lang.String
Copies characters from this String into the specified haracter array.
getChars(int, int, char[], int). Method in class java.lang.StringBuffer
Copies the characters of the specified substring (determined by srcBegin and srcEnd) into the character array, starting at the array's dstBegin location.
getCheckboxGroup(). Method in class java.awt.Checkbox
Returns the checkbox group.
getClass(). Method in class java.lang.Object
Returns the Class of this Object.
getClassContext(). Method in class java.lang.SecurityManager
Gets the context of this Class.
getClassLoader(). Method in class java.lang.Class
Returns the Class loader of this Class.
getClassLoader(). Method in class sun.tools.debug.RemoteClass
Return the classloader for this class.
getClazz(). Method in class sun.tools.debug.RemoteObject
Returns the object's class.
getClipRect(). Method in class java.awt.Graphics
Returns the bounding rectangle of the current clipping area.
getCodeBase(). Method in class java.applet.Applet
Gets the base URL.
getCodeBase(). Method in interface java.applet.AppletStub
Gets the base URL.
getColor(). Method in class java.awt.Graphics
Gets the current color.
getColor(String). Static method in class java.awt.Color
Gets the specified Color property.
getColor(String, Color). Static method in class java.awt.Color
Gets the specified Color property of the specified Color.
getColor(String, int). Static method in class java.awt.Color
Gets the specified Color property of the color value.
getColorModel(). Method in class java.awt.Component
Gets the ColorModel used to display the component on the output device.
getColorModel(). Method in interface java.awt.peer.ComponentPeer
getColorModel(). Method in class java.awt.Toolkit
Returns the ColorModel of the screen.
getColumns(). Method in class java.awt.TextArea
Returns the number of columns in the TextArea.
getColumns(). Method in class java.awt.TextField
Returns the number of columns in this TextField.
getComponent(int). Method in class java.awt.Container
Gets the nth component in this container.
getComponents(). Method in class java.awt.Container
Gets all the components in this container.
getConstraints(Component). Method in class java.awt.GridBagLayout
Retrieves the constraints for the specified component.
getContent(). Method in class java.net.URL
Gets the contents from this opened connection.
getContent(). Method in class java.net.URLConnection
Gets the object referred to by this URL.
getContent(URLConnection). Method in class java.net.ContentHandler
Given an input stream positioned at the beginning of the representation of an object, reads that stream and recreates the object from it.
getContentEncoding(). Method in class java.net.URLConnection
Gets the content encoding.
getContentLength(). Method in class java.net.URLConnection

Gets the content length.
getContentType(). Method in class java.net.URLConnection
Gets the content type.
getCurrent(). Method in class java.awt.CheckboxGroup
Gets the current choice.
getCurrentFrame(). Method in class sun.tools.debug.RemoteThread
Get the current stack frame.
getCurrentFrameIndex(). Method in class sun.tools.debug.RemoteThread
Return the current stackframe index
getCursorType(). Method in class java.awt.Frame
Return the cursor type
getData(). Method in class java.net.DatagramPacket
getDate(). Method in class java.util.Date
Returns the day of the month.
getDate(). Method in class java.net.URLConnection
Gets the sending date of the object.
getDay(). Method in class java.util.Date
Returns the day of the week.
getDefaultAllowUserInteraction(). Static method in class java.net. URLConnection
getDefaultRequestProperty(String). Static method in class java.net.URLConnection
getDefaultToolkit(). Static method in class java.awt. Toolkit
Returns the default toolkit.
getDefaultUseCaches(). Method in class java.net.URLConnection
Sets/gets the default value of the UseCaches flag.
getDescent(). Method in class java.awt.FontMetrics
Gets the font descent.
getDirectory(). Method in class java.awt.FileDialog
Gets the directory of the Dialog.
getDocumentBase(). Method in class java.applet.Applet
Gets the document URL.
getDocumentBase(). Method in interface java.applet.AppletStub
Gets the document URL.
getDoInput(). Method in class java.net.URLConnection
getDoOutput(). Method in class java.net.URLConnection
getEchoChar(). Method in class java.awt.TextField
Returns the character to be used for echoing.
getElement(int). Method in class sun.tools.debug.RemoteArray
Return an array element.
getElements(). Method in class sun.tools.debug.RemoteArray
Returns a copy of the array as instances of RemoteValue.
getElements(int, int). Method in class sun.tools.debug.RemoteArray
Returns a copy of a portion of the array as instances of RemoteValue.
getElementType(). Method in class sun.tools.debug.RemoteArray
Return the element type as a "TC_" constant, such as "TC_CHAR".
getenv(String). Static method in class java.lang.System
Obsolete.
getErrorsAny(). Method in class java.awt.MediaTracker
Returns a list of all media that have encountered an error.
getErrorsID(int). Method in class java.awt.MediaTracker
Returns a list of media with the specified ID that have encountered an error.
getErrorStream(). Method in class java.lang.Process
Returns the an InputStream connected to the error stream of the child process.
getExceptionCatchList(). Method in class sun.tools.debug.RemoteDebugger
Return the list of the exceptions the debugger will stop on.
getExpiration(). Method in class java.net.URLConnection
Gets the expriation date of the object.

getFamily(). Method in class java.awt.Font
Gets the platform specific family name of the font.
getFD(). Method in class java.io.FileInputStream
Returns the opaque file descriptor object associated with this stream.
getFD(). Method in class java.io.FileOutputStream
Returns the file descriptor associated with this stream.
getFD(). Method in class java.io.RandomAccessFile
Returns the opaque file descriptor object.
getField(int). Method in class sun.tools.debug.RemoteClass
Return the static field, specified by index.
getField(int). Method in class sun.tools.debug.RemoteObject
Return an instance variable, specified by slot number.
getField(String). Method in class sun.tools.debug.RemoteClass
Return the static field, specified by name.
getField(String). Method in class sun.tools.debug.RemoteObject
Return an instance variable, specified by name.
getFields(). Method in class sun.tools.debug.RemoteClass
Return all the static fields for this class.
getFields(). Method in class sun.tools.debug.RemoteObject
Return the instance (non-static) fields of an object.
getFieldValue(int). Method in class sun.tools.debug.RemoteClass
Return the value of a static field, specified by its index
getFieldValue(int). Method in class sun.tools.debug.RemoteObject
Returns the value of an object's instance variable.
getFieldValue(String). Method in class sun.tools.debug.RemoteClass
Return the value of a static field, specified by name.
getFieldValue(String). Method in class sun.tools.debug.RemoteObject
Returns the value of an object's instance variable.
getFile(). Method in class java.awt.FileDialog
Gets the file of the Dialog.
getFile(). Method in class java.net.URL
Gets the file name.
getFileDescriptor(). Method in class java.net.SocketImpl
getFilenameFilter(). Method in class java.awt. FileDialog
Gets the filter.
getFilePointer(). Method in class java.io.RandomAccessFile
Returns the current location of the file pointer.
getFilterInstance(ImageConsumer). Method in class java.awt.image.ImageFilter
Returns a unique instance of an ImageFilter object which will actually perform the filtering for the specified ImageConsumer.
getFont(). Method in class java.awt.Component
Gets the font of the component.
getFont(). Method in class java.awt.FontMetrics
Gets the font.
getFont(). Method in class java.awt.Graphics
Gets the current font.
getFont(). Method in class java.awt.MenuComponent
Gets the font used for this MenuItem.
getFont(). Method in interface java.awt.MenuContainer
getFont(String). Static method in class java.awt.Font
Gets a font from the system properties list.
getFont(String, Font). Static method in class java.awt.Font
Gets the specified font from the system properties list.
getFontList(). Method in class java.awt.Toolkit
Returns the names of the available fonts.
getFontMetrics(). Method in class java.awt.Graphics

Gets the current font metrics.
getFontMetrics(Font). Method in class java.awt.Component
Gets the font metrics for this component.
getFontMetrics(Font). Method in interface java.awt.peer.ComponentPeer
getFontMetrics(Font). Method in class java.awt. Graphics
Gets the current font metrics for the specified font.
getFontMetrics(Font). Method in class java.awt.Toolkit
Returns the screen metrics of the font.
getForeground(). Method in class java.awt.Component
Gets the foreground color.
getGraphics(). Method in class java.awt.Component
Gets a Graphics context for this component.
getGraphics(). Method in interface java.awt.peer.ComponentPeer
getGraphics(). Method in class java.awt.Image
Gets a graphics object to draw into this image.
getGreen(). Method in class java.awt.Color
Gets the green component.
getGreen(int). Method in class java.awt.image.ColorModel
The subclass must provide a function which provides the green color compoment for the specified pixel.
getGreen(int). Method in class java.awt.image.DirectColorModel
Returns the green color compoment for the specified pixel in the range 0-255.
getGreen(int). Method in class java.awt.image.IndexColorModel
Returns the green color compoment for the specified pixel in the range 0-255.
getGreenMask(). Method in class java.awt.image.DirectColorModel
Returns the mask indicating which bits in a pixel contain the green color component.
getGreens(byte[]). Method in class java.awt.image.IndexColorModel
Copies the array of green color components into the given array.
getHeaderField(int). Method in class java.net.URLConnection
Returns the value for the nth header field.
getHeaderField(String). Method in class java.net.URLConnection
Gets a header field by name.
getHeaderFieldDate(String, long). Method in class java.net.URLConnection
Gets a header field by name.
getHeaderFieldInt(String, int). Method in class java.net.URLConnection
Gets a header field by name.
getHeaderFieldKey(int). Method in class java.net.URLConnection
Returns the key for the nth header field.
getHeight(). Method in class java.awt.FontMetrics
Gets the total height of the font.
getHeight(ImageObserver). Method in class java.awt.Image
Gets the actual height of the image.
getHelpMenu(). Method in class java.awt.MenuBar
Gets the help menu on the menu bar.
getHost(). Method in class java.net.URL
Gets the host name.
getHostName(). Method in class java.net.InetAddress
Gets the hostname for this address; also the key in the hashtable.
getHours(). Method in class java.util.Date
Returns the hour.
getHSBColor(float, float, float). Static method in class java.awt.Color
A static Color factory for generating a Color object from HSB values.
getIconImage(). Method in class java.awt.Frame
Returns the icon image for this Frame.
getId(). Method in class sun.tools.debug.RemoteObject
Returns the id of the object.

getIfModifiedSince(). Method in class java.net.URLConnection
getImage(String). Method in class java.awt.Toolkit
Returns an image which gets pixel data from the specified file.
getImage(URL). Method in class java.applet.Applet
Gets an image given a URL.
getImage(URL). Method in interface java.applet.AppletContext
Gets an image.
getImage(URL). Method in class java.awt.Toolkit
Returns an image which gets pixel data from the specified URL.
getImage(URL, String). Method in class java.applet.Applet
Gets an image relative to a URL.
getInCheck(). Method in class java.lang.SecurityManager
Returns whether there is a security check in progress.
getInetAddress(). Method in class java.net.ServerSocket
Gets the address to which the socket is connected.
getInetAddress(). Method in class java.net.Socket
Gets the address to which the socket is connected.
getInetAddress(). Method in class java.net.SocketImpl
getInputStream(). Method in class java.lang.Process
Returns a Stream connected to the output of the child process.
getInputStream(). Method in class java.net.Socket
Gets an InputStream for this socket.
getInputStream(). Method in class java.net.SocketImpl
Gets an InputStream for this socket.
getInputStream(). Method in class java.net.URLConnection
Calls this routine to get an InputStream that reads from the object.
getInstanceField(int). Method in class sun.tools.debug.RemoteClass
Return the instance field, specified by its index.
getInstanceFields(). Method in class sun.tools.debug.RemoteClass
Return all the instance fields for this class.
getInteger(String). Static method in class java.lang.Integer
Gets an Integer property.
getInteger(String, int). Static method in class java.lang.Integer
Gets an Integer property.
getInteger(String, Integer). Static method in class java.lang.Integer
Gets an Integer property.
getInterfaces(). Method in class java.lang.Class
Returns the interfaces of this Class.
getInterfaces(). Method in class sun.tools.debug.RemoteClass
Return the interfaces for this class.
getItem(int). Method in class java.awt.Choice
Returns the String at the specified index in the Choice.
getItem(int). Method in class java.awt.List
Gets the item associated with the specified index.
getItem(int). Method in class java.awt.Menu
Returns the item located at the specified index of this menu.
getLabel(). Method in class java.awt.Button
Gets the label of the button.
getLabel(). Method in class java.awt.Checkbox
Gets the label of the button.
getLabel(). Method in class java.awt.MenuItem
Gets the label for this menu item.
getLastModified(). Method in class java.net.URLConnection
Gets the last modified date of the object.
getLayout(). Method in class java.awt.Container
Gets the layout manager for this container.

getLayoutDimensions(). Method in class java.awt.GridBagLayout
GetLayoutInfo(Container, int). Method in class java.awt.GridBagLayout
getLayoutOrigin(). Method in class java.awt. GridBagLayout
getLayoutWeights(). Method in class java.awt. GridBagLayout
getLeading(). Method in class java.awt.FontMetrics
Gets the standard leading, or line spacing, for the font.
getLength(). Method in class java.net.DatagramPacket
getLineIncrement(). Method in class java.awt.Scrollbar
Gets the line increment for this scrollbar.
getLineNumber(). Method in class java.io.LineNumberInputStream
Returns the current line number.
getLineNumber(). Method in class sun.tools.debug.RemoteStackFrame
Return the source file line number.
getLocalHost(). Static method in class java.net.InetAddress
Returns the local host.
getLocalizedInputStream(InputStream). Method in class java.lang.Runtime
Localize an input stream.
getLocalizedOutputStream(OutputStream). Method in class java.lang.Runtime
Localize an output stream.
getLocalPort(). Method in class java.net.DatagramSocket
Returns the local port that this socket is bound to.
getLocalPort(). Method in class java.net.ServerSocket
Gets the port on which the socket is listening.
getLocalPort(). Method in class java.net.Socket
Gets the local port to which the socket is connected.
getLocalPort(). Method in class java.net.SocketImpl
getLocalVariable(String). Method in class sun.tools.debug.RemoteStackFrame
Return a specific (named) stack variable.
getLocalVariables(). Method in class sun.tools.debug.RemoteStackFrame
Return an array of all valid local variables and method arguments for this stack frame.
getLong(String). Static method in class java.lang.Long
Get a Long property.
getLong(String, Long). Static method in class java.lang.Long
Get a Long property.
getLong(String, long). Static method in class java.lang.Long
Get a Long property.
getMapSize(). Method in class java.awt.image.IndexColorModel
Returns the size of the color component arrays in this IndexColorModel.
getMaxAdvance(). Method in class java.awt.FontMetrics
Gets the maximum advance width of any character in this Font.
getMaxAscent(). Method in class java.awt.FontMetrics
Gets the maximum ascent of all characters in this Font.
getMaxDecent(). Method in class java.awt.FontMetrics
For backward compatibility only.
getMaxDescent(). Method in class java.awt.FontMetrics
Gets the maximum descent of all characters.
getMaximum(). Method in class java.awt.Scrollbar
Returns the maximum value of this Scrollbar.
getMaxPriority). Method in class java.lang.ThreadGroup
Gets the maximum priority of the group.
getMenu(int). Method in class java.awt.MenuBar
Gets the specified menu.
getMenuBar(). Method in class java.awt.Frame
Gets the menu bar for this Frame.
getMessage(). Method in class java.lang.Throwable
Gets the detail message of the Throwable.

getMethod(String). Method in class sun.tools.debug.RemoteClass
Return the method, specified by name.
getMethodName(). Method in class sun.tools.debug.RemoteStackFrame
Get the method name referenced by this stackframe.
getMethodNames(). Method in class sun.tools.debug.RemoteClass
Return the names of all methods supported by this class.
getMethods(). Method in class sun.tools.debug.RemoteClass
Return the class's methods.
getMinimum(). Method in class java.awt.Scrollbar
Returns the minimum value of this Scrollbar.
GetMinSize(Container, GridBagLayoutInfo). Method in class java.awt. GridBagLayout
getMinutes(). Method in class java.util.Date
Returns the minute.
getMode(). Method in class java.awt.FileDialog
Gets the mode of the file dialog.
getModifiers(). Method in class sun.tools.debug.RemoteField
Returns a string with the field's modifiers, such as "public", "static", "final", etc.
getMonth(). Method in class java.util.Date
Returns the month.
getName(). Method in class java.lang.Class
Returns the name of this Class.
getName(). Method in class java.io.File
Gets the name of the file.
getName(). Method in class java.awt.Font
Gets the logical name of the font.
getName(). Method in class sun.tools.debug.RemoteClass
Returns the name of the class.
getName(). Method in class sun.tools.debug.RemoteField
Returns the name of the field.
getName(). Method in class sun.tools.debug.RemoteStackVariable
Return the name of a stack variable or argument.
getName(). Method in class sun.tools.debug.RemoteThread
Return the name of the thread.
getName(). Method in class sun.tools.debug.RemoteThreadGroup
Return the threadgroup's name.
getName(). Method in class java.lang.Thread
Gets and returns this Thread's name.
getName(). Method in class java.lang.ThreadGroup
Gets the name of this Thread group.
getOrientation(). Method in class java.awt.Scrollbar
Returns the orientation for this Scrollbar.
getOutputStream(). Method in class java.lang.Process
Returns a Stream connected to the input of the child process.
getOutputStream(). Method in class java.net.Socket
Gets an OutputStream for this socket.
getOutputStream(). Method in class java.net.SocketImpl
Gets an OutputStream for this socket.
getOutputStream(). Method in class java.net.URLConnection
Calls this routine to get an OutputStream that writes to the object.
getPageIncrement(). Method in class java.awt.Scrollbar
Gets the page increment for this scrollbar.
getParameter(String). Method in class java.applet.Applet
Gets a parameter of the applet.
getParameter(String). Method in interface java.applet.AppletStub
Gets a parameter of the applet.
getParameterInfo(). Method in class java.applet.Applet

Returns an array of strings describing the parameters that are understood by this applet.
getParent(). Method in class java.awt.Component
Gets the parent of the component.
getParent(). Method in class java.io.File
Gets the name of the parent directory.
getParent(). Method in class java.awt.MenuComponent
Returns the parent container.
getParent(). Method in class java.lang.ThreadGroup
Gets the parent of this Thread group.
getPath(). Method in class java.io.File
Gets the path of the file.
getPC(). Method in class sun.tools.debug.RemoteStackFrame
Get the program counter referenced by this stackframe.
getPeer(). Method in class java.awt.Component
Gets the peer of the component.
getPeer(). Method in class java.awt.MenuComponent
Gets the MenuComponent's peer.
getPixelSize(). Method in class java.awt.image.ColorModel
Returns the number of bits per pixel described by this ColorModel.
getPort(). Method in class java.net.DatagramPacket
getPort(). Method in class java.net.Socket
Gets the remote port to which the socket is connected.
getPort(). Method in class java.net.SocketImpl
getPort(). Method in class java.net.URL
Gets the port number.
getPriority(). Method in class java.lang.Thread
Gets and returns the Thread's priority.
getProperties(). Static method in class java.lang.System
Gets the System properties.
getProperty(String). Method in class java.util.Properties
Gets a property with the specified key.
getProperty(String). Static method in class java.lang.System
Gets the System property indicated by the specified key.
getProperty(String, ImageObserver). Method in class java.awt.Image
Gets a property of the image by name.
getProperty(String, String). Method in class java.util.Properties
Gets a property with the specified key and default.
getProperty(String, String). Static method in class java.lang.System
Gets the System property indicated by the specified key and def.
getProtocol(). Method in class java.net.URL
Gets the protocol name.
getRed(). Method in class java.awt.Color
Gets the red component.
getRed(int). Method in class java.awt.image.ColorModel
The subclass must provide a function which provides the red color compoment for the specified pixel.
getRed(int). Method in class java.awt.image.DirectColorModel
Returns the red color compoment for the specified pixel in the range 0-255.
getRed(int). Method in class java.awt.image.IndexColorModel
Returns the red color compoment for the specified pixel in the range 0-255.
getRedMask(). Method in class java.awt.image.DirectColorModel
Returns the mask indicating which bits in a pixel contain the red color component.
getReds(byte[]). Method in class java.awt.image.IndexColorModel
Copies the array of red color components into the given array.
getRef(). Method in class java.net.URL
Gets the ref.
getRemoteClass(). Method in class sun.tools.debug.RemoteStackFrame

Get the class this stackframe references.
getRequestProperty(String). Method in class java.net.URLConnection
getRGB(). Method in class java.awt.Color
Gets the RGB value representing the color in the default RGB ColorModel.
getRGB(int). Method in class java.awt.image.ColorModel
Returns the color of the pixel in the default RGB color model.
getRGB(int). Method in class java.awt.image.DirectColorModel
Returns the color of the pixel in the default RGB color model.
getRGB(int). Method in class java.awt.image.IndexColorModel
Returns the color of the pixel in the default RGB color model.
getRGBdefault(). Static method in class java.awt.image.ColorModel
Return a ColorModel which describes the default format for integer RGB values used throughout the AWT image interfaces.
getRows(). Method in class java.awt.List
Returns the number of visible lines in this list.
getRows(). Method in class java.awt.TextArea
Returns the number of rows in the TextArea.
getRuntime(). Static method in class java.lang.Runtime
Returns the runtime.
getScreenResolution(). Method in class java.awt.Toolkit
Returns the screen resolution in dots-per-inch.
getScreenSize(). Method in class java.awt.Toolkit
Gets the size of the screen.
getSeconds(). Method in class java.util.Date
Returns the second.
getSecurityContext(). Method in class java.lang.SecurityManager
Returns an implementation-dependent Object which encapsulates enough information about the current execution environment to perform some of the security checks later.
getSecurityManager(). Static method in class java.lang.System
Gets the system security interface.
getSelectedIndex(). Method in class java.awt.Choice
Returns the index of the currently selected item.
getSelectedIndex(). Method in class java.awt.List
Get the selected item on the list or -1 if no item is selected.
getSelectedIndexes(). Method in class java.awt.List
Returns the selected indexes on the list.
getSelectedIndexes(). Method in interface java.awt.peer.ListPeer
getSelectedItem(). Method in class java.awt.Choice
Returns a String representation of the current choice.
getSelectedItem(). Method in class java.awt.List
Returns the selected item on the list or null if no item is selected.
getSelectedItems(). Method in class java.awt.List
Returns the selected items on the list.
getSelectedText(). Method in class java.awt.TextComponent
Returns the selected text contained in this TextComponent.
getSelectionEnd(). Method in class java.awt.TextComponent
Returns the selected text's end position.
getSelectionEnd(). Method in interface java.awt.peer.TextComponentPeer
getSelectionStart(). Method in class java.awt. TextComponent
Returns the selected text's start position.
getSelectionStart(). Method in interface java.awt.peer.TextComponentPeer
getSize(). Method in class java.awt.Font
Gets the point size of the font.
getSize(). Method in class sun.tools.debug.RemoteArray
Return the number of elements in the array.
getSource(). Method in class java.awt.Image

Gets the object that produces the pixels for the image.
getSourceFile(). Method in class sun.tools.debug.RemoteClass
Get the source file referenced by this stackframe.
getSourceFileName(). Method in class sun.tools.debug.RemoteClass
Get the name of the source file referenced by this stackframe.
getSourcePath(). Method in class sun.tools.debug.RemoteDebugger
Return the source file path the Agent is currently using.
getStackVariable(String). Method in class sun.tools.debug.RemoteThread
Return a stack variable from the current stackframe.
getStackVariables(). Method in class sun.tools.debug.RemoteThread
Return the arguments and local variable from the current stackframe.
getState(). Method in class java.awt.Checkbox
Returns the boolean state of the Checkbox.
getState(). Method in class java.awt.CheckboxMenuItem
Returns the state of this MenuItem.
getStaticFields(). Method in class sun.tools.debug.RemoteClass
Return all the static fields for this class.
getStatus(). Method in class sun.tools.debug.RemoteThread
Return the thread status description
getStyle(). Method in class java.awt.Font
Gets the style of the font.
getSuperclass(). Method in class java.lang.Class
Returns the superclass of this Class.
getSuperclass(). Method in class sun.tools.debug.RemoteClass
Return the superclass for this class.
getText(). Method in class java.awt.Label
Gets the text of this label.
getText(). Method in class java.awt.TextComponent
Returns the text contained in this TextComponent.
getText(). Method in interface java.awt.peer.TextComponentPeer
getThreadGroup(). Method in class java.lang.Thread
Gets and returns this Thread group.
getTime(). Method in class java.util.Date
Returns the time in milliseconds since the epoch.
getTimezoneOffset(). Method in class java.util.Date
Return the time zone offset in minutes for the current locale that is appropriate for this time.
getTitle(). Method in class java.awt.Dialog
Gets the title of the Dialog.
getTitle(). Method in class java.awt.Frame
Gets the title of the Frame.
getToolkit(). Method in class java.awt.Component
Gets the toolkit of the component.
getToolkit(). Method in interface java.awt.peer.ComponentPeer
getToolkit(). Method in class java.awt.Window
Returns the toolkit of this frame.
getTransparentPixel(). Method in class java.awt.image.IndexColorModel
Returns the index of the transparent pixel in this IndexColorModel or -1 if there is no transparent pixel.
getType(). Method in class sun.tools.debug.RemoteField
Returns a type string describing the field.
getType(). Method in class sun.tools.debug.RemoteValue
Returns the RemoteValue's type.
getURL(). Method in class java.net.URLConnection
Gets the URL for this connection.
getUseCaches(). Method in class java.net.URLConnection
getValue(). Method in class sun.tools.debug. RemoteStackVariable

Return the value of a stack variable or argument.
getValue(). Method in class java.awt.Scrollbar
Returns the current value of this Scrollbar.
getVisible(). Method in class java.awt.Scrollbar
Returns the visible amount of the Scrollbar.
getVisibleIndex(). Method in class java.awt.List
Gets the index of the item that was last made visible by the method makeVisible.
getWarningString(). Method in class java.awt.Window
Gets the warning string for this window.
getWidth(ImageObserver). Method in class java.awt.Image
Gets the actual width of the image.
getWidths(). Method in class java.awt.FontMetrics
Gets the widths of the first 256 characters in the Font.
getYear(). Method in class java.util.Date
Returns the year after 1900.
GOT_FOCUS. Static variable in class java.awt.Event
A component gained the focus.
gotFocus(Event, Object). Method in class java.awt.Component
Indicates that this component has received the input focus.
grabPixels(). Method in class java.awt.image.PixelGrabber
Request the Image or ImageProducer to start delivering pixels and wait for all of the pixels in the rectangle of interest to be delivered.
grabPixels(long). Method in class java.awt.image.PixelGrabber
Request the Image or ImageProducer to start delivering pixels and wait for all of the pixels in the rectangle of interest to be delivered or until the specified timeout has elapsed.
Graphics(). Constructor for class java.awt.Graphics
Constructs a new Graphics Object.
gray. Static variable in class java.awt.Color
The color gray.
green. Static variable in class java.awt.Color
The color green.
GridBagConstraints(). Constructor for class java.awt.GridBagConstraints
GridBagLayout(). Constructor for class java.awt. GridBagLayout
Creates a gridbag layout.
gridheight. Variable in class java.awt.GridBagConstraints
GridLayout(int, int). Constructor for class java.awt. GridLayout
Creates a grid layout with the specified rows and columns.
GridLayout(int, int, int, int). Constructor for class java.awt.GridLayout
Creates a grid layout with the specified rows, columns, horizontal gap, and vertical gap.
gridwidth. Variable in class java.awt.GridBagConstraints
gridx. Variable in class java.awt.GridBagConstraints
gridy. Variable in class java.awt.GridBagConstraints
grow(int, int). Method in class java.awt.Rectangle
Grows the rectangle horizontally and vertically.
guessContentTypeFromName(String). Static method in class java.net. URLConnection
A useful utility routine that tries to guess the content-type of an object based upon its extension.
guessContentTypeFromStream(InputStream). Static method in class java.net. URLConnection
This method is used to check for files that have some type that can be determined by inspection.

H

HAND_CURSOR.Static variable in class java.awt.Frame
handleEvent(Event). Method in class java.awt.Component

Handles the event.
handleEvent(Event). Method in interface java.awt.peer.ComponentPeer
hasChanged(). Method in class java.util.Observable
Returns a true boolean if an observable change has occurred.
hashCode(). Method in class java.util.BitSet
Gets the hashcode.
hashCode(). Method in class java.lang.Boolean
Returns a hashcode for this Boolean.
hashCode(). Method in class java.lang.Character
Returns a hashcode for this Character.
hashCode(). Method in class java.awt.Color
Computes the hash code.
hashCode(). Method in class java.util.Date
Computes a hashCode.
hashCode(). Method in class java.lang.Double
Returns a hashcode for this Double.
hashCode(). Method in class java.io.File
Computes a hashcode for the file.
hashCode(). Method in class java.lang.Float
Returns a hashcode for this Float.
hashCode(). Method in class java.awt.Font
Returns a hashcode for this font.
hashCode(). Method in class java.net.InetAddress
Returns a hashcode for this InetAddress.
hashCode(). Method in class java.lang.Integer
Returns a hashcode for this Integer.
hashCode(). Method in class java.lang.Long
Computes a hashcode for this Long.
hashCode(). Method in class java.lang.Object
Returns a hashcode for this Object.
hashCode(). Method in class java.awt.Point
Returns the hashcode for this Point.
hashCode(). Method in class java.awt.Rectangle
Returns the hashcode for this Rectangle.
hashCode(). Method in class java.lang.String
Returns a hashcode for this String.
hashCode(). Method in class java.net.URL
Creates an integer suitable for hash table indexing.
Hashtable(). Constructor for class java.util.Hashtable
Constructs a new, empty hashtable.
Hashtable(int). Constructor for class java.util.Hashtable
Constructs a new, empty hashtable with the specified initial capacity.
Hashtable(int, float). Constructor for class java.util.Hashtable
Constructs a new, empty hashtable with the specified initial capacity and the specified load factor.
hasMoreElements(). Method in interface java.util.Enumeration
Returns true if the enumeration contains more elements; false if its empty.
hasMoreElements(). Method in class java.util.StringTokenizer
Returns true if the Enumeration has more elements.
hasMoreTokens(). Method in class java.util.StringTokenizer
Returns true if more tokens exist.
height. Variable in class java.awt.Dimension
The height dimension.
HEIGHT. Static variable in interface java.awt.image.ImageObserver
The height of the base image is now available and can be taken from the height argument to the imageUpdate callback method.
height. Variable in class java.awt.Rectangle

The height of the rectangle.
hide(). Method in class java.awt.Component
Hides the component.
hide(). Method in interface java.awt.peer.ComponentPeer
HOME. Static variable in class java.awt.Event
The home key.
HORIZONTAL. Static variable in class java.awt.GridBagConstraints
HORIZONTAL. Static variable in class java.awt.Scrollbar
The horizontal Scrollbar variable.
HSBtoRGB(float, float, float). Static method in class java.awt.Color
Returns the RGB value defined by the default RGB ColorModel, of the color corresponding to the given HSB color components.

I

id.Variable in class java.awt.Event
The type of this event.
IEEEremainder(double, double). Static method in class java.lang.Math
Returns the remainder of f1 divided by f2 as defined by IEEE 754.
ifModifiedSince. Variable in class java.net.URLConnection
ignoreExceptions(). Method in class sun.tools.debug. RemoteClass
Don't enter the debugger when an instance of this class is thrown.
IllegalAccessError(). Constructor for class java.lang.IllegalAccessError
Constructs an IllegalAccessError with no detail message.
IllegalAccessError(String). Constructor for class java.lang.IllegalAccessError
Constructs an IllegalAccessError with the specified detail message.
IllegalAccessException(). Constructor for class java.lang. IllegalAccessException
Constructs a IllegalAccessException without a detail message.
IllegalAccessException(String). Constructor for class java.lang. IllegalAccessException
Constructs a IllegalAccessException with a detail message.
IllegalArgumentException(). Constructor for class java.lang. IllegalArgumentException
Constructs an IllegalArgumentException with no detail message.
IllegalArgumentException(String). Constructor for class java.lang. IllegalArgumentException
Constructs an IllegalArgumentException with the specified detail message.
IllegalMonitorStateException(). Constructor for class java.lang. IllegalMonitorStateException
Constructs an IllegalMonitorStateException with no detail message.
IllegalMonitorStateException(String). Constructor for class java.lang. IllegalMonitorStateException
Constructs an IllegalMonitorStateException with the specified detail message.
IllegalThreadStateException(). Constructor for class java.lang. IllegalThreadStateException
Constructs an IllegalThreadStateException with no detail message.
IllegalThreadStateException(String). Constructor for class java.lang. IllegalThreadStateException
Constructs an IllegalThreadStateException with the specified detail message.
Image(). Constructor for class java.awt.Image
IMAGEABORTED. Static variable in interface java.awt.image.ImageConsumer
The image creation process was deliberately aborted.
imageComplete(int). Method in interface java.awt.image.ImageConsumer
The imageComplete method is called when the ImageProducer is finished delivering all of the pixels that the source image contains, or when a single frame of a multi-frame animation has been completed, or when an error in loading or producing the image has occured.
imageComplete(int). Method in class java.awt.image.ImageFilter
Filters the information provided in the imageComplete method of the ImageConsumer interface.
imageComplete(int). Method in class java.awt.image.PixelGrabber
The imageComplete method is part of the ImageConsumer API which this class must implement to

retrieve the pixels.
IMAGEERROR. Static variable in interface java.awt.image.ImageConsumer
An error was encountered while producing the image.
ImageFilter(). Constructor for class java.awt.image.ImageFilter
imageUpdate(Image, int, int, int, int, int). Method in class java.awt.Component
Repaints the component when the image has changed.
imageUpdate(Image, int, int, int, int, int). Method in interface java.awt.image. ImageObserver
This method is called when information about an image which was previously requested using an asynchronous interface becomes available.
in. Static variable in class java.io.FileDescriptor
Handle to standard input.
in. Variable in class java.io.FilterInputStream
The actual input stream.
in. Static variable in class java.lang.System
Standard input stream.
inCheck. Variable in class java.lang.SecurityManager
inClass(String). Method in class java.lang. SecurityManager
Returns true if the specified String is in this Class.
inClassLoader(). Method in class java.lang.SecurityManager
Returns a boolean indicating whether or not the current ClassLoader is equal to null.
IncompatibleClassChangeError(). Constructor for class java.lang. IncompatibleClassChangeError
Constructs an IncompatibleClassChangeError with no detail message.
IncompatibleClassChangeError(String). Constructor for class java.lang. IncompatibleClassChangeError
Constructs an IncompatibleClassChangeError with the specified detail message.
IndexColorModel(int, int, byte[], byte[], byte[]). Constructor for class java.awt.image.IndexColorModel
Constructs an IndexColorModel from the given arrays of red, green, and blue components.
IndexColorModel(int, int, byte[], byte[], byte[], byte[]). Constructor for class java.awt.image.IndexColorModel
Constructs an IndexColorModel from the given arrays of red, green, blue and alpha components.
IndexColorModel(int, int, byte[], byte[], byte[], int). Constructor for class java.awt.image.IndexColorModel
Constructs an IndexColorModel from the given arrays of red, green, and blue components.
IndexColorModel(int, int, byte[], int, boolean). Constructor for class java.awt.image.IndexColorModel
Constructs an IndexColorModel from a single arrays of packed red, green, blue and optional alpha components.
IndexColorModel(int, int, byte[], int, boolean, int). Constructor for class java.awt.image.IndexColorModel
Constructs an IndexColorModel from a single arrays of packed red, green, blue and optional alpha components.
indexOf(int). Method in class java.lang.String
Returns the index within this String of the first occurrence of the specified character.
indexOf(int, int). Method in class java.lang.String
Returns the index within this String of the first occurrence of the specified character, starting the search at fromIndex.
indexOf(Object). Method in class java.util.Vector
Searches for the specified object, starting from the first position and returns an index to it.
indexOf(Object, int). Method in class java.util.Vector
Searches for the specified object, starting at the specified position and returns an index to it.
indexOf(String). Method in class java.lang.String
Returns the index within this String of the first occurrence of the specified substring.
indexOf(String, int). Method in class java.lang.String
Returns the index within this String of the first occurrence of the specified substring.
IndexOutOfBoundsException(). Constructor for class java.lang. IndexOutOfBoundsException
Constructs an IndexOutOfBoundsException with no detail message.
IndexOutOfBoundsException(String). Constructor for class java.lang. IndexOutOfBoundsException
Constructs a IndexOutOfBoundsException with the specified detail message.
init(). Method in class java.applet.Applet
Initializes the applet.

InternalError(String). Constructor for class java.lang.InternalError
Constructs an InternalError with the specified detail message.
interrupt(). Method in class java.lang.Thread
Send an interrupt to a thread.
interrupted(). Static method in class java.lang.Thread
Ask if you have been interrupted.
InterruptedException(). Constructor for class java.lang.InterruptedException
Constructs an InterruptedException with no detail message.
InterruptedException(String). Constructor for class java.lang. InterruptedException
Constructs an InterruptedException with the specified detail message.
InterruptedIOException(). Constructor for class java.io.InterruptedIOException
Constructs an IOException with no detail message.
InterruptedIOException(String). Constructor for class java.io. InterruptedIOException
Constructs an IOException with the specified detail message.
intersection(Rectangle). Method in class java.awt.Rectangle
Computes the intersection of two rectangles.
intersects(Rectangle). Method in class java.awt.Rectangle
Checks if two rectangles intersect.
intValue(). Method in class java.lang.Double
Returns the integer value of this Double (by casting to an int).
intValue(). Method in class java.lang.Float
Returns the integer value of this Float (by casting to an int).
intValue(). Method in class java.lang.Integer
Returns the value of this Integer as an int.
intValue(). Method in class java.lang.Long
Returns the value of this Long as an int.
intValue(). Method in class java.lang.Number
Returns the value of the number as an int.
invalidate(). Method in class java.awt.Component
Invalidates a component.
IOException(). Constructor for class java.io.IOException
Constructs an IOException with no detail message.
IOException(String). Constructor for class java.io.IOException
Constructs an IOException with the specified detail message.
ipadx. Variable in class java.awt.GridBagConstraints
ipady. Variable in class java.awt.GridBagConstraints
isAbsolute(). Method in class java.io.File
Returns a boolean indicating whether the file name is absolute.
isActive(). Method in class java.applet.Applet
Returns true if the applet is active.
isActive(). Method in interface java.applet.AppletStub
Returns true if the applet is active.
isAlive(). Method in class java.lang.Thread
Returns a boolean indicating if the Thread is active.
isBold(). Method in class java.awt.Font
Returns true if the font is bold.
isConsumer(ImageConsumer). Method in class java.awt.image.FilteredImageSource
Determines whether an ImageConsumer is on the list of consumers currently interested in data for this image.
isConsumer(ImageConsumer). Method in interface java.awt.image.ImageProducer
This method determines if a given ImageConsumer object is currently registered with this ImageProducer as one of its consumers.
isConsumer(ImageConsumer). Method in class java.awt.image.MemoryImageSource
Determine if an ImageConsumer is on the list of consumers currently interested in data for this image.
isDaemon(). Method in class java.lang.Thread

Returns the daemon flag of the Thread.
isDaemon(). Method in class java.lang.ThreadGroup
Returns the daemon flag of the Thread group.
isDigit(char). Static method in class java.lang.Character
Determines if the specified character is a ISO-LATIN-1 digit.
isDirectory(). Method in class java.io.File
Returns a boolean indicating whether or not a directory file exists.
isEditable(). Method in class java.awt.TextComponent
Returns the boolean indicating whether this TextComponent is editable or not.
isEmpty(). Method in class java.util.Dictionary
Returns true if the Dictionary contains no elements.
isEmpty(). Method in class java.util.Hashtable
Returns true if the hashtable contains no elements.
isEmpty(). Method in class java.awt.Rectangle
Determines whether the rectangle is empty.
isEmpty(). Method in class java.util.Vector
Returns true if the collection contains no values.
isEnabled(). Method in class java.awt.Component
Checks if this Component is enabled.
isEnabled(). Method in class java.awt.MenuItem
Checks whether the menu item is enabled.
isErrorAny(). Method in class java.awt.MediaTracker
Checks the error status of all of the images.
isErrorID(int). Method in class java.awt.MediaTracker
Checks the error status of all of the images with the specified ID.
isFile(). Method in class java.io.File
Returns a boolean indicating whether or not a normal file exists.
isInfinite(). Method in class java.lang.Double
Returns true if this Double value is infinitely large in magnitude.
isInfinite(). Method in class java.lang.Float
Returns true if this Float value is infinitely large in magnitude.
isInfinite(double). Static method in class java.lang.Double
Returns true if the specified number is infinitely large in magnitude.
isInfinite(float). Static method in class java.lang.Float
Returns true if the specified number is infinitely large in magnitude.
isInterface(). Method in class java.lang.Class
Returns a boolean indicating whether or not this Class is an interface.
isInterface(). Method in class sun.tools.debug.RemoteClass
Is this RemoteClass an interface?
isInterrupted(). Method in class java.lang.Thread
Ask if another thread has been interrupted.
isItalic(). Method in class java.awt.Font
Returns true if the font is italic.
isLowerCase(char). Static method in class java.lang.Character
Determines if the specified character is ISO-LATIN-1 lower case.
isModal(). Method in class java.awt.Dialog
Returns true if the Dialog is modal.
isNaN(). Method in class java.lang.Double
Returns true if this Double value is the special Not-a-Number (NaN) value.
isNaN(). Method in class java.lang.Float
Returns true if this Float value is Not-a-Number (NaN).
isNaN(double). Static method in class java.lang.Double
Returns true if the specified number is the special Not-a-Number (NaN) value.
isNaN(float). Static method in class java.lang.Float
Returns true if the specified number is the special Not-a-Number (NaN) value.
isObject(). Method in class sun.tools.debug.RemoteValue

Returns whether the RemoteValue is an Object (as opposed to a primitive type, such as int).
isPlain(). Method in class java.awt.Font
Returns true if the font is plain.
isResizable(). Method in class java.awt.Dialog
Returns true if the user can resize the frame.
isResizable(). Method in class java.awt.Frame
Returns true if the user can resize the Frame.
isSelected(int). Method in class java.awt.List
Returns true if the item at the specified index has been selected; false otherwise.
isShowing(). Method in class java.awt.Component
Checks if this Component is showing on screen.
isSpace(char). Static method in class java.lang.Character
Determines if the specified character is ISO-LATIN-1 white space according to Java.
isStatic(). Method in class sun.tools.debug.RemoteField
Returns whether the field is static (a class variable or method).
isSuspended(). Method in class sun.tools.debug.RemoteThread
Return whether this thread is suspended.
isTearOff(). Method in class java.awt.Menu
Returns true if this is a tear-off menu.
isUpperCase(char). Static method in class java.lang.Character
Determines if the specified character is ISO-LATIN-1 upper case.
isValid(). Method in class java.awt.Component
Checks if this Component is valid.
isVisible(). Method in class java.awt.Component
Checks if this Component is visible.
ITALIC. Static variable in class java.awt.Font
The italicized style constant.
itrace(boolean). Method in clas sun.tools.debug.RemoteDebugger
Turn on/off instruction tracing.

J

join().Method in class java.lang.Thread
Waits forever for this Thread to die.
join(long). Method in class java.lang.Thread
Waits for this Thread to die.
join(long, int). Method in class java.lang.Thread
Waits for the Thread to die, with more precise time.

K

key.Variable in class java.awt.Event
The key that was pressed in a keyboard event.
KEY_ACTION. Static variable in class java.awt.Event
The key action keyboard event.
KEY_ACTION_RELEASE. Static variable in class java.awt.Event
The key action keyboard event.
KEY_PRESS. Static variable in class java.awt.Event
The key press keyboard event.
KEY_RELEASE. Static variable in class java.awt.Event

The key release keyboard event.
keyDown(Event, int). Method in class java.awt.Component
Called if a character is pressed.
keys(). Method in class java.util.Dictionary
Returns an enumeration of the Dictionary's keys.
keys(). Method in class java.util.Hashtable
Returns an enumeration of the hashtable's keys.
keyUp(Event, int). Method in class java.awt.Component
Called if a character is released.

L

Label().Constructor for class java.awt.Label
Constructs an empty label.
Label(String). Constructor for class java.awt.Label
Constructs a new label with the specified String of text.
Label(String, int). Constructor for class java.awt.Label
Constructs a new label with the specified String of text and the specified alignment.
last(Container). Method in class java.awt.CardLayout
Flips to the last card of the specified container.
lastElement(). Method in class java.util.Vector
Returns the last element of the sequence.
lastIndexOf(int). Method in class java.lang.String
Returns the index within this String of the last occurrence of the specified character.
lastIndexOf(int, int). Method in class java.lang.String
Returns the index within this String of the last occurrence of the specified character.
lastIndexOf(Object). Method in class java.util.Vector
Searches backwards for the specified object, starting from the last position and returns an index to it.
lastIndexOf(Object, int). Method in class java.util.Vector
Searches backwards for the specified object, starting from the specified position and returns an index to it.
lastIndexOf(String). Method in class java.lang.String
Returns the index within this String of the last occurrence of the specified substring.
lastIndexOf(String, int). Method in class java.lang.String
Returns the index within this String of the last occurrence of the specified substring.
lastModified(). Method in class java.io.File
Returns the last modification time.
layout(). Method in class java.awt.Component
Lays out the component.
layout(). Method in class java.awt.Container
Does a layout on this Container.
layoutContainer(Container). Method in class java.awt.BorderLayout
Lays out the specified container.
layoutContainer(Container). Method in class java.awt.CardLayout
Performs a layout in the specified panel.
layoutContainer(Container). Method in class java.awt.FlowLayout
Lays out the container.
layoutContainer(Container). Method in class java.awt.GridBagLayout
Lays out the container in the specified panel.
layoutContainer(Container). Method in class java.awt.GridLayout
Lays out the container in the specified panel.
layoutContainer(Container). Method in interface java.awt.LayoutManager
Lays out the container in the specified panel.

listThreadGroups(RemoteThreadGroup). Method in class sun.tools.debug. RemoteDebugger
List threadgroups
listThreads(boolean). Method in class sun.tools.debug.RemoteThreadGroup
List a threadgroup's threads
LOAD. Static variable in class java.awt.FileDialog
The file load variable.
load(InputStream). Method in class java.util.Properties
Loads properties from an InputStream.
load(String). Method in class java.lang.Runtime
Loads a dynamic library, given a complete path name.
load(String). Static method in class java.lang.System
Loads a dynamic library, given a complete path name.
LOAD_FILE. Static variable in class java.awt.Event
A file loading event.
loadClass(String, boolean). Method in class java.lang.ClassLoader
Resolves the specified name to a Class.
LOADING. Static variable in class java.awt.MediaTracker
Flag indicating some media is currently being loaded.
loadLibrary(String). Method in class java.lang.Runtime
Loads a dynamic library with the specified library name.
loadLibrary(String). Static method in class java.lang.System
Loads a dynamic library with the specified library name.
localport. Variable in class java.net.SocketImpl
locate(int, int). Method in class java.awt.Component
Returns the component or subcomponent that contains the x,y location.
locate(int, int). Method in class java.awt.Container
Locates the component that contains the x,y position.
location(). Method in class java.awt.Component
Returns the current location of this component.
location(int, int). Method in class java.awt.GridBagLayout
log(double). Static method in class java.lang.Math
Returns the natural logarithm (base e) of a.
Long(long). Constructor for class java.lang.Long
Constructs a Long object initialized to the specified value.
Long(String). Constructor for class java.lang.Long
Constructs a Long object initialized to the value specified by the String parameter.
longBitsToDouble(long). Static method in class java.lang.Double
Returns the double-float corresponding to a given bit represention.
longValue(). Method in class java.lang.Double
Returns the long value of this Double (by casting to a long).
longValue(). Method in class java.lang.Float
Returns the long value of this Float (by casting to a long).
longValue(). Method in class java.lang.Integer
Returns the value of this Integer as a long.
longValue(). Method in class java.lang.Long
Returns the value of this Long as a long.
longValue(). Method in class java.lang.Number
Returns the value of the number as a long.
lookupConstraints(Component). Method in class java.awt.GridBagLayout
Retrieves the constraints for the specified component.
loop(). Method in interface java.applet.AudioClip
Starts playing the clip in a loop.
LOST_FOCUS. Static variable in class java.awt.Event
A component lost the focus.
lostFocus(Event, Object). Method in class java.awt.Component
Indicates that this component has lost the input focus.

lowerCaseMode(boolean). Method in class java.io.StreamTokenizer
Examines a boolean to decide whether TT_WORD tokens are forced to be lower case.

M

magenta.Static variable in class java.awt.Color
The color magneta.
makeVisible(int). Method in class java.awt.List
Forces the item at the specified index to be visible.
makeVisible(int). Method in interface java.awt.peer.ListPeer
MalformedURLException(). Constructor for class java.net.MalformedURLException
Constructs a MalformedURLException with no detail message.
MalformedURLException(String). Constructor for class java.net. MalformedURLException
Constructs a MalformedURLException with the specified detail message.
mark(int). Method in class java.io.BufferedInputStream
Marks the current position in the input stream.
mark(int). Method in class java.io.FilterInputStream
Marks the current position in the input stream.
mark(int). Method in class java.io.InputStream
Marks the current position in the input stream.
mark(int). Method in class java.io.LineNumberInputStream
Marks the current position in the input stream.
marklimit. Variable in class java.io.BufferedInputStream
The maximum readahead allowed after a mark() before subsequent calls to reset() fail.
markpos. Variable in class java.io.BufferedInputStream
The position in the buffer of the current mark.
markSupported(). Method in class java.io.BufferedInputStream
Returns a boolean indicating if this stream type supports mark/reset.
markSupported(). Method in class java.io.FilterInputStream
Returns true if this stream type supports mark/reset.
markSupported(). Method in class java.io.InputStream
Returns a boolean indicating whether or not this stream type supports mark/reset.
markSupported(). Method in class java.io.PushbackInputStream
Returns true if this stream type supports mark/reset.
max(double, double). Static method in class java.lang.Math
Takes two double values, a and b, and returns the greater number of the two.
max(float, float). Static method in class java.lang.Math
Takes two float values, a and b, and returns the greater number of the two.
max(int, int). Static method in class java.lang.Math
Takes two int values, a and b, and returns the greater number of the two.
max(long, long). Static method in class java.lang.Math
Takes two long values, a and b, and returns the greater number of the two.
MAX_PRIORITY. Static variable in class java.lang.Thread
The maximum priority that a Thread can have.
MAX_RADIX. Static variable in class java.lang.Character
The maximum radix available for conversion to and from Strings.
MAX_VALUE. Static variable in class java.lang.Boolean
The maximum value a Character can have.
MAX_VALUE. Static variable in class java.lang.Double
The maximum value a double can have.
MAX_VALUE. Static variable in class java.lang.Float
The maximum value a float can have.
MAX_VALUE. Static variable in class java.lang.Integer

The maximum value an Integer can have.
MAX_VALUE. Static variable in class java.lang.Long
The maximum value a Long can have.
MAXGRIDSIZE. Static variable in class java.awt.GridBagLayout
MediaTracker(Component). Constructor for class java.awt.MediaTracker
Creates a Media tracker to track images for a given Component.
MemoryImageSource(int, int, ColorModel, byte[], int, int). Constructor for class java.awt.image.MemoryImageSource
Constructs an ImageProducer object which uses an array of bytes to produce data for an Image object.
MemoryImageSource(int, int, ColorModel, byte[], int, int, Hashtable). Constructor for class java.awt.image.MemoryImageSource
Constructs an ImageProducer object which uses an array of bytes to produce data for an Image object.
MemoryImageSource(int, int, ColorModel, int[], int, int). Constructor for class java.awt.image.MemoryImageSource
Constructs an ImageProducer object which uses an array of integers to produce data for an Image object.
MemoryImageSource(int, int, ColorModel, int[], int, int, Hashtable). Constructor for class java.awt.image.MemoryImageSource
Constructs an ImageProducer object which uses an array of integers to produce data for an Image object.
MemoryImageSource(int, int, int[], int, int). Constructor for class java.awt.image.MemoryImageSource
Constructs an ImageProducer object which uses an array of integers in the default RGB ColorModel to produce data for an Image object.
MemoryImageSource(int, int, int[], int, int, Hashtable). Constructor for class java.awt.image.MemoryImageSource
Constructs an ImageProducer object which uses an array of integers in the default RGB ColorModel to produce data for an Image object.
Menu(String). Constructor for class java.awt.Menu
Constructs a new Menu with the specified label.
Menu(String, boolean). Constructor for class java.awt.Menu
Constructs a new Menu with the specified label.
MenuBar(). Constructor for class java.awt.MenuBar
Creates a new menu bar.
MenuComponent(). Constructor for class java.awt.MenuComponent
MenuItem(String). Constructor for class java.awt. MenuItem
Constructs a new MenuItem with the specified label.
META_MASK. Static variable in class java.awt.Event
The meta modifier constant.
metaDown(). Method in class java.awt.Event
Checks if the meta key is down.
min(double, double). Static method in class java.lang.Math
Takes two double values, a and b, and returns the smallest number of the two.
min(float, float). Static method in class java.lang.Math
Takes two float values, a and b, and returns the smallest number of the two.
min(int, int). Static method in class java.lang.Math
Takes two integer values, a and b, and returns the smallest number of the two.
min(long, long). Static method in class java.lang.Math
Takes two long values, a and b, and returns the smallest number of the two.
MIN_PRIORITY. Static variable in class java.lang.Thread
The minimum priority that a Thread can have.
MIN_RADIX. Static variable in class java.lang.Character
The minimum radix available for conversion to and from Strings.
MIN_VALUE. Static variable in class java.lang.Boolean
The minimum value a Charater can have.

MIN_VALUE. Static variable in class java.lang.Double
The minimum value a double can have.
MIN_VALUE. Static variable in class java.lang.Float
The minimum value a float can have.
MIN_VALUE. Static variable in class java.lang.Integer
The minimum value an Integer can have.
MIN_VALUE. Static variable in class java.lang.Long
The minimum value a Long can have.
minimumLayoutSize(Container). Method in class java.awt.BorderLayout
Returns the minimum dimensions needed to layout the components contained in the specified target container.
minimumLayoutSize(Container). Method in class java.awt.CardLayout
Calculates the minimum size for the specified panel.
minimumLayoutSize(Container). Method in class java.awt.FlowLayout
Returns the minimum dimensions needed to layout the components contained in the specified target container.
minimumLayoutSize(Container). Method in class java.awt.GridBagLayout
Returns the minimum dimensions needed to layout the components contained in the specified panel.
minimumLayoutSize(Container). Method in class java.awt.GridLayout
Returns the minimum dimensions needed to layout the components contained in the specified panel.
minimumLayoutSize(Container). Method in interface java.awt.LayoutManager
Calculates the minimum size dimensions for the specified panel given the components in the specified parent container.
minimumSize(). Method in class java.awt.Component
Returns the minimum size of this component.
minimumSize(). Method in interface java.awt.peer.ComponentPeer
minimumSize(). Method in class java.awt.Container
Returns the minimum size of this container.
minimumSize(). Method in class java.awt.List
Returns the minimum dimensions needed for the list.
minimumSize(). Method in class java.awt.TextArea
Returns the minimum size Dimensions of the TextArea.
minimumSize(). Method in class java.awt.TextField
Returns the minimum size Dimensions needed for this TextField.
minimumSize(int). Method in class java.awt.List
Returns the minimum dimensions needed for the amount of rows in the list.
minimumSize(int). Method in interface java.awt.peer.ListPeer
minimumSize(int). Method in class java.awt.TxtField
Returns the minimum size Dimensions needed for this TextField with the specified amount of columns.
minimumSize(int). Method in interface java.awt.peer.TextFieldPeer
minimumSize(int, int). Method in class java.awt. TextArea
Returns the specified minimum size Dimensions of the TextArea.
minimumSize(int, int). Method in interface java.awt.peer.TextAreaPeer
MINSIZE. Static variable in class java.awt. GridBagLayout
mkdir(). Method in class java.io.File
Creates a directory and returns a boolean indicating the success of the creation.
mkdirs(). Method in class java.io.File
Creates all directories in this path.
modifiers. Variable in class java.awt.Event
The state of the modifier keys.
MOUSE_DOWN. Static variable in class java.awt.Event
The mouse down event.
MOUSE_DRAG. Static variable in class java.awt.Event
The mouse drag event.
MOUSE_ENTER. Static variable in class java.awt.Event

The mouse enter event.
MOUSE_EXIT. Static variable in class java.awt.Event
The mouse exit event.
MOUSE_MOVE. Static variable in class java.awt.Event
The mouse move event.
MOUSE_UP. Static variable in class java.awt.Event
The mouse up event.
mouseDown(Event, int, int). Method in class java.awt.Component
Called if the mouse is down.
mouseDrag(Event, int, int). Method in class java.awt.Component
Called if the mouse is dragged (the mouse button is down).
mouseEnter(Event, int, int). Method in class java.awt.Component
Called when the mouse enters the component.
mouseExit(Event, int, int). Method in class java.awt.Component
Called when the mouse exits the component.
mouseMove(Event, int, int). Method in class java.awt.Component
Called if the mouse moves (the mouse button is up).
mouseUp(Event, int, int). Method in class java.awt.Component
Called if the mouse is up.
move(int, int). Method in class java.awt.Component
Moves the Component to a new location.
move(int, int). Method in class java.awt.Point
Moves the point.
move(int, int). Method in class java.awt.Rectangle
Moves the rectangle.
MOVE_CURSOR. Static variable in class java.awt.Frame

N

N_RESIZE_CURSOR.Static variable in class java.awt.Frame
name. Variable in class java.awt.Font
The logical name of this font.
NaN. Static variable in class java.lang.Double
Not-a-Number.
NaN. Static variable in class java.lang.Float
Not-a-Number.
NE_RESIZE_CURSOR. Static variable in class java.awt.Frame
NEGATIVE_INFINITY. Static variable in class java.lang.Double
Negative infinity.
NEGATIVE_INFINITY. Static variable in class java.lang.Float
Negative infinity.
NegativeArraySizeException(). Constructor for class java.lang. NegativeArraySizeException
Constructs a NegativeArraySizeException with no detail message.
NegativeArraySizeException(String). Constructor for class java.lang. NegativeArraySizeException
Constructs a NegativeArraySizeException with the specified detail message.
newInstance(). Method in class java.lang.Class
Creates a new instance of this Class.
newmodel. Variable in class java.awt.image.RGBImageFilter
next(). Method in class sun.tools.debug.RemoteThread
Continue execution of this thread to the next line, but don't step into a method call.
next(Container). Method in class java.awt.CardLayout
Flips to the next card of the specified container.

nextDouble(). Method in class java.util.Random
Generates a pseudorandom uniformally distributed `double` value between 0.0 and 1.0.
nextElement(). Method in interface java.util.Enumeration
Returns the next element of the enumeration.
nextElement(). Method in class java.util.StringTokenizer
Returns the next element in the Enumeration.
nextFloat(). Method in class java.util.Random
Generates a pseudorandom uniformally distributed `float` value between 0.0 and 1.0.
nextFocus(). Method in class java.awt.Component
Moves the focus to the next component.
nextFocus(). Method in interface java.awt.peer.ComponentPeer
nextGaussian(). Method in class java.util.Random
Generates a pseudorandom Gaussian distributed `double` value with mean 0.0 and standard deviation 1.0.
nextInt(). Method in class java.util.Random
Generates a pseudorandom uniformally distributed `int` value.
nextLong(). Method in class java.util.Random
Generate a pseudorandom uniformally distributed `long` value.
nextToken(). Method in class java.io.StreamTokenizer
Parses a token from the input stream.
nextToken(). Method in class java.util.StringTokenizer
Returns the next token of the String.
nextToken(String). Method in class java.util.StringTokenizer
Returns the next token, after switching to the new delimiter set.
NoClassDefFoundError(). Constructor for class java.lang.NoClassDefFoundError
Constructs a NoClassDefFoundError with no detail message.
NoClassDefFoundError(String). Constructor for class java.lang. NoClassDefFoundError
Constructs a NoClassDefFoundError with the specified detail message.
NONE. Static variable in class java.awt.GridBagConstraints
NORM_PRIORITY. Static variable in class java.lang. Thread
The default priority that is assigned to a Thread.
NORTH. Static variable in class java.awt.GridBagConstraints
NORTHEAST. Static variable in class java.awt. GridBagConstraints
NORTHWEST. Static variable in class java.awt. GridBagConstraints
NoSuchElementException(). Constructor for class java.util.NoSuchElementException
Constructs a NoSuchElementException with no detail message.
NoSuchElementException(String). Constructor for class java.util. NoSuchElementException
Constructs a NoSuchElementException with the specified detail message.
NoSuchFieldError(). Constructor for class java.lang.NoSuchFieldError
Constructs a NoSuchFieldException without a detail message.
NoSuchFieldError(String). Constructor for class java.lang.NoSuchFieldError
Constructs a NoSuchFieldException with a detail message.
NoSuchMethodError(). Constructor for class java.lang.NoSuchMethodError
NoSuchMethodError(String). Constructor for class java.lang.NoSuchMethodError
Constructs a NoSuchMethodException with a detail message.
NoSuchMethodException(). Constructor for class java.lang.NoSuchMethodException
Constructs a NoSuchMethodException without a detail message.
NoSuchMethodException(String). Constructor for class java.lang. NoSuchMethodException
Constructs a NoSuchMethodException with a detail message.
notify(). Method in class java.lang.Object
Notifies a single waiting thread on a change in condition of another thread.
notifyAll(). Method in class java.lang.Object
Notifies all of the threads waiting for a condition to change.
notifyObservers(). Method in class java.util.Observable
Notifies all observers if an observable change occurs.

O

P

pack().Method in class java.awt.Window
Packs the components of the Window.
paint(Graphics). Method in class java.awt.Canvas
Paints the canvas in the default background color.
paint(Graphics). Method in class java.awt.Component
Paints the component.
paint(Graphics). Method in interface java.awt.peer.ComponentPeer
paintAll(Graphics). Method in class java.awt.Component
Paints the component and its subcomponents.
paintComponents(Graphics). Method in class java.awt.Container
Paints the components in this container.
Panel(). Constructor for class java.awt.Panel
Creates a new panel.
paramString(). Method in class java.awt.Button
Returns the parameter String of this button.
paramString(). Method in class java.awt.Checkbox
Returns the parameter String of this Checkbox.
paramString(). Method in class java.awt.CheckboxMenuItem
Returns the parameter String of this button.
paramString(). Method in class java.awt.Choice
Returns the parameter String of this Choice.
paramString(). Method in class java.awt.Component
Returns the parameter String of this Component.
paramString(). Method in class java.awt.Container
Returns the parameter String of this Container.
paramString(). Method in class java.awt.Dialog
Returns the parameter String of this Dialog.
paramString(). Method in class java.awt.Event
Returns the parameter String of this Event.
paramString(). Method in class java.awt.FileDialog
Returns the parameter String of this file dialog.
paramString(). Method in class java.awt.Frame
Returns the parameter String of this Frame.
paramString(). Method in class java.awt.Label
Returns the parameter String of this label.
paramString(). Method in class java.awt.List
Returns the parameter String of this list.
paramString(). Method in class java.awt.MenuComponent
Returns the String parameter of this MenuComponent.
paramString(). Method in class java.awt.MenuItem
Returns the String parameter of the menu item.
paramString(). Method in class java.awt.Scrollbar
Returns the String parameters for this Scrollbar.
paramString(). Method in class java.awt.TextArea
Returns the String of parameters for this TextArea.
paramString(). Method in class java.awt.TextComponent
Returns the String of parameters for this TextComponent.
paramString(). Method in class java.awt.TextField
Returns the String of parameters for this TExtField.
parentOf(ThreadGroup). Method in class java.lang.ThreadGroup
Checks to see if this Thread group is a parent of or is equal to another Thread group.
parse(String). Static method in class java.util.Date

Given a string representing a time, parse it and return the time value.
parseInt(String). Static method in class java.lang.Integer
Assuming the specified String represents an integer, returns that integer's value.
parseInt(String, int). Static method in class java.lang.Integer
Assuming the specified String represents an integer, returns that integer's value.
parseLong(String). Static method in class java.lang.Long
Assuming the specified String represents a long, return that long's value.
parseLong(String, int). Static method in class java.lang.Long
Assuming the specified String represents a long, returns that long's value.
parseNumbers(). Method in class java.io.StreamTokenizer
Specifies that numbers should be parsed.
parseURL(URL, String, int, int). Method in class java.net.URLStreamHandler
This method is called to parse the string spec into URL u.
pathSeparator. Static variable in class java.io.File
The system dependent path separator string.
pathSeparatorChar. Static variable in class java.io.File
The system dependent path separator character.
peek(). Method in class java.util.Stack
Peeks at the top of the stack.
PGDN. Static variable in class java.awt.Event
The page down key.
PGUP. Static variable in class java.awt.Event
The page up key.
PI. Static variable in class java.lang.Math
The float representation of the value Pi.
pink. Static variable in class java.awt.Color
The color pink.
PipedInputStream(). Constructor for class java.io.PipedInputStream
Creates an input file that isn't connected to anything (yet).
PipedInputStream(PipedOutputStream). Constructor for class java.io. PipedInputStream
Creates an input file from the specified PiledOutputStream.
PipedOutputStream(). Constructor for class java.io.PipedOutputStream
Creates an output file that isn't connected to anything (yet).
PipedOutputStream(PipedInputStream). Constructor for class java.io. PipedOutputStream
Creates an output file connected to the specified PipedInputStream.
pixel_bits. Variable in class java.awt.image.ColorModel
PixelGrabber (Image, int, int, int, int, int[], int, int). Constructor for class java.awt.image.PixelGrabber
Create a PixelGrabber object to grab the (x, y, w, h) rectangular section of pixels from the specified image into the given array.
PixelGrabber(ImageProducer, int, int, int, int, int[], int, int). Constructor for class java.awt.image.PixelGrabber
Create a PixelGrabber object to grab the (x, y, w, h) rectangular section of pixels from the image produced by the specified ImageProducer into the given array.
PLAIN. Static variable in class java.awt.Font
The plain style constant.
play(). Method in interface java.applet.AudioClip
Starts playing the clip.
play(URL). Method in class java.applet.Applet
Plays an audio clip.
play(URL, String). Method in class java.applet.Applet
Plays an audio clip.
Point(int, int). Constructor for class java.awt.Point
Constructs and initializes a Point from the specified x and y coordinates.
Polygon(). Constructor for class java.awt.Polygon
Creates an empty polygon.
Polygon(int[], int[], int). Constructor for class java.awt.Polygon

preferredSize(int, int). Method in class java.awt. TextArea
Returns the specified row and column Dimensions of the TextArea.
preferredSize(int, int). Method in interface java.awt.peer.TextAreaPeer
prepareImage(Image, ImageObserver). Method in class java.awt.Component
Prepares an image for rendering on this Component.
prepareImage(Image, int, int, ImageObserver). Method in class java.awt. Component
Prepares an image for rendering on this Component at the specified width and height.
prepareImage(Image, int, int, ImageObserver). Method in interface java.awt.peer.ComponentPeer
prepareImage(Image, int, int, ImageObserver). Method in class java.awt.Toolkit
Prepares an image for rendering on the default screen at the specified width and height.
previous(Container). Method in class java.awt.CardLayout
Flips to the previous card of the specified container.
print(boolean). Method in class java.io.PrintStream
Prints a boolean.
print(char). Method in class java.io.PrintStream
Prints an character.
print(char[]). Method in class java.io.PrintStream
Prints an array of characters.
print(double). Method in class java.io.PrintStream
Prints a double.
print(float). Method in class java.io.PrintStream
Prints a float.
print(Graphics). Method in class java.awt.Component
Prints this component.
print(Graphics). Method in interface java.awt.peer.ComponentPeer
print(int). Method in class java.io.PrintStream
Prints an integer.
print(long). Method in class java.io.PrintStream
Prints a long.
print(Object). Method in class java.io.PrintStream
Prints an object.
print(String). Method in class java.io.PrintStream
Prints a String.
printAll(Graphics). Method in class java.awt.Component
Prints the component and its subcomponents.
printComponents(Graphics). Method in class java.awt.Container
Prints the components in this container.
println(). Method in class java.io.PrintStream
Prints a newline.
println(boolean). Method in class java.io.PrintStream
Prints a boolean followed by a newline.
println(char). Method in class java.io.PrintStream
Prints a character followed by a newline.
println(char[]). Method in class java.io.PrintStream
Prints an array of characters followed by a newline.
println(double). Method in class java.io.PrintStream
Prints a double followed by a newline.
println(float). Method in class java.io.PrintStream
Prints a float followed by a newline.
println(int). Method in class java.io.PrintStream
Prints an integer followed by a newline.
println(long). Method in class java.io.PrintStream
Prints a long followed by a newline.
println(Object). Method in class java.io.PrintStream
Prints an object followed by a newline.
println(String). Method in class java.io.PrintStream

Q

R

Creates a new random number generator using a single `long` seed.

RandomAccessFile(File, String). Constructor for class java.io.RandomAccessFile
Creates a RandomAccessFile from a specified File object and mode ("r" or "rw").

RandomAccessFile(String, String). Constructor for class java.io. RandomAccessFile
Creates a RandomAccessFile with the specified system dependent file name and the specified mode.

RANDOMPIXELORDER. Static variable in interface java.awt.image.ImageConsumer
The pixels will be delivered in a random order.

read(). Method in class java.io.BufferedInputStream
Reads a byte of data.

read(). Method in class java.io.ByteArrayInputStream
Reads a byte of data.

read(). Method in class java.io.FileInputStream
Reads a byte of data.

read(). Method in class java.io.FilterInputStream
Reads a byte.

read(). Method in class java.io.InputStream
Reads a byte of data.

read(). Method in class java.io.LineNumberInputStream
Reads a byte of data.

read(). Method in class java.io.PipedInputStream
Reads a byte of data.

read(). Method in class java.io.PushbackInputStream
Reads a byte of data.

read(). Method in class java.io.RandomAccessFile
Reads a byte of data.

read(). Method in class java.io.SequenceInputStream
Reads a stream, and upon reaching an EOF, flips to the next stream.

read(). Method in class java.io.StringBufferInputStream
Reads a byte of data.

read(byte[]). Method in class java.io.DataInputStream
Reads data into an array of bytes.

read(byte[]). Method in class java.io.FileInputStream
Reads data into an array of bytes.

read(byte[]). Method in class java.io.FilterInputStream
Reads into an array of bytes.

read(byte[]). Method in class java.io.InputStream
Reads into an array of bytes.

read(byte[]). Method in class java.io.RandomAccessFile
Reads data into an array of bytes.

read(byte[], int, int). Method in class java.io.BufferedInputStream
Reads into an array of bytes.

read(byte[], int, int). Method in class java.io.ByteArrayInputStream
Reads into an array of bytes.

read(byte[], int, int). Method in class java.io.DataInputStream
Reads data into an array of bytes.

read(byte[], int, int). Method in class java.io.FileInputStream
Reads data into an array of bytes.

read(byte[], int, int). Method in class java.io.FilterInputStream
Reads into an array of bytes.

read(byte[], int, int). Method in class java.io.InputStream
Reads into an array of bytes.

read(byte[], int, int). Method in class java.io.LineNumberInputStream
Reads into an array of bytes.

read(byte[], int, int). Method in class java.io.PipedInputStream
Reads into an array of bytes.

read(byte[], int, int). Method in class java.io.PushbackInputStream
Reads into an array of bytes.
read(byte[], int, int). Method in class java.io.RandomAccessFile
Reads a sub array as a sequence of bytes.
read(byte[], int, int). Method in class java.io.SequenceInputStream
Reads data into an array of bytes, and upon reaching an EOF, flips to the next stream.
read(byte[], int, int). Method in class java.io.StringBufferInputStream
Reads into an array of bytes.
readBoolean(). Method in interface java.io.DataInput
Reads in a boolean.
readBoolean(). Method in class java.io.DataInputStream
Reads a boolean.
readBoolean(). Method in class java.io.RandomAccessFile
Reads a boolean.
readByte(). Method in interface java.io.DataInput
Reads an 8 bit byte.
readByte(). Method in class java.io.DataInputStream
Reads an 8 bit byte.
readByte(). Method in class java.io.RandomAccessFile
Reads a byte.
readChar(). Method in interface java.io.DataInput
Reads a 16 bit char.
readChar(). Method in class java.io.DataInputStream
Reads a 16 bit char.
readChar(). Method in class java.io.RandomAccessFile
Reads a 16 bit char.
readDouble(). Method in interface java.io.DataInput
Reads a 64 bit double.
readDouble(). Method in class java.io.DataInputStream
Reads a 64 bit double.
readDouble(). Method in class java.io.RandomAccessFile
Reads a 64 bit double.
readFloat(). Method in interface java.io.DataInput
Reads a 32 bit float.
readFloat(). Method in class java.io.DataInputStream
Reads a 32 bit float.
readFloat(). Method in class java.io.RandomAccessFile
Reads a 32 bit float.
readFully(byte[]). Method in interface java.io.DataInput
Reads bytes, blocking until all bytes are read.
readFully(byte[]). Method in class java.io.DataInputStream
Reads bytes, blocking until all bytes are read.
readFully(byte[]). Method in class java.io.RandomAccessFile
Reads bytes, blocking until all bytes are read.
readFully(byte[], int, int). Method in interface java.io.DataInput
Reads bytes, blocking until all bytes are read.
readFully(byte[], int, int). Method in class java.io.DataInputStream
Reads bytes, blocking until all bytes are read.
readFully(byte[], int, int). Method in class java.io.RandomAccessFile
Reads bytes, blocking until all bytes are read.
readInt(). Method in interface java.io.DataInput
Reads a 32 bit int.
readInt(). Method in class java.io.DataInputStream
Reads a 32 bit int.
readInt(). Method in class java.io.RandomAccessFile
Reads a 32 bit int.

readLine(). Method in interface java.io.DataInput
readLine(). Method in class java.io.DataInputStream
Reads in a line that has been terminated by a \n, \r, \r\n or EOF.
readLine(). Method in class java.io.RandomAccessFile
Reads a line terminated by a '\n' or EOF.
readLong(). Method in interface java.io.DataInput
Reads a 64 bit long.
readLong(). Method in class java.io.DataInputStream
Reads a 64 bit long.
readLong(). Method in class java.io.RandomAccessFile
Reads a 64 bit long.
readShort(). Method in interface java.io.DataInput
Reads a 16 bit short.
readShort(). Method in class java.io.DataInputStream
Reads a 16 bit short.
readShort(). Method in class java.io.RandomAccessFile
Reads 16 bit short.
readUnsignedByte(). Method in interface java.io.DataInput
Reads an unsigned 8 bit byte.
readUnsignedByte(). Method in class java.io.DataInputStream
Reads an unsigned 8 bit byte.
readUnsignedByte(). Method in class java.io.RandomAccessFile
Reads an unsigned 8 bit byte.
readUnsignedShort(). Method in interface java.io.DataInput
Reads an unsigned 16 bit short.
readUnsignedShort(). Method in class java.io.DataInputStream
Reads 16 bit short.
readUnsignedShort(). Method in class java.io.RandomAccessFile
Reads 16 bit short.
readUTF(). Method in interface java.io.DataInput
readUTF(). Method in class java.io.DataInputStream
Reads a UTF format String.
readUTF(). Method in class java.io.RandomAccessFile
Reads a UTF formatted String.
readUTF(DataInput). Static method in class java.io.DataInputStream
Reads a UTF format String from the given input stream.
receive(DatagramPacket). Method in class java.net.DatagramSocket
Receives datagram packet.
Rectangle(). Constructor for class java.awt.Rectangle
Constructs a new rectangle.
Rectangle(Dimension). Constructor for class java.awt.Rectangle
Constructs a rectangle and initializes it to the specified width and height.
Rectangle(int, int). Constructor for class java.awt.Rectangle
Constructs a rectangle and initializes it with the specified width and height parameters.
Rectangle(int, int, int, int). Constructor for class java.awt.Rectangle
Constructs and initializes a rectangle with the specified parameters.
Rectangle(Point). Constructor for class java.awt.Rectangle
Constructs a rectangle and initializes it to the specified point.
Rectangle(Point, Dimension). Constructor for class java.awt.Rectangle
Constructs a rectangle and initializes it to a specified point and dimension.
red. Static variable in class java.awt.Color
The color red.
regionMatches(boolean, int, String, int, int). Method in class java.lang. String
Determines whether a region of this String matches the specified region of the specified String.
regionMatches(int, String, int, int). Method in class java.lang.String
Determines whether a region of this String matches the specified region of the specified String.

rehash(). Method in class java.util.Hashtable
Rehashes the content of the table into a bigger table.
RELATIVE. Static variable in class java.awt.GridBagConstraints
REMAINDER. Static variable in class java.awt. GridBagConstraints
RemoteDebugger(String, DebuggerCallback, boolean). Constructor for class sun.tools.debug.RemoteDebugger
Create a remote debugger, and connect it to a new client interpreter.

RemoteDebugger (String, String, DebuggerCallback, boolean). Constructor for class sun.tools.debug.RemoteDebugger
Create a remote debugger, connecting it with a running Java interpreter.

To connect to a running interpreter, it must be started with the "-debug" option, whereupon it will print out the password for that debugging session.
RemoteInt(int). Constructor for class sun.tools.debug.RemoteInt
remove(Component). Method in class java.awt.Container
Removes the specified component from this container.
remove(int). Method in class java.awt.Menu
Deletes the item from this menu at the specified index.
remove(int). Method in class java.awt.MenuBar
Removes the menu located at the specified index from the menu bar.
remove(MenuComponent). Method in class java.awt.Frame
Removes the specified menu bar from this Frame.
remove(MenuComponent). Method in class java.awt.Menu
Deletes the specified item from this menu.
remove(MenuComponent). Method in class java.awt.MenuBar
Removes the specified menu from the menu bar.
remove(MenuComponent). Method in interface java.awt.MenuContainer
remove(Object). Method in class java.util.Dictionary
Removes the element corresponding to the key.
remove(Object). Method in class java.util.Hashtable
Removes the element corresponding to the key.
removeAll(). Method in class java.awt.Container
Removes all the components from this container.
removeAllElements(). Method in class java.util.Vector
Removes all elements of the vector.
removeConsumer(ImageConsumer). Method in class java.awt.image. FilteredImageSource
Removes an ImageConsumer from the list of consumers interested in data for this image.
removeConsumer(ImageConsumer). Method in interface java.awt.image.ImageProducer
This method removes the given ImageConsumer object from the list of consumers currently registered to receive image data.
removeConsumer(ImageConsumer). Method in class java.awt.image.MemoryImageSource
Remove an ImageConsumer from the list of consumers interested in data for this image.
removeElement(Object). Method in class java.util.Vector
Removes the element from the vector.
removeElementAt(int). Method in class java.util.Vector
Deletes the element at the specified index.
removeLayoutComponent(Component). Method in class java.awt.BorderLayout
Removes the specified component from the layout.
removeLayoutComponent(Component). Method in class java.awt.CardLayout
Removes the specified component from the layout.
removeLayoutComponent(Component). Method in class java.awt.FlowLayout
Removes the specified component from the layout.
removeLayoutComponent(Component). Method in class java.awt.GridBagLayout
Removes the specified component from the layout.
removeLayoutComponent(Component). Method in class java.awt.GridLayout

Removes the specified component from the layout.
removeLayoutComponent(Component). Method in interface java.awt.LayoutManager
Removes the specified component from the layout.
removeNotify(). Method in class java.awt.Component
Notifies the Component to destroy the peer.
removeNotify(). Method in class java.awt.Container
Notifies the container to remove its peer.
removeNotify(). Method in class java.awt.List
Removes the peer for this list.
removeNotify(). Method in class java.awt.Menu
Removes the menu's peer.
removeNotify(). Method in class java.awt.MenuBar
Removes the menu bar's peer.
removeNotify(). Method in class java.awt.MenuComponent
Removes the menu component's peer.
removeNotify(). Method in class java.awt.TextComponent
Removes the TextComponent's peer.
renameTo(File). Method in class java.io.File
Renames a file and returns a boolean indicating whether or not this method was successful.
repaint(). Method in class java.awt.Component
Repaints the component.
repaint(int, int, int, int). Method in class java.awt.Component
Repaints part of the component.
repaint(long). Method in class java.awt.Component
Repaints the component.
repaint(long, int, int, int, int). Method in class java.awt.Component
Repaints part of the component.
repaint(long, int, int, int, int). Method in interface java.awt.peer. ComponentPeer
replace(char, char). Method in class java.lang.String
Converts this String by replacing all occurences of oldChar with newChar.
replaceItem(String, int). Method in class java.awt.List
Replaces the item at the given index.
replaceText(String, int, int). Method in class java.awt.TextArea
Replaces text from the indicated start to end position with the new text specified.
replaceText(String, int, int). Method in interface java.awt.peer.TextAreaPeer
requestFocus(). Method in class java.awt.Component
Requests the input focus.
requestFocus(). Method in interface java.awt.peer.ComponentPeer
requestTopDownLeftRightResend(ImageConsumer). Method in class java.awt.image.FilteredImageSource
Requests that a given ImageConsumer have the image data delivered one more time in top-down, left-right order.
requestTopDownLeftRightResend(ImageConsumer). Method in interface java.awt.image.ImageProducer
This method is used by an ImageConsumer to request that the ImageProducer attempt to resend the image data one more time in TOPDOWNLEFTRIGHT order so that higher quality conversion algorithms which depend on receiving pixels in order can be used to produce a better output version of the image.
requestTopDownLeftRightResend(ImageConsumer). Method in class java.awt.image. MemoryImageSource
Requests that a given ImageConsumer have the image data delivered one more time in top-down, left-right order.
resendTopDownLeftRight(ImageProducer). Method in class java.awt.image. ImageFilter
Responds to a request for a TopDownLeftRight (TDLR) ordered resend of the pixel data from an ImageConsumer.
reset(). Method in class java.io.BufferedInputStream

Repositions the stream to the last marked position.
reset(). Method in class java.io.ByteArrayInputStream
Resets the buffer to the beginning.
reset(). Method in class java.io.ByteArrayOutputStream
Resets the buffer so that you can use it again without throwing away the already allocated buffer.
reset(). Method in class java.io.FilterInputStream
Repositions the stream to the last marked position.
reset(). Method in class java.io.InputStream
Repositions the stream to the last marked position.
reset(). Method in class java.io.LineNumberInputStream
Repositions the stream to the last marked position.
reset(). Method in class java.io.StringBufferInputStream
Resets the buffer to the beginning.
resetCurrentFrameIndex(). Method in class sun.tools.debug.RemoteThread
Reset the current stackframe
resetSyntax(). Method in class java.io.StreamTokenizer
Resets the syntax table so that all characters are special.
reshape(int, int, int, int). Method in class java.awt.Component
Reshapes the Component to the specified bounding box.
reshape(int, int, int, int). Method in interface java.awt.peer.ComponentPeer
reshape(int, int, int, int). Method in class java.awt. Rectangle
Reshapes the rectangle.
resize(Dimension). Method in class java.applet.Applet
Requests thatthe applet be resized.
resize(Dimension). Method in class java.awt.Component
Resizes the Component to the specified dimension.
resize(int, int). Method in class java.applet.Applet
Requests that the applet be resized.
resize(int, int). Method in class java.awt.Component
Resizes the Component to the specified width and height.
resize(int, int). Method in class java.awt.Rectangle
Resizes the rectangle.
resolveClass(Class). Method in class java.lang.ClassLoader
Resolves classes referenced by this Class.
resume(). Method in class sun.tools.debug.RemoteThread
Resume execution of this thread.
resume(). Method in class java.lang.Thread
Resumes this Thread execution.
resume(). Method in class java.lang.ThreadGroup
Resumes all the Threads in this Thread group and all of its sub groups.
RGBImageFilter(). Constructor for class java.awt.image.RGBImageFilter
RGBtoHSB(int, int, int, float[]). Static method in class java.awt.Color
Returns the HSB values corresponding to the color defined by the red, green, and blue components.
RIGHT. Static variable in class java.awt.Event
The right arrow key.
RIGHT. Static variable in class java.awt.FlowLayout
The right alignment variable.
right. Variable in class java.awt.Insets
The inset from the right.
RIGHT. Static variable in class java.awt.Label
The right alignment.
rint(double). Static method in class java.lang.Math
Converts a double value into an integral value in double format.
round(double). Static method in class java.lang.Math
Rounds off a double value by first adding 0.5 to it and then returning the largest integer that is less than or equal to this new value.

round(float). Static method in class java.lang.Math
Rounds off a float value by first adding 0.5 to it and then returning the largest integer that is less than or equal to this new value.
rowHeights. Variable in class java.awt.GridBagLayout
rowWeights. Variable in class java.awt.GridBagLayout
run(). Method in interface java.lang.Runnable
The method that is executed when a Runnable object is activated.
run(). Method in class java.lang.Thread
The actual body of this Thread.
run(int, String[]). Method in class sun.tools.debug.RemoteDebugger
Load and run a runnable Java class, with any optional parameters.
runFinalization(). Method in class java.lang.Runtime
Runs the finalization methods of any objects pending finalization.
runFinalization(). Static method in class java.lang.System
Runs the finalization methods of any objects pending finalization.
RuntimeException(). Constructor for class java.lang.RuntimeException
Constructs a RuntimeException with no detail message.
RuntimeException(String). Constructor for class java.lang.RuntimeException
Constructs a RuntimeException with the specified detail message.

S

S_RESIZE_CURSOR.Static variable in class java.awt.Frame
sameFile(URL). Method in class java.net.URL
Compares two URLs, excluding the "ref" fields: sameFile is true if the true references the same remote object, but not necessarily the same subpiece of that object.
SAVE. Static variable in class java.awt.FileDialog
The file save variable.
save(OutputStream, String). Method in class java.util.Properties
Save properties to an OutputStream.
SAVE_FILE. Static variable in class java.awt.Event
A file saving event.
SCROLL_ABSOLUTE. Static variable in class java.awt.Event
The absolute scroll event.
SCROLL_LINE_DOWN. Static variable in class java.awt.Event
The line down scroll event.
SCROLL_LINE_UP. Static variable in class java.awt.Event
The line up scroll event.
SCROLL_PAGE_DOWN. Static variable in class java.awt.Event
The page down scroll event.
SCROLL_PAGE_UP. Static variable in class java.awt.Event
The page up scroll event.
Scrollbar(). Constructor for class java.awt.Scrollbar
Constructs a new vertical Scrollbar.
Scrollbar(int). Constructor for class java.awt.Scrollbar
Constructs a new Scrollbar with the specified orientation.
Scrollbar(int, int, int, int, int). Constructor for class java.awt.Scrollbar
Constructs a new Scrollbar with the specified orientation, value, page size, and minumum and maximum values.
SE_RESIZE_CURSOR. Static variable in class java.awt.Frame
search(Object). Method in class java.util.Stack
Sees if an object is on the stack.
SecurityException(). Constructor for class java.lang.SecurityException

Constructs a SecurityException with no detail message.
SecurityException(String). Constructor for class java.lang.SecurityException
Constructs a SecurityException with the specified detail message.
SecurityManager(). Constructor for class java.lang.SecurityManager
Constructs a new SecurityManager.
seek(long). Method in class java.io.RandomAccessFile
Sets the file pointer to the specified absolute position.
select(int). Method in class java.awt.Choice
Selects the item with the specified postion.
select(int). Method in interface java.awt.peer.ChoicePeer
select(int). Method in class java.awt.List
Selects the item at the specified index.
select(int). Method in interface java.awt.peer.ListPeer
select(int, int). Method in class java.awt. TextComponent
Selects the text found between the specified start and end locations.
select(int, int). Method in interface java.awt.peer.TextComponentPeer
select(String). Method in class java.awt.Choice
Selects the item with the specified String.
selectAll(). Method in class java.awt.TextComponent
Selects all the text in the TextComponent.
send(DatagramPacket). Method in class java.net.DatagramSocket
Sends Datagram Packet to the destination address
separator. Static variable in class java.io.File
The system dependent file separator String.
separatorChar. Static variable in class java.io.File
The system dependent file separator character.
SequenceInputStream(Enumeration). Constructor for class java.io. SequenceInputStream
Constructs a new SequenceInputStream initialized to the specified list.
SequenceInputStream(InputStream, InputStream). Constructor for class java.io. SequenceInputStream
Constructs a new SequenceInputStream initialized to the two specified input streams.
ServerSocket(int). Constructor for class java.net.ServerSocket
Creates a server socket on a specified port.
ServerSocket(int, int). Constructor for class java.net.ServerSocket
Creates a server socket, binds it to the specified local port and listens to it.
set(int). Method in class java.util.BitSet
Sets a bit.
set(String, String, int, String, String). Method in class java.net.URL
Sets the fields of the URL.
setAlignment(int). Method in class java.awt.Label
Sets the alignment for this label to the specified alignment.
setAlignment(int). Method in interface java.awt.peer.LabelPeer
setAllowUserInteraction(boolean). Method in class java.net.URLConnection
Some URL connections occasionally need to to interactions with the user.
setBackground(Color). Method in class java.awt.Component
Sets the background color.
setBackground(Color). Method in interface java.awt.peer.ComponentPeer
setBreakpointLine(int). Method in class sun.tools.debug.RemoteClass
Set a breakpoint at a specified source line number in a class.
setBreakpointMethod(RemoteField). Method in class sun.tools.debug.RemoteClass
Set a breakpoint at the first line of a class method.
setChanged(). Method in class java.util.Observable
Sets a flag to note an observable change.
setCharAt(int, char). Method in class java.lang.StringBuffer
Changes the character at the specified index to be ch.
setCheckboxGroup(CheckboxGroup). Method in class java.awt.Checkbox
Sets the CheckboxGroup to the specified group.

setCheckboxGroup(CheckboxGroup). Method in interface java.awt.peer.CheckboxPeer
setColor(Color). Method in class java.awt.Graphics
Sets the current color to the specified color.
setColorModel(ColorModel). Method in interface java.awt.image.ImageConsumer
The ColorModel object used for the majority of the pixels reported using the setPixels method calls.
setColorModel(ColorModel). Method in class java.awt.image.ImageFilter
Filter the information provided in the setColorModel method of the ImageConsumer interface.
setColorModel(ColorModel). Method in class java.awt.image.PixelGrabber
The setColorModel method is part of the ImageConsumer API which this class must implement to retrieve the pixels.
setColorModel(ColorModel). Method in class java.awt.image.RGBImageFilter
If the ColorModel is an IndexColorModel, and the subclass has set the canFilterIndexColorModel flag to true, we substitute a filtered version of the color model here and wherever that original ColorModel object appears in the setPixels methods.
setConstraints(Component, GridBagConstraints). Method in class java.awt. GridBagLayout
Sets the constraints for the specified component.
setContentHandlerFactory(ContentHandlerFactory). Static method in class java.net.URLConnection
Sets the ContentHandler factory.
setCurrent(Checkbox). Method in class java.awt.CheckboxGroup
Sets the current choice to the specified Checkbox.
setCurrentFrameIndex(int). Method in class sun.tools.debug.RemoteThread
Set the current stackframe index
setCursor(int). Method in class java.awt.Frame
Set the cursor image to a predefined cursor.
setCursor(int). Method in interface java.awt.peer.FramePeer
setDaemon(boolean). Method in class java.lang.Thread
Marks this Thread as a daemon Thread or a user Thread.
setDaemon(boolean). Method in class java.lang.ThreadGroup
Changes the daemon status of this group.
setDate(int). Method in class java.util.Date
Sets the date.
setDefaultAllowUserInteraction(boolean). Static method in class java.net. URLConnection
Sets/gets the default value of the allowUserInteraction flag.
setDefaultRequestProperty(String, String). Static method in class java.net. URLConnection
Sets/gets the default value of a general request property.
setDefaultUseCaches(boolean). Method in class java.net.URLConnection
setDimensions(int, int). Method in class java.awt.image.CropImageFilter
Override the source image's dimensions and pass the dimensions of the rectangular cropped region to the ImageConsumer.
setDimensions(int, int). Method in interface java.awt.image.ImageConsumer
The dimensions of the source image are reported using the setDimensions method call.
setDimensions(int, int). Method in class java.awt.image.ImageFilter
Filters the information provided in the setDimensions method of the ImageConsumer interface.
setDimensions(int, int). Method in class java.awt.image.PixelGrabber
The setDimensions method is part of the ImageConsumer API which this class must implement to retrieve the pixels.
setDirectory(String). Method in class java.awt.FileDialog
Set the directory of the Dialog to the specified directory.
setDirectory(String). Method in interface java.awt.peer.FileDialogPeer
setDoInput(boolean). Method in class java.net. URLConnection
A URL connection can be used for input and/or output.
setDoOutput(boolean). Method in class java.net.URLConnection
A URL connection can be used for input and/or output.
setEchoCharacter(char). Method in class java.awt.TextField
Sets the echo character for this TextField.
setEchoCharacter(char). Method in interface java.awt.peer.TextFieldPeer

setEditable(boolean). Method in class java.awt. TextComponent
Sets the specified boolean to indicate whether or not this TextComponent should be editable.
setEditable(boolean). Method in interface java.awt.peer.TextComponentPeer
setElementAt(Object, int). Method in class java.util. Vector
Sets the element at the specified index to be the specified object.
setFile(String). Method in class java.awt.FileDialog
Sets the file for this dialog to the specified file.
setFile(String). Method in interface java.awt.peer.FileDialogPeer
setFilenameFilter(FilenameFilter). Method in class java.awt.FileDialog
Sets the filter for this dialog to the specified filter.
setFilenameFilter(FilenameFilter). Method in interface java.awt.peer. FileDialogPeer
setFont(Font). Method in class java.awt.Component
Sets the font of the component.
setFont(Font). Method in interface java.awt.peer.ComponentPeer
setFont(Font). Method in class java.awt.Graphics
Sets the font for all subsequent text-drawing operations.
setFont(Font). Method in class java.awt.MenuComponent
Sets the font to be used for this MenuItem to the specified font.
setForeground(Color). Mehod in class java.awt.Component
Sets the foreground color.
setForeground(Color). Method in interface java.awt.peer.ComponentPeer
setHelpMenu(Menu). Method in class java.awt.MenuBar
Sets the help menu to the specified menu on the menu bar.
setHints(int). Method in interface java.awt.image.ImageConsumer
The ImageProducer can deliver the pixels in any order, but the ImageConsumer may be able to scale or convert the pixels to the destination ColorModel more efficiently or with higher quality if it knows some information about how the pixels will be delivered up front.
setHints(int). Method in class java.awt.image.ImageFilter
Filters the information provided in the setHints method of the ImageConsumer interface.
setHints(int). Method in class java.awt.image.PixelGrabber
The setHints method is part of the ImageConsumer API which this class must implement to retrieve the pixels.
setHours(int). Method in class java.util.Date
Sets the hours.
setIconImage(Image). Method in class java.awt.Frame
Sets the image to display when this Frame is iconized.
setIconImage(Image). Method in interface java.awt.peer.FramePeer
setIfModifiedSince(long). Method in class java.net. URLConnection
Some protocols support skipping fetching unless the object is newer than some amount of time.
setLabel(String). Method in class java.awt.Button
Sets the button with the specified label.
setLabel(String). Method in interface java.awt.peer.ButtonPeer
setLabel(String). Method in class java.awt.Checkbox
Sets the button with the specified label.
setLabel(String). Method in interface java.awt.peer.CheckboxPeer
setLabel(String). Method in class java.awt.MenuItem
Sets the label to be the specified label.
setLabel(String). Method in interface java.awt.peer.MenuItemPeer
setLayout(LayoutManager). Method in class java.awt. Container
Sets the layout manager for this container.
setLength(int). Method in class java.lang.StringBuffer
Sets the length of the String.
setLineIncrement(int). Method in class java.awt.Scrollbar
Sets the line increment for this scrollbar.
setLineIncrement(int). Method in interface java.awt.peer.ScrollbarPeer
setLineNumber(int). Method in class java.io. LineNumberInputStream

Sets the current line number.
setMaxPriority(int). Method in class java.lang.ThreadGroup
Sets the maximum priority of the group.
setMenuBar(MenuBar). Method in class java.awt.Frame
Sets the menubar for this Frame to the specified menubar.
setMenuBar(MenuBar). Method in interface java.awt.peer.FramePeer
setMinutes(int). Method in class java.util.Date
Sets the minutes.
setMonth(int). Method in class java.util.Date
Sets the month.
setMultipleSelections(boolean). Method in class java.awt.List
Sets whether this list should allow multiple selections or not.
setMultipleSelections(boolean). Method in interface java.awt.peer.ListPeer
setName(String). Method in class java.lang.Thread
Sets the Thread's name.
setPageIncrement(int). Method in class java.awt.Scrollbar
Sets the page increment for this scrollbar.
setPageIncrement(int). Method in interface java.awt.peer.ScrollbarPeer
setPaintMode(). Method in class java.awt.Graphics
Sets the paint mode to overwrite the destination with the current color.
setPixels(int, int, int, int, ColorModel, byte[], int, int). Method in class java.awt.image.CropImageFilter
Determine whether the delivered byte pixels intersect the region to be extracted and passes through only that subset of pixels that appear in the output region.
setPixels(int, int, int, int, ColorModel, byte[], int, int). Method in interface java.awt.image.ImageConsumer
The pixels of the image are delivered using one or more calls to the setPixels method.
setPixels(int, int, int, int, ColorModel, byte[], int, int). Method in class java.awt.image.ImageFilter
Filters the information provided in the setPixels method of the ImageConsumer interface which takes an array of bytes.
setPixels(int, int, int, int, ColorModel, byte[], int, int). Method in class java.awt.image.PixelGrabber
The setPixels method is part of the ImageConsumer API which this class must implement to retrieve the pixels.
setPixels(int, int, int, int, ColorModel, byte[], int, int). Method in class java.awt.image.RGBImageFilter
If the ColorModel object is the same one that has already been converted, then simply passes the pixels through with the converted ColorModel.
setPixels(int, int, int, int, ColorModel, int[], int, int). Method in class java.awt.image.CropImageFilter
Determine if the delivered int pixels intersect the region to be extracted and pass through only that subset of pixels that appear in the output region.
setPixels(int, int, int, int, ColorModel, int[], int, int). Method in interface java.awt.image.ImageConsumer
The pixels of the image are delivered using one or more calls to the setPixels method.
setPixels(int, int, int, int, ColorModel, int[], int, int). Method in class java.awt.image.ImageFilter
Filters the information provided in the setPixels method of the ImageConsumer interface which takes an array of integers.
setPixels(int, int, int, int, ColorModel, int[], int, int). Method in class java.awt.image.PixelGrabber
The setPixels method is part of the ImageConsumer API which this class must implement to retrieve the pixels.
setPixels(int, int, int, int, ColorModel, int[], int, int). Method in class java.awt.image.RGBImageFilter
If the ColorModel object is the same one that has already been converted, then simply passes the pixels through with the converted ColorModel, otherwise converts the buffer of integer pixels to the default RGB ColorModel and passes the converted buffer to the filterRGBPixels method to be converted one by one.
setPriority(int). Method in class java.lang.Thread
Sets the Thread's priority.
setProperties(Hashtable). Method in class java.awt.image.CropImageFilter
Passes along the properties from the source object after adding a property indicating the cropped region.
setProperties(Hashtable). Method in interface java.awt.image.ImageConsumer

Sets the extensible list of properties associated with this image.
setProperties(Hashtable). Method in class java.awt.image.ImageFilter
Passes the properties from the source object along after adding a property indicating the stream of filters it has been run through.
setProperties(Hashtable). Method in class java.awt.image.PixelGrabber
The setProperties method is part of the ImageConsumer API which this class must implement to retrieve the pixels.
setProperties(Properties). Static method in class java.lang.System
Sets the System properties to the specified properties.
setRequestProperty(String, String). Method in class java.net.URLConnection
Sets/gets a general request property.
setResizable(boolean). Method in class java.awt.Dialog
Sets the resizable flag.
setResizable(boolean). Method in interface java.awt.peer.DialogPeer
setResizable(boolean). Method in class java.awt.Frame
Sets the resizable flag.
setResizable(boolean). Method in interface java.awt.peer.FramePeer
setSeconds(int). Method in class java.util.Date
Sets the seconds.
setSecurityManager(SecurityManager). Static method in class java.lang.System
Sets the System security.
setSeed(long). Method in class java.util.Random
Sets the seed of the random number generator using a single `long` seed.
setSize(int). Method in class java.util.Vector
Sets the size of the vector.
setSocketFactory(SocketImplFactory). Static method in class java.net. ServerSocket
Sets the system's server SocketImplFactory.
setSocketImplFactory(SocketImplFactory). Static method in class java.net.Socket
Sets the system's client SocketImplFactory.
setSourcePath(String). Method in class sun.tools.debug.RemoteDebugger
Specify the list of paths to use when searching for a source file.
setState(boolean). Method in class java.awt.Checkbox
Sets the Checkbox to the specifed boolean state.
setState(boolean). Method in class java.awt.CheckboxMenuItem
Sets the state of this MenuItem if it is a Checkbox.
setState(boolean). Method in interface java.awt.peer.CheckboxMenuItemPeer
setState(boolean). Method in interface java.awt.peer. CheckboxPeer
setStub(AppletStub). Method in class java.applet. Applet
Sets the applet stub.
setText(String). Method in class java.awt.Label
Sets the text for this label to the specified text.
setText(String). Method in interface java.awt.peer.LabelPeer
setText(String). Method in class java.awt. TextComponent
Sets the text of this TextComponent to the specified text.
setText(String). Method in interface java.awt.peer.TextComponentPeer
setTime(long). Method in class java.util.Date
Sets the time.
setTitle(String). Method in class java.awt.Dialog
Sets the title of the Dialog.
setTitle(String). Method in interface java.awt.peer.DialogPeer
setTitle(String). Method in class java.awt.Frame
Sets the title for this Frame to the specified title.
setTitle(String). Method in interface java.awt.peer.FramePeer
setURL(URL, String, String, int, String, String). Method in class java.net.URLStreamHandler
Calls the (protected) set method out of the URL given.
setURLStreamHandlerFactory(URLStreamHandlerFactory). Static method in class java.net.URL

Sets the URLStreamHandler factory.
setUseCaches(boolean). Method in class java.net.URLConnection
Some protocols do caching of documents.
setValue(int). Method in class java.awt.Scrollbar
Sets the value of this Scrollbar to the specified value.
setValue(int). Method in interface java.awt.peer.ScrollbarPeer
setValues(int, int, int, int). Method in class java.awt.Scrollbar
Sets the values for this Scrollbar.
setValues(int, int, int, int). Method in interface java.awt.peer.ScrollbarPeer
setXORMode(Color). Method in class java.awt.Graphics
Sets the paint mode to alternate between the current color and the new specified color.
setYear(int). Method in class java.util.Date
Sets the year.
SHIFT_MASK. Static variable in class java.awt.Event
The shift modifier constant.
shiftDown(). Method in class java.awt.Event
Checks if the shift key is down.
show(). Method in class java.awt.Component
Shows the component.
show(). Method in interface java.awt.peer.ComponentPeer
show(). Method in class java.awt.Window
Shows the Window.
show(boolean). Method in class java.awt.Component
Conditionally shows the component.
show(Container, String). Method in class java.awt.CardLayout
Flips to the specified component name in the specified container.
showDocument(URL). Method in interface java.applet.AppletContext
Shows a new document.
showDocument(URL, String). Method in interface java.applet.AppletContext
Show a new document in a target window or frame.
showStatus(String). Method in class java.applet.Applet
Shows a status message in the applet's context.
showStatus(String). Method in interface java.applet.AppletContext
Show a status string.
sin(double). Static method in class java.lang.Math
Returns the trigonometric sine of an angle.
SINGLEFRAME. Static variable in interface java.awt.image.ImageConsumer
The image contain a single static image.
SINGLEFRAMEDONE. Static variable in interface java.awt.image.ImageConsumer
One frame of the image is complete but there are more frames to be delivered.
SINGLEPASS. Static variable in interface java.awt.image.ImageConsumer
The pixels will be delivered in a single pass.
size. Variable in class java.awt.Font
The point size of this font.
size(). Method in class java.util.BitSet
Calculates and returns the set's size
size(). Method in class java.io.ByteArrayOutputStream
Returns the current size of the buffer.
size(). Method in class java.awt.Component
Returns the current size of this component.
size(). Method in class java.io.DataOutputStream
Returns the number of bytes written.
size(). Method in class java.util.Dictionary
Returns the number of elements contained within the Dictionary.
size(). Method in class java.util.Hashtable
Returns the number of elements contained in the hashtable.

size(). Method in class java.util.Vector
Returns the number of elements in the vector.
skip(long). Method in class java.io.BufferedInputStream
Skips n bytes of input.
skip(long). Method in class java.io.ByteArrayInputStream
Skips n bytes of input.
skip(long). Method in class java.io.FileInputStream
Skips n bytes of input.
skip(long). Method in class java.io.FilterInputStream
Skips bytes of input.
skip(long). Method in class java.io.InputStream
Skips n bytes of input.
skip(long). Method in class java.io.LineNumberInputStream
Skips n bytes of input.
skip(long). Method in class java.io.StringBufferInputStream
Skips n bytes of input.
skipBytes(int). Method in interface java.io.DataInput
Skips bytes, block until all bytes are skipped.
skipBytes(int). Method in class java.io.DataInputStream
Skips bytes, blocks until all bytes are skipped.
skipBytes(int). Method in class java.io.RandomAccessFile
slashSlashComments(boolean). Method in class java.io. StreamTokenizer
If the flag is true, recognize C++ style(//) comments.
slashStarComments(boolean). Method in class java.io.StreamTokenizer
If the flag is true, recognize C style(/*) comments.
sleep(long). Static method in class java.lang.Thread
Causes the currently executing Thread to sleep for the specified number of milliseconds.
sleep(long, int). Static method in class java.lang.Thread
Sleep, in milliseconds and additional nanosecond.
Socket(InetAddress, int). Constructor for class java.net.Socket
Creates a stream socket and connects it to the specified address on the specified port.
Socket(InetAddress, int, boolean). Constructor for class java.net.Socket
Creates a socket and connects it to the specified address on the specified port.
Socket(String, int). Constructor for class java.net.Socket
Creates a stream socket and connects it to the specified port on the specified host.
Socket(String, int, boolean). Constructor for class java.net.Socket
Creates a socket and connects it to the specified port on the specified host.
SocketException(). Constructor for class java.net.SocketException
Constructs a new SocketException with no detail message.
SocketException(String). Constructor for class java.net.SocketException
Constructs a new SocketException with the specified detail message.
SocketImpl(). Constructor for class java.net.SocketImpl
SOMEBITS. Static variable in interface java.awt.image. ImageObserver
More pixels needed for drawing a scaled variation of the image are available.
SOUTH. Static variable in class java.awt.GridBagConstraints
SOUTHEAST. Static variable in class java.awt. GridBagConstraints
SOUTHWEST. Static variable in class java.awt. GridBagConstraints
sqrt(double). Static method in class java.lang.Math
Returns the square root of a.
Stack(). Constructor for class java.util.Stack
StackFrame(). Constructor for class sun.tools.debug. StackFrame
StackOverflowError(). Constructor for class java.lang. StackOverflowError
Constructs a StackOverflowError with no detail message.
StackOverflowError(String). Constructor for class java.lang.StackOverflowError
Constructs a StackOverflowError with the specified detail message.
start(). Method in class java.applet.Applet

Called to start the applet.

start(). Method in class java.lang.Thread
Starts this Thread.

startProduction(ImageConsumer). Method in class java.awt.image. FilteredImageSource
Adds an ImageConsumer to the list of consumers interested in data for this image, and immediately starts delivery of the image data through the ImageConsumer interface.

startProduction(ImageConsumer). Method in interface java.awt.image.ImageProducer
This method both registers the given ImageConsumer object as a consumer and starts an immediate reconstruction of the image data which will then be delivered to this consumer and any other consumer which may have already been registered with the producer.

startProduction(ImageConsumer). Method in class java.awt.image. MemoryImageSource
Adds an ImageConsumer to the list of consumers interested in data for this image, and immediately start delivery of the image data through the ImageConsumer interface.

startsWith(String). Method in class java.lang.String
Determines whether this String starts with some prefix.

startsWith(String, int). Method in class java.lang.String
Determines whether this String starts with some prefix.

STATICIMAGEDONE. Static variable in interface java.awt.image.ImageConsumer
The image is complete and there are no more pixels or frames to be delivered.

status(). Method in class java.awt.image.PixelGrabber
Return the status of the pixels.

statusAll(boolean). Method in class java.awt.MediaTracker
Returns the boolean OR of the status of all of the media being tracked.

statusID(int, boolean). Method in class java.awt.MediaTracker
Returns the boolean OR of the status of all of the media with a given ID.

step(boolean). Method in class sun.tools.debug.RemoteThread
Continue execution of this thread to the next instruction or line.

stop(). Method in class java.applet.Applet
Called to stop the applet.

stop(). Method in interface java.applet.AudioClip
Stops playing the clip.

stop(). Method in class sun.tools.debug.RemoteThread
Stop the remote thread.

stop(). Method in class sun.tools.debug.RemoteThreadGroup
Stop the remote threadgroup.

stop(). Method in class java.lang.Thread
Stops a Thread by tossing an object.

stop(). Method in class java.lang.ThreadGroup
Stops all the Threads in this Thread group and all of its sub groups.

stop(Throwable). Method in class java.lang.Thread
Stops a Thread by tossing an object.

StreamTokenizer(InputStream). Constructor for class java.io.StreamTokenizer
Creates a stream tokenizer that parses the specified input stream.

String(). Constructor for class java.lang.String
Constructs a new empty String.

String(byte[], int). Constructor for class java.lang.String
Constructs a new String whose value is the specified array of bytes.

String(byte[], int, int, int). Constructor for class java.lang.String
Constructs a new String whose initial value is the specified sub array of bytes.

String(char[]). Constructor for class java.lang.String
Constructs a new String whose initial value is the specified array of characters.

String(char[], int, int). Constructor for class java.lang.String
Constructs a new String whose initial value is the specified sub array of characters.

String(String). Constructor for class java.lang.String
Constructs a new String that is a copy of the specified String.

String(StringBuffer). Constructor for class java.lang.String

Construct a new string whose value is the current contents of the given string buffer
StringBuffer(). Constructor for class java.lang.StringBuffer
Constructs an empty String buffer.
StringBuffer(int). Constructor for class java.lang.StringBuffer
Constructs an empty String buffer with the specified initial length.
StringBuffer(String). Constructor for class java.lang.StringBuffer
Constructs a String buffer with the specified initial value.
StringBufferInputStream(String). Constructor for class java.io. StringBufferInputStream
Creates an StringBufferInputStream from the specified array of bytes.
StringIndexOutOfBoundsException(). Constructor for class java.lang. StringIndexOutOfBoundsException
Constructs a StringIndexOutOfBoundsException with no detail message.
StringIndexOutOfBoundsException(int). Constructor for class java.lang. StringIndexOutOfBoundsException
Constructs a StringIndexOutOfBoundsException initialized with the specified index.
StringIndexOutOfBoundsException(String). Constructor for class java.lang. StringIndexOutOfBoundsException
Constructs a StringIndexOutOfBoundsException with the specified detail message.
StringTokenizer(String). Constructor for class java.util.StringTokenizer
Constructs a StringTokenizer on the specified String, using the default delimiter set (which is " \t\n\r").
StringTokenizer(String, String). Constructor for class java.util. StringTokenizer
Constructs a StringTokenizer on the specified String, using the specified delimiter set.
StringTokenizer(String, String, boolean). Constructor for class java.util. StringTokenizer
Constructs a StringTokenizer on the specified String, using the specified delimiter set.
stringWidth(String). Method in class java.awt.FontMetrics
Returns the width of the specified String in this Font.
style. Variable in class java.awt.Font
The style of the font.
substituteColorModel(ColorModel, ColorModel). Method in class java.awt.image. RGBImageFilter
Registers two ColorModel objects for substitution.
substring(int). Method in class java.lang.String
Returns the substring of this String.
substring(int, int). Method in class java.lang.String
Returns the substring of a String.
suspend(). Method in class sun.tools.debug.RemoteThread
Suspend execution of this thread.
suspend(). Method in class java.lang.Thread
Suspends this Thread's execution.
suspend(). Method in class java.lang.ThreadGroup
Suspends all the Threads in this Thread group and all of its sub groups.
sval. Variable in class java.io.StreamTokenizer
The Stream value.
SW_RESIZE_CURSOR. Static variable in class java.awt.Frame
sync(). Method in class java.awt.Toolkit
Syncs the graphics state; useful when doing animation.

T

tan(double).Static method in class java.lang.Math
Returns the trigonometric tangent of an angle.
target. Variable in class java.awt.Event
The target component.
TEXT_CURSOR. Static variable in class java.awt.Frame

TextArea(). Constructor for class java.awt.TextArea
Constructs a new TextArea.
TextArea(int, int). Constructor for class java.awt.TextArea
Constructs a new TextArea with the specified number of rows and columns.
TextArea(String). Constructor for class java.awt.TextArea
Constructs a new TextArea with the specified text displayed.
TextArea(String, int, int). Constructor for class java.awt.TextArea
Constructs a new TextArea with the specified text and number of rows and columns.
TextField(). Constructor for class java.awt.TextField
Constructs a new TextField.
TextField(int). Constructor for class java.awt.TextField
Constructs a new TextField initialized with the specified columns.
TextField(String). Constructor for class java.awt.TextField
Constructs a new TextField initialized with the specified text.
TextField(String, int). Constructor for class java.awt.TextField
Constructs a new TextField initialized with the specified text and columns.
Thread(). Constructor for class java.lang.Thread
Constructs a new Thread.
Thread(Runnable). Constructor for class java.lang.Thread
Constructs a new Thread which applies the run() method of the specified target.
Thread(Runnable, String). Constructor for class java.lang.Thread
Constructs a new Thread with the specified name and applies the run() method of the specified target.
Thread(String). Constructor for class java.lang.Thread
Constructs a new Thread with the specified name.
Thread(ThreadGroup, Runnable). Constructor for class java.lang.Thread
Constructs a new Thread in the specified Thread group that applies the run() method of the specified target.
Thread(ThreadGroup, Runnable, String). Constructor for class java.lang.Thread
Constructs a new Thread in the specified Thread group with the specified name and applies the run() method of the specified target.
Thread(ThreadGroup, String). Constructor for class java.lang.Thread
Constructs a new Thread in the specified Thread group with the specified name.
ThreadDeath(). Constructor for class java.lang.ThreadDeath
threadDeathEvent(RemoteThread). Method in interface sun.tools.debug.DebuggerCallback
A thread has died.
ThreadGroup(String). Constructor for class java.lang.ThreadGroup
Creates a new ThreadGroup.
ThreadGroup(ThreadGroup, String). Constructor for class java.lang.ThreadGroup
Creates a new ThreadGroup with a specified name in the specified Thread group.
Throwable(). Constructor for class java.lang.Throwable
Constructs a new Throwable with no detail message.
Throwable(String). Constructor for class java.lang.Throwable
Constructs a new Throwable with the specified detail message.
toBack(). Method in class java.awt.Window
Sends the frame to the back of the Window.
toBack(). Method in interface java.awt.peer.WindowPeer
toByteArray(). Method in class java.io. ByteArrayOutputStream
Returns a copy of the input data.
toCharArray(). Method in class java.lang.String
Converts this String to a character array.
toExternalForm(). Method in class java.net.URL
Reverses the parsing of the URL.
toExternalForm(URL). Method in class java.net.URLStreamHandler
Reverses the parsing of the URL.
toFront(). Method in class java.awt.Window

Brings the frame to the front of the Window.
toFront(). Method in interface java.awt.peer.WindowPeer
toGMTString(). Method in class java.util.Date
Converts a date to a String, using the Internet GMT conventions.
toHex(int). Static method in class sun.tools.debug.RemoteValue
Convert an int to a hexadecimal string.
toLocaleString(). Method in class java.util.Date
Converts a date to a String, using the locale conventions.
toLowerCase(). Method in class java.lang.String
Converts all of the characters in this String to lower case.
toLowerCase(char). Static method in class java.lang.Character
Returns the lower case character value of the specified ISO-LATIN-1 character.
Toolkit(). Constructor for class java.awt.Toolkit
top. Variable in class java.awt.Insets
The inset from the top.
TOPDOWNLEFTRIGHT. Static variable in interface java.awt.image.ImageConsumer
The pixels will be delivered in top-down, left-to-right order.
toString(). Method in class java.util.BitSet
Converts the BitSet to a String.
toString(). Method in class java.lang.Boolean
Returns a new String object representing this Boolean's value.
toString(). Method in class java.awt.BorderLayout
Returns the String representation of this BorderLayout's values.
toString(). Method in class java.io.ByteArrayOutputStream
Converts input data to a string.
toString(). Method in class java.awt.CardLayout
Returns the String representation of this CardLayout's values.
toString(). Method in class java.lang.Character
Returns a String object representing this character's value.
toString(). Method in class java.awt.CheckboxGroup
Returns the String representation of this CheckboxGroup's values.
toString(). Method in class java.lang.Class
Returns the name of this class or interface.
toString(). Method in class java.awt.Color
Returns the String representation of this Color's values.
toString(). Method in class java.awt.Component
Returns the String representation of this Component's values.
toString(). Method in class java.util.Date
Converts a date to a String, using the UNIX ctime conventions.
toString(). Method in class java.awt.Dimension
Returns the String representation of this Dimension's values.
toString(). Method in class java.lang.Double
Returns a String representation of this Double object.
toString(). Method in class java.awt.Event
Returns the String representation of this Event's values.
toString(). Method in class java.io.File
Returns a String object representing this file's path.
toString(). Method in class java.lang.Float
Returns a String representation of this Float object.
toString(). Method in class java.awt.FlowLayout
Returns the String representation of this FlowLayout's values.
toString(). Method in class java.awt.Font
Converts this object to a String representation.
toString(). Method in class java.awt.FontMetrics
Returns the String representation of this FontMetric's values.
toString(). Method in class java.awt.Graphics

Returns a String object representing this Graphic's value.
toString(). Method in class java.awt.GridBagLayout
Returns the String representation of this GridLayout's values.
toString(). Method in class java.awt.GridLayout
Returns the String representation of this GridLayout's values.
toString(). Method in class java.util.Hashtable
Converts to a rather lengthy String.
toString(). Method in class java.net.InetAddress
Converts the InetAddress to a String.
toString(). Method in class java.awt.Insets
Returns a String object representing this Inset's values.
toString(). Method in class java.lang.Integer
Returns a String object representing this Integer's value.
toString(). Method in class java.lang.Long
Returns a String object representing this Long's value.
toString(). Method in class java.awt.MenuComponent
Returns the String representation of this MenuComponent's values.
toString(). Method in class java.lang.Object
Returns a String that represents the value of this Object.
toString(). Method in class java.awt.Point
Returns the String representation of this Point's coordinates.
toString(). Method in class java.awt.Rectangle
Returns the String representation of this Rectangle's values.
toString(). Method in class sun.tools.debug.RemoteArray
Return a string version of the array.
toString(). Method in class sun.tools.debug.RemoteBoolean
Return the boolean's value as a string.
toString(). Method in class sun.tools.debug.RemoteByte
Return the byte's value as a string.
toString(). Method in class sun.tools.debug.RemoteChar
Return the char's value as a string.
toString(). Method in class sun.tools.debug.RemoteClass
Return a (somewhat verbose) description.
toString(). Method in class sun.tools.debug.RemoteDouble
Return the double's value as a string.
toString(). Method in class sun.tools.debug.RemoteField
Returns a String that represents the value of this Object.
toString(). Method in class sun.tools.debug.RemoteFloat
Return the float's value as a string.
toString(). Method in class sun.tools.debug.RemoteInt
Return the int's value as a string.
toString(). Method in class sun.tools.debug.RemoteLong
Return the long's value as a string.
toString(). Method in class sun.tools.debug.RemoteObject
Return object as a string.
toString(). Method in class sun.tools.debug.RemoteShort
Return the short's value as a string.
toString(). Method in class sun.tools.debug.RemoteString
Return the string value, or "null"
toString(). Method in class java.net.ServerSocket
Returns the implementation address and implementation port of this ServerSocket as a String.
toString(). Method in class java.net.Socket
Converts the Socket to a String.
toString(). Method in class java.net.SocketImpl
Returns the address and port of this Socket as a String.
toString(). Method in class sun.tools.debug.StackFrame

Returns a String that represents the value of this Object.
toString(). Method in class java.io.StreamTokenizer
Returns the String representation of the stream token.
toString(). Method in class java.lang.String
Converts this String to a String.
toString(). Method in class java.lang.StringBuffer
Converts to a String representing the data in the buffer.
toString(). Method in class java.lang.Thread
Returns a String representation of the Thread, including the thread's name, priority and thread group.
toString(). Method in class java.lang.ThreadGroup
Returns a String representation of the Thread group.
toString(). Method in class java.lang.Throwable
Returns a short description of the Throwable.
toString(). Method in class java.net.URL
Converts to a human-readable form.
toString(). Method in class java.net.URLConnection
Returns the String representation of the URL connection.
toString(). Method in class java.util.Vector
Converts the vector to a string.
toString(double). Static method in class java.lang.Double
Returns a String representation for the specified double value.
toString(float). Static method in class java.lang.Float
Returns a String representation for the specified float value.
toString(int). Method in class java.io.ByteArrayOutputStream
Converts input data to a string.
toString(int). Static method in class java.lang.Integer
Returns a new String object representing the specified integer.
toString(int, int). Static method in class java.lang.Integer
Returns a new String object representing the specified integer in the specified radix.
toString(long). Static method in class java.lang.Long
Returns a new String object representing the specified integer.
toString(long, int). Static method in class java.lang.Long
Returns a new String object representing the specified long in the specified radix.
totalMemory(). Method in class sun.tools.debug.RemoteDebugger
Report the total memory usage of the Java interpreter being debugged.
totalMemory(). Method in class java.lang.Runtime
Returns the total number of bytes in system memory.
toUpperCase(). Method in class java.lang.String
Converts all of the characters in this String to upper case.
toUpperCase(char). Static method in class java.lang.Character
Returns the upper case character value of the specified ISO-LATIN-1 character.
trace(boolean). Method in class sun.tools.debug.RemoteDebugger
Turn on/off method call tracing.
traceInstructions(boolean). Method in class java.lang.Runtime
Enables/Disables tracing of instructions.
traceMethodCalls(boolean). Method in class java.lang.Runtime
Enables/Disables tracing of method calls.
translate(int, int). Method in class java.awt.Event
Translates an event relative to the given component.
translate(int, int). Method in class java.awt.Graphics
Translates the specified parameters into the origin of the graphics context.
translate(int, int). Method in class java.awt.Point
Translates the point.
translate(int, int). Method in class java.awt.Rectangle
Translates the rectangle.
trim(). Method in class java.lang.String

U

V

Validates this Container and all of the components contained within it.

valueOf(boolean). Static method in class java.lang.String
Returns a String object that represents the state of the specified boolean.

valueOf(char). Static method in class java.lang.String
Returns a String object that contains a single character

valueOf(char[]). Static method in class java.lang.String
Returns a String that is equivalent to the specified character array.

valueOf(char[], int, int). Static method in class java.lang.String
Returns a String that is equivalent to the specified character array.

valueOf(double). Static method in class java.lang.String
Returns a String object that represents the value of the specified double.

valueOf(float). Static method in class java.lang.String
Returns a String object that represents the value of the specified float.

valueOf(int). Static method in class java.lang.String
Returns a String object that represents the value of the specified integer.

valueOf(long). Static method in class java.lang.String
Returns a String object that represents the value of the specified long.

valueOf(Object). Static method in class java.lang.String
Returns a String that represents the String value of the object.

valueOf(String). Static method in class java.lang.Boolean
Returns the boolean value represented by the specified String.

valueOf(String). Static method in class java.lang.Double
Returns a new Double value initialized to the value represented by the specified String.

valueOf(String). Static method in class java.lang.Float
Returns the floating point value represented by the specified String.

valueOf(String). Static method in class java.lang.Integer
Assuming the specified String represents an integer, returns a new Integer object initialized to that value.

valueOf(String). Static method in class java.lang.Long
Assuming the specified String represents a long, returns a new Long object initialized to that value.

valueOf(String, int). Static method in class java.lang.Integer
Assuming the specified String represents an integer, returns a new Integer object initialized to that value.

valueOf(String, int). Static method in class java.lang.Long
Assuming the specified String represents a long, returns a new Long object initialized to that value.

Vector(). Constructor for class java.util.Vector
Constructs an empty vector.

Vector(int). Constructor for class java.util.Vector
Constructs an empty vector with the specified storage capacity.

Vector(int, int). Constructor for class java.util.Vector
Constructs an empty vector with the specified storage capacity and the specified capacityIncrement.

VerifyError(). Constructor for class java.lang.VerifyError
Constructor.

VerifyError(String). Constructor for class java.lang.VerifyError
Constructor with a detail message.

VERTICAL. Static variable in class java.awt.GridBagConstraints

VERTICAL. Static variable in class java.awt.Scrollbar
The vertical Scrollbar variable.

VirtualMachineError(). Constructor for class java.lang.VirtualMachineError
Constructs a VirtualMachineError with no detail message.

VirtualMachineError(String). Constructor for class java.lang. VirtualMachineError
Constructs a VirtualMachineError with the specified detail message.

W

write(byte[]). Method in class java.io.FilterOutputStream
Writes an array of bytes.
write(byte[]). Method in class java.io.OutputStream
Writes an array of bytes.
write(byte[]). Method in class java.io.RandomAccessFile
Writes an array of bytes.
write(byte[], int, int). Method in class java.io.BufferedOutputStream
Writes a subarray of bytes.
write(byte[], int, int). Method in class java.io.ByteArrayOutputStream
Writes bytes to the buffer.
write(byte[], int, int). Method in interface java.io.DataOutput
Writes a subarray of bytes.
write(byte[], int, int). Method in class java.io.DataOutputStream
Writes a sub array of bytes.
write(byte[], int, int). Method in class java.io.FileOutputStream
Writes a sub array of bytes.
write(byte[], int, int). Method in class java.io.FilterOutputStream
Writes a subarray of bytes.
write(byte[], int, int). Method in class java.io.OutputStream
Writes a sub array of bytes.
write(byte[], int, int). Method in class java.io.PipedOutputStream
Writes a sub array of bytes.
write(byte[], int, int). Method in class java.io.PrintStream
Writes a sub array of bytes.
write(byte[], int, int). Method in class java.io.RandomAccessFile
Wrotes a sub array of bytes.
write(int). Method in class java.io.BufferedOutputStream
Writes a byte.
write(int). Method in class java.io.ByteArrayOutputStream
Writes a byte to the buffer.
write(int). Method in interface java.io.DataOutput
Writes a byte.
write(int). Method in class java.io.DataOutputStream
Writes a byte.
write(int). Method in class java.io.FileOutputStream
Writes a byte of data.
write(int). Method in class java.io.FilterOutputStream
Writes a byte.
write(int). Method in class java.io.OutputStream
Writes a byte.
write(int). Method in class java.io.PipedOutputStream
Write a byte.
write(int). Method in class java.io.PrintStream
Writes a byte.
write(int). Method in class java.io.RandomAccessFile
Writes a byte of data.
writeBoolean(boolean). Method in interface java.io.DataOutput
Writes a boolean.
writeBoolean(boolean). Method in class java.io.DataOutputStream
Writes a boolean.
writeBoolean(boolean). Method in class java.io.RandomAccessFile
Writes a boolean.
writeByte(int). Method in interface java.io.DataOutput
Writes an 8 bit byte.
writeByte(int). Method in class java.io.DataOutputStream
Writes an 8 bit byte.

writeByte(int). Method in class java.io.RandomAccessFile
Writes a byte.
writeBytes(String). Method in interface java.io.DataOutput
Writes a String as a sequence of bytes.
writeBytes(String). Method in class java.io.DataOutputStream
Writes a String as a sequence of bytes.
writeBytes(String). Method in class java.io.RandomAccessFile
Writes a String as a sequence of bytes.
writeChar(int). Method in interface java.io.DataOutput
Writes a 16 bit char.
writeChar(int). Method in class java.io.DataOutputStream
Writes a 16 bit char.
writeChar(int). Method in class java.io.RandomAccessFile
Writes a character.
writeChars(String). Method in interface java.io.DataOutput
Writes a String as a sequence of chars.
writeChars(String). Method in class java.io.DataOutputStream
Writes a String as a sequence of chars.
writeChars(String). Method in class java.io.RandomAccessFile
Writes a String as a sequence of chars.
writeDouble(double). Method in interface java.io.DataOutput
Writes a 64 bit double.
writeDouble(double). Method in class java.io.DataOutputStream
Writes a 64 bit double.
writeDouble(double). Method in class java.io.RandomAccessFile
writeFloat(float). Method in interface java.io. DataOutput
Writes a 32 bit float.
writeFloat(float). Method in class java.io.DataOutputStream
Writes a 32 bit float.
writeFloat(float). Method in class java.io.RandomAccessFile
writeInt(int). Method in interface java.io.DataOutput
Writes a 32 bit int.
writeInt(int). Method in class java.io.DataOutputStream
Writes a 32 bit int.
writeInt(int). Method in class java.io.RandomAccessFile
Writes an integer.
writeLong(long). Method in interface java.io.DataOutput
Writes a 64 bit long.
writeLong(long). Method in class java.io.DataOutputStream
Writes a 64 bit long.
writeLong(long). Method in class java.io.RandomAccessFile
Writes a long.
writeShort(int). Method in interface java.io.DataOutput
Writes a 16 bit short.
writeShort(int). Method in class java.io.DataOutputStream
Writes a 16 bit short.
writeShort(int). Method in class java.io.RandomAccessFile
Writes a short.
writeTo(OutputStream). Method in class java.io.ByteArrayOutputStream
Writes the contents of the buffer to another stream.
writeUTF(String). Method in interface java.io.DataOutput
Writes a String in UTF format.
writeUTF(String). Method in class java.io.DataOutputStream
Writes a String in UTF format.
writeUTF(String). Method in class java.io.RandomAccessFile
Writes a String in UTF format.

written. Variable in class java.io.DataOutputStream
The number of bytes written so far.

X

x.Variable in class java.awt.Event
The x coordinate of the event.
x. Variable in class java.awt.Point
The x coordinate.
x. Variable in class java.awt.Rectangle
The x coordinate of the rectangle.
xor(BitSet). Method in class java.util.BitSet
Logically XORs this bit set with the specified set of bits.
xpoints. Variable in class java.awt.Polygon
The array of x coordinates.

Y

y.Variable in class java.awt.Event
The y coordinate of the event.
y. Variable in class java.awt.Point
The y coordinate.
y. Variable in class java.awt.Rectangle
The y coordinate of the rectangle.
yellow. Static variable in class java.awt.Color
The color yellow.
yield(). Static method in class java.lang.Thread
Causes the currently executing Thread object to yield.
ypoints. Variable in class java.awt.Polygon
The array of y coordinates.

Class java.applet.Applet

Class java.applet.Applet

```
java.lang.Object
    |
    +----java.awt.Component
            |
            +----java.awt.Container
                    |
                    +----java.awt.Panel
                            |
                            +----java.applet.Applet
```

---

public class **Applet**
extends Panel

Base applet class.

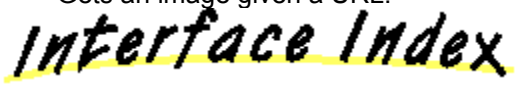
Constructor Index

- **Applet**()

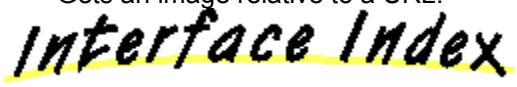
Method Index

- **destroy**()
  Cleans up whatever resources are being held.
- **getAppletContext**()
  Gets a handle to the applet context.
- **getAppletInfo**()
  Returns a string containing information about the author, version and copyright of the applet.
- **getAudioClip**(URL)
  Gets an audio clip.
- **getAudioClip**(URL, String)
  Gets an audio clip.
- **getCodeBase**()
  Gets the base URL.
- **getDocumentBase**()
  Gets the document URL.

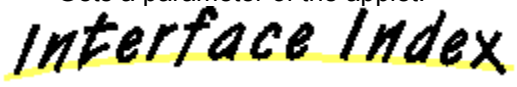
- **getImage**(URL)
  Gets an image given a URL.



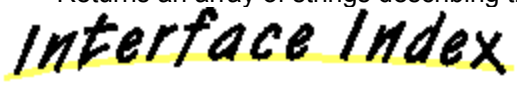
getImage(URL, String)
Gets an image relative to a URL.



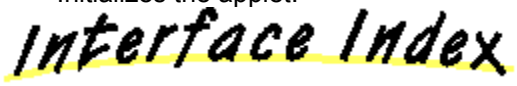
getParameter(String)
Gets a parameter of the applet.



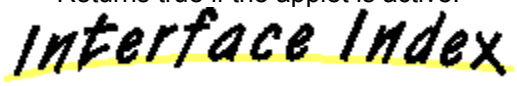
getParameterInfo()
Returns an array of strings describing the parameters that are understood by this applet.



init()
Initializes the applet.



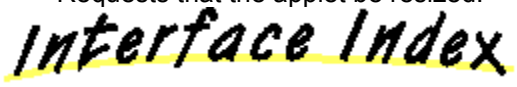
isActive()
Returns true if the applet is active.



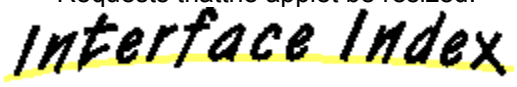
play(URL)
Plays an audio clip.

play(URL, String)
Plays an audio clip.

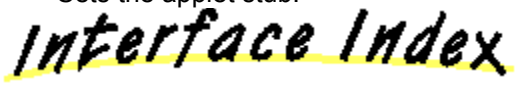
resize(int, int)
Requests that the applet be resized.



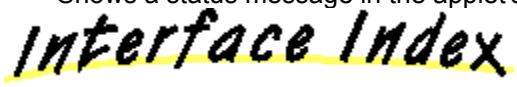
resize(Dimension)
Requests thatthe applet be resized.



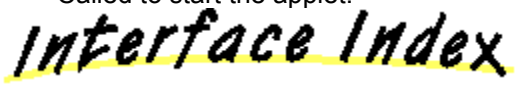
setStub(AppletStub)
Sets the applet stub.



showStatus(String)
Shows a status message in the applet's context.



start()
Called to start the applet.



stop()
Called to stop the applet.

Interface Index Applet

```
public Applet()
```

Methods

Interface Index setStub

```
public final void setStub(AppletStub stub)
```

Sets the applet stub. This is done by automatically by the system.

Interface Index isActive

```
public boolean isActive()
```

Returns true if the applet is active. An applet is marked active just before the start method is called.

See Also:

start

Interface Index getDocumentBase

```
public URL getDocumentBase()
```

Gets the document URL. This is the URL of the document in which the applet is embedded.

See Also:

getCodeBase

Interface Index getCodeBase

```
public URL getCodeBase()
```

Gets the base URL. This is the URL of the applet itself.

See Also:

getDocumentBase

getParameter

```
public String getParameter(String name)
```

Gets a parameter of the applet.

getAppletContext

```
public AppletContext getAppletContext()
```

Gets a handle to the applet context. The applet context lets an applet control the applet's environment which is usually the browser or the applet viewer.

resize

```
public void resize(int width,
                   int height)
```

Requests that the applet be resized.

Overrides:

resize in class Component

resize

```
public void resize(Dimension d)
```

Requests thatthe applet be resized.

Overrides:

resize in class Component

showStatus

```
public void showStatus(String msg)
```

Shows a status message in the applet's context.

 getImage

```
public Image getImage(URL url)
```

Gets an image given a URL. Note that this method always returns an image object immediatly, even if the image does not exist. The actual image data is loaded when it is first needed.

 getImage

```
public Image getImage(URL url,
                      String name)
```

Gets an image relative to a URL. This methods returns immediately, even if the image does not exist. The actual image data is loaded when it is first needed.

See Also:

getImage

 getAudioClip

```
public AudioClip getAudioClip(URL url)
```

Gets an audio clip.

 getAudioClip

```
public AudioClip getAudioClip(URL url,
                              String name)
```

Gets an audio clip.

See Also:

getAudioClip

 getAppletInfo

```
public String getAppletInfo()
```

Returns a string containing information about the author, version and copyright of the applet.

 getParameterInfo

```
public String[][] getParameterInfo()
```

Returns an array of strings describing the parameters that are understood by this applet. The array consists of sets of three strings: name/type/description. For example:

```
String pinfo[][] = {
  {"fps",    "1-10",    "frames per second"},
  {"repeat", "boolean", "repeat image loop"},
  {"imgs",   "url",     "directory in which the images live"}
};
```

 play

```
public void play(URL url)
```

Plays an audio clip. Nothing happens if the audio clip could not be found.

 play

```
public void play(URL url,
                 String name)
```

Plays an audio clip. Nothing happens if the audio clip could not be found.

 init

```
public void init()
```

Initializes the applet. You never need to call this directly, it is called automatically by the system once the applet is created.

See Also:

start, stop, destroy

 start

```
public void start()
```

Called to start the applet. You never need to call this method directly, it is called when the applet's document is visited.

See Also:

init, stop, destroy

stop

```
public void stop()
```

Called to stop the applet. It is called when the applet's document is no longer on the screen. It is guaranteed to be called before destroy() is called. You never need to call this method directly.

See Also:

init, start, destroy

destroy

```
public void destroy()
```

Cleans up whatever resources are being held. If the applet is active it is stopped stopped.

See Also:

init, start, stop

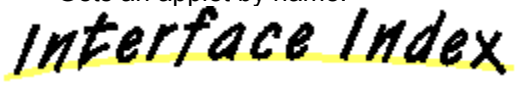
Interface java.applet.AppletContext

Interface java.applet.AppletContext

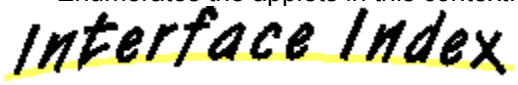
public interface **AppletContext**
extends Object

This interface corresponds to an applet's environment. It can be used by an applet to obtain information from the applet's environment, which is usually the browser or the applet viewer.

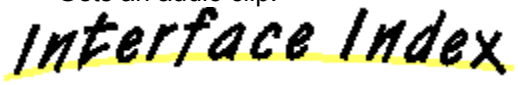
getApplet(String)
Gets an applet by name.



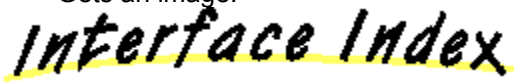
getApplets()
Enumerates the applets in this context.



getAudioClip(URL)
Gets an audio clip.

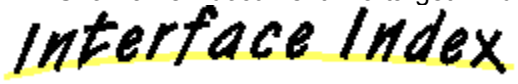


getImage(URL)
Gets an image.



showDocument(URL)
Shows a new document.

showDocument(URL, String)
Show a new document in a target window or frame.



showStatus(String)
Show a status string.

getAudioClip

```
public abstract AudioClip getAudioClip(URL url)
```

Gets an audio clip.

getImage

```
public abstract Image getImage(URL url)
```

Gets an image. This usually involves downloading it over the net. However, the environment may decide to cache images. This method takes an array of URLs, each of which will be tried until the image is found.

getApplet

```
public abstract Applet getApplet(String name)
```

Gets an applet by name.
Returns:
null if the applet does not exist.

getApplets

```
public abstract Enumeration getApplets()
```

Enumerates the applets in this context. Only applets that are accessible will be returned. This list always includes the applet itself.

showDocument

```
public abstract void showDocument(URL url)
```

Shows a new document. This may be ignored by the applet context.

showDocument

```
public abstract void showDocument(URL url,
                                  String target)
```

Show a new document in a target window or frame. This may be ignored by the applet context. This method accepts the target strings: _self show in current frame _parent show in parent frame

_top show in top-most frame _blank show in new unnamed top-level window show in new top-level window named


```
public abstract void showStatus(String status)
```

Show a status string.

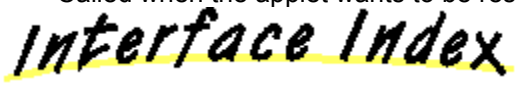
Interface java.applet.AppletStub

Interface java.applet.AppletStub

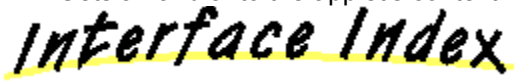
public interface **AppletStub**
extends Object

This interface is used to implement an applet viewer. It is not normally used by applet programmers.

appletResize(int, int)
Called when the applet wants to be resized.



getAppletContext()
Gets a handler to the applet's context.



getCodeBase()
Gets the base URL.

getDocumentBase()
Gets the document URL.

getParameter(String)
Gets a parameter of the applet.

isActive()
Returns true if the applet is active.

isActive

```
public abstract boolean isActive()
```

Returns true if the applet is active.

getDocumentBase

```
public abstract URL getDocumentBase()
```

Gets the document URL.

getCodeBase

```
public abstract URL getCodeBase()
```

Gets the base URL.

getParameter

```
public abstract String getParameter(String name)
```

Gets a parameter of the applet.

getAppletContext

```
public abstract AppletContext getAppletContext()
```

Gets a handler to the applet's context.

appletResize

```
public abstract void appletResize(int width,
                                  int height)
```

Called when the applet wants to be resized.

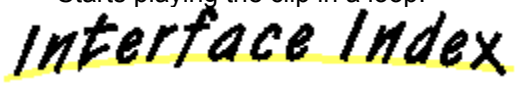
Interface java.applet.AudioClip

Interface java.applet.AudioClip

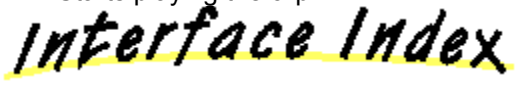
public interface **AudioClip**
extends Object

A very high level abstraction of audio.

loop()
Starts playing the clip in a loop.



play()
Starts playing the clip.



stop()
Stops playing the clip.

play

```
public abstract void play()
```

Starts playing the clip. Each time this method is called, the clip is restarted from the beginning.

loop

```
public abstract void loop()
```

Starts playing the clip in a loop.

stop

```
public abstract void stop()
```

Stops playing the clip.

Class java.awt.AWTError

Class java.awt.AWTError

```
java.lang.Object
    |
    +----java.lang.Throwable
            |
            +----java.lang.Error
                    |
                    +----java.awt.AWTError
```

---

public class **AWTError**
extends Error

An AWT Error.

Interface Index

Interface Index

AWTError(String)

Constructors

Interface Index

AWTError

```
public AWTError(String msg)
```

Class java.awt.AWTException

Class java.awt.AWTException

```
java.lang.Object
    |
    +----java.lang.Throwable
            |
            +----java.lang.Exception
                    |
                    +----java.awt.AWTException
```

---

public class **AWTException**
extends Exception

Signals that an Absract Window Toolkit exception has occurred.

Interface Index

Interface Index

AWTException(String)
Constructs an AWTException with the specified detail message.

Constructors

Interface Index

AWTException

```
public AWTException(String msg)
```

Constructs an AWTException with the specified detail message. A detail message is a String that describes this particular exception.
Parameters:
msg - the detail message

Class java.awt.BorderLayout

Class java.awt.BorderLayout

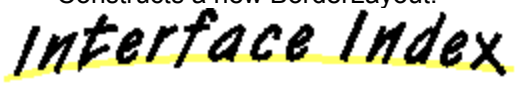
```
java.lang.Object
    |
    +----java.awt.BorderLayout
```

---

public class **BorderLayout**
extends Object
implements LayoutManager

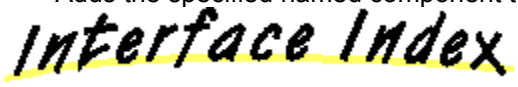
A TNT style border bag layout. It will layout a container using members named "North", "South", "East", "West" and "Center". The "North", "South", "East" and "West" components get layed out according to their preferred sizes and the constraints of the container's size. The "Center" component will get any space left over.

BorderLayout()
Constructs a new BorderLayout.

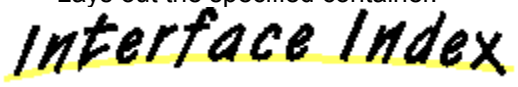


BorderLayout(int, int)
Constructs a BorderLayout with the specified gaps.

addLayoutComponent(String, Component)
Adds the specified named component to the layout.



layoutContainer(Container)
Lays out the specified container.



minimumLayoutSize(Container)
Returns the minimum dimensions needed to layout the components contained in the specified target

container.

Interface Index **preferredLayoutSize**(Container)

Returns the preferred dimensions for this layout given the components in the specified target container.

Interface Index **removeLayoutComponent**(Component)

Removes the specified component from the layout.

Interface Index **toString**()

Returns the String representation of this BorderLayout's values.

Constructors

Interface Index BorderLayout

```
public BorderLayout()
```

Constructs a new BorderLayout.

Interface Index BorderLayout

```
public BorderLayout(int hgap,
                    int vgap)
```

Constructs a BorderLayout with the specified gaps.

Parameters:

hgap - the horizontal gap
vgap - the vertical gap

Methods

Interface Index addLayoutComponent

```
public void addLayoutComponent(String name,
                               Component comp)
```

Adds the specified named component to the layout.

Parameters:

name - the String name
comp - the component to be added

removeLayoutComponent

```
public void removeLayoutComponent(Component comp)
```

Removes the specified component from the layout.
Parameters:
comp - the component to be removed

minimumLayoutSize

```
public Dimension minimumLayoutSize(Container target)
```

Returns the minimum dimensions needed to layout the components contained in the specified target container.
Parameters:
target - the Container on which to do the layout
See Also:
Container, preferredLayoutSize

preferredLayoutSize

```
public Dimension preferredLayoutSize(Container target)
```

Returns the preferred dimensions for this layout given the components in the specified target container.
Parameters:
target - the component which needs to be laid out
See Also:
Container, minimumLayoutSize

layoutContainer

```
public void layoutContainer(Container target)
```

Lays out the specified container. This method will actually reshape the components in the specified target container in order to satisfy the constraints of the BorderLayout object.

Parameters:

target - the component being laid out

See Also:

Container

toString

```
public String toString()
```

Returns the String representation of this BorderLayout's values.

Overrides:

toString in class Object

Class java.awt.Button

Class java.awt.Button

```
java.lang.Object
    |
    +----java.awt.Component
            |
            +----java.awt.Button
```

---

public class **Button**
extends Component

A class that produces a labeled button component.

Button()

Constructs a Button with no label.

Button(String)

Constructs a Button with a string label.

addNotify()

Creates the peer of the button.

getLabel()

Gets the label of the button.

paramString()

Returns the parameter String of this button.

Interface Index **setLabel**(String)

Sets the button with the specified label.

Constructors

Interface Index **Button**

```
public Button()
```

Constructs a Button with no label.

Interface Index **Button**

```
public Button(String label)
```

Constructs a Button with a string label.

Parameters:

label - the button label

Methods

Interface Index **addNotify**

```
public synchronized void addNotify()
```

Creates the peer of the button. This peer allows us to change the look of the button without changing its functionality.

Overrides:

addNotify in class Component

Interface Index **getLabel**

```
public String getLabel()
```

Gets the label of the button.
See Also:
setLabel

setLabel

```
public void setLabel(String label)
```

Sets the button with the specified label.
Parameters:
label - the label to set the button with
See Also:
getLabel

paramString

```
protected String paramString()
```

Returns the parameter String of this button.
Overrides:
paramString in class Component

Class java.awt.Canvas

Class java.awt.Canvas

```
java.lang.Object
    |
    +----java.awt.Component
              |
              +----java.awt.Canvas
```

---

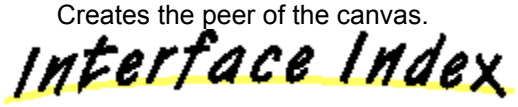
public class **Canvas**
extends Component

A Canvas component. This is a generic component which needs to be subclassed in order to add some interesting functionality.

Canvas()

addNotify()

Creates the peer of the canvas.



paint(Graphics)

Paints the canvas in the default background color.

Canvas

```
public Canvas()
```

addNotify

```
public synchronized void addNotify()
```

Creates the peer of the canvas. This peer allows you to change the user interface of the canvas without changing its functionality.

Overrides:

addNotify in class Component

paint

```
public void paint(Graphics g)
```

Paints the canvas in the default background color.

Parameters:

g - the specified Graphics window

Overrides:

paint in class Component

Class java.awt.CardLayout

Class java.awt.CardLayout

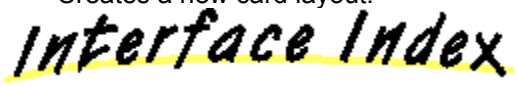
```
java.lang.Object
    |
    +----java.awt.CardLayout
```

---

public class **CardLayout**
extends Object
implements LayoutManager

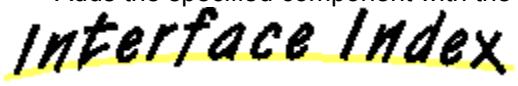
A layout manager for a container that contains several 'cards'. Only one card is visible at a time, allowing you to flip through the cards.

CardLayout()
Creates a new card layout.

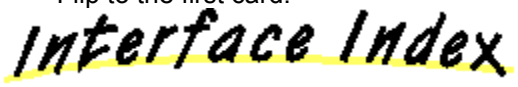


CardLayout(int, int)
Creates a card layout with the specified gaps.

addLayoutComponent(String, Component)
Adds the specified component with the specified name to the layout.



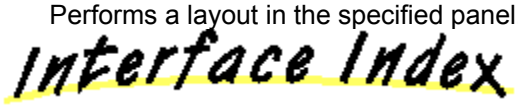
first(Container)
Flip to the first card.



last(Container)
Flips to the last card of the specified container.

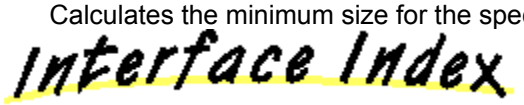
layoutContainer(Container)

Performs a layout in the specified panel.



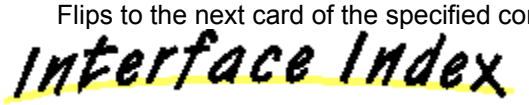
minimumLayoutSize(Container)

Calculates the minimum size for the specified panel.



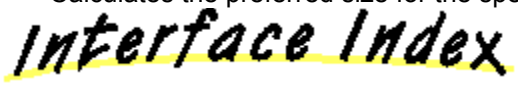
next(Container)

Flips to the next card of the specified container.



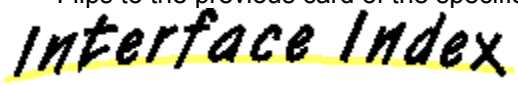
preferredLayoutSize(Container)

Calculates the preferred size for the specified panel.



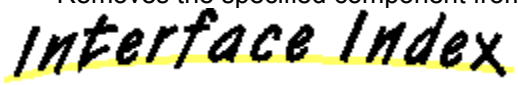
previous(Container)

Flips to the previous card of the specified container.



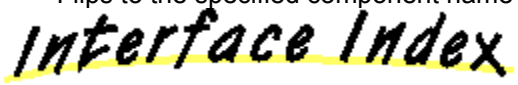
removeLayoutComponent(Component)

Removes the specified component from the layout.



show(Container, String)

Flips to the specified component name in the specified container.



toString()

Returns the String representation of this CardLayout's values.

CardLayout

```
public CardLayout()
```

Creates a new card layout.

CardLayout

```
public CardLayout(int hgap,
                  int vgap)
```

Creates a card layout with the specified gaps.

Parameters:

hgap - the horizontal gap

vgap - the vertical gap

addLayoutComponent

```
public void addLayoutComponent(String name,
                               Component comp)
```

Adds the specified component with the specified name to the layout.
Parameters:
name - the name of the component
comp - the component to be added

removeLayoutComponent

```
public void removeLayoutComponent(Component comp)
```

Removes the specified component from the layout.
Parameters:
comp - the component to be removed

preferredLayoutSize

```
public Dimension preferredLayoutSize(Container parent)
```

Calculates the preferred size for the specified panel.
Parameters:
parent - the name of the parent container
Returns:
the dimensions of this panel.
See Also:
minimumLayoutSize

minimumLayoutSize

```
public Dimension minimumLayoutSize(Container parent)
```

Calculates the minimum size for the specified panel.
Parameters:
parent - the name of the parent container
Returns:
the dimensions of this panel.
See Also:
preferredLayoutSize

layoutContainer

```
public void layoutContainer(Container parent)
```

Performs a layout in the specified panel.
Parameters:
parent - the name of the parent container

first

```
public void first(Container parent)
```

Flip to the first card.
Parameters:
parent - the name of the parent container

next

```
public void next(Container parent)
```

Flips to the next card of the specified container.
Parameters:
parent - the name of the container

previous

```
public void previous(Container parent)
```

Flips to the previous card of the specified container.
Parameters:
parent - the name of the parent container

last

```
public void last(Container parent)
```

Flips to the last card of the specified container.
Parameters:
parent - the name of the parent container

show

```
public void show(Container parent,
                 String name)
```

Flips to the specified component name in the specified container.
Parameters:
parent - the name of the parent container
name - the component name

toString

```
public String toString()
```

Returns the String representation of this CardLayout's values.
Overrides:
toString in class Object

Class java.awt.Checkbox

Class java.awt.Checkbox

```
java.lang.Object
    |
    +----java.awt.Component
             |
             +----java.awt.Checkbox
```

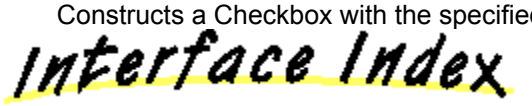
---

public class **Checkbox**
extends Component

A Checkbox object is a graphical user interface element that has a boolean state.

Checkbox()
Constructs a Checkbox with no label, no Checkbox group, and initialized to a false state.

Checkbox(String)
Constructs a Checkbox with the specified label, no Checkbox group, and initialized to a false state.



Checkbox(String, CheckboxGroup, boolean)
Constructs a Checkbox with the specified label, specified Checkbox group, and specified boolean state.

addNotify()
Creates the peer of the Checkbox.

getCheckboxGroup()
Returns the checkbox group.

Interface Index

Interface Index **Checkbox**

```
public Checkbox()
```

Constructs a Checkbox with no label, no Checkbox group, and initialized to a false state.

Interface Index **Checkbox**

```
public Checkbox(String label)
```

Constructs a Checkbox with the specified label, no Checkbox group, and initialized to a false state.

Parameters:

label - the label on the Checkbox

Interface Index **Checkbox**

```
public Checkbox(String label,
                CheckboxGroup group,
                boolean state)
```

Constructs a Checkbox with the specified label, specified Checkbox group, and specified boolean state. If the specified CheckboxGroup is not equal to null, then this Checkbox becomes a Checkbox button. If the Checkbox becomes a button, this simply means that only one Checkbox in a CheckboxGroup may be set at a time.

Parameters:

label - the label on the Checkbox
group - the CheckboxGroup this Checkbox is in
state - is the initial state of this Checkbox

addNotify

```
public synchronized void addNotify()
```

Creates the peer of the Checkbox. The peer allows you to change the look of the Checkbox without changing its functionality.

Overrides:

addNotify in class Component

getLabel

```
public String getLabel()
```

Gets the label of the button.

See Also:

setLabel

setLabel

```
public void setLabel(String label)
```

Sets the button with the specified label.

Parameters:

label - the label of the button

See Also:

getLabel

getState

```
public boolean getState()
```

Returns the boolean state of the Checkbox.

See Also:

setState

setState

```
public void setState(boolean state)
```

Sets the Checkbox to the specifed boolean state.

Parameters:

state - the boolean state

See Also:

getState

getCheckboxGroup

```
public CheckboxGroup getCheckboxGroup()
```

Returns the checkbox group.

See Also:

setCheckboxGroup

setCheckboxGroup

```
public void setCheckboxGroup(CheckboxGroup g)
```

Sets the CheckboxGroup to the specified group.

Parameters:

g - the new CheckboxGroup

See Also:

getCheckboxGroup

paramString

```
protected String paramString()
```

Returns the parameter String of this Checkbox.

Overrides:

paramString in class Component

Class java.awt.CheckboxGroup

Class java.awt.CheckboxGroup

```
java.lang.Object
    |
    +----java.awt.CheckboxGroup
```

---

public class **CheckboxGroup**
extends Object

This class is used to create a multiple-exclusion scope for a set of Checkbox buttons. For example, creating a set of Checkbox buttons with the same CheckboxGroup object means that only one of those Checkbox buttons will be allowed to be "on" at a time.

CheckboxGroup()
Creates a new CheckboxGroup.

getCurrent()
Gets the current choice.

setCurrent(Checkbox)
Sets the current choice to the specified Checkbox.

toString()
Returns the String representation of this CheckboxGroup's values.

CheckboxGroup

```
public CheckboxGroup()
```

Creates a new CheckboxGroup.

getCurrent

```
public Checkbox getCurrent()
```

Gets the current choice.

setCurrent

```
public synchronized void setCurrent(Checkbox box)
```

Sets the current choice to the specified Checkbox. If the Checkbox belongs to a different group, just return.

Parameters:

box - the current Checkbox choice

toString

```
public String toString()
```

Returns the String representation of this CheckboxGroup's values. Convert to String.

Overrides:

toString in class Object

Class java.awt.CheckboxMenuItem

Class java.awt.CheckboxMenuItem

```
java.lang.Object
    |
    +----java.awt.MenuComponent
            |
            +----java.awt.MenuItem
                    |
                    +----java.awt.CheckboxMenuItem
```

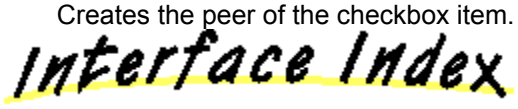
---

public class **CheckboxMenuItem**
extends MenuItem

This class produces a checkbox that represents a choice in a menu.

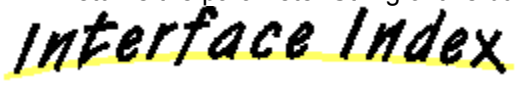
CheckboxMenuItem(String)
Creates the checkbox item with the specified label.

addNotify()
Creates the peer of the checkbox item.



getState()
Returns the state of this MenuItem.

paramString()
Returns the parameter String of this button.



setState(boolean)
Sets the state of this MenuItem if it is a Checkbox.

Interface Index

CheckboxMenuItem

```
public CheckboxMenuItem(String label)
```

Creates the checkbox item with the specified label.

Parameters:

label - the button label

Interface Index

Interface Index

addNotify

```
public synchronized void addNotify()
```

Creates the peer of the checkbox item. This peer allows us to change the look of the checkbox item without changing its functionality.

Overrides:

addNotify in class MenuItem

Interface Index

getState

```
public boolean getState()
```

Returns the state of this MenuItem. This method is only valid for a Checkbox.

Interface Index

setState

```
public void setState(boolean t)
```

Sets the state of this MenuItem if it is a Checkbox.

Parameters:

t - the specified state of the checkbox

paramString

```
public String paramString()
```

Returns the parameter String of this button.

Overrides:

paramString in class MenuItem

Class java.awt.Choice

Class java.awt.Choice

```
java.lang.Object
    |
    +----java.awt.Component
            |
            +----java.awt.Choice
```

---

public class **Choice**
extends Component

The Choice class is a pop-up menu of choices. The current choice is displayed as the title of the menu.

Choice()
Constructs a new Choice.

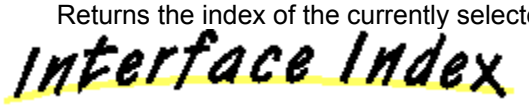
addItem(String)
Adds an item to this Choice.

addNotify()
Creates the Choice's peer.

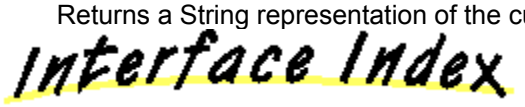
countItems()
Returns the number of items in this Choice.

getItem(int)
Returns the String at the specified index in the Choice.

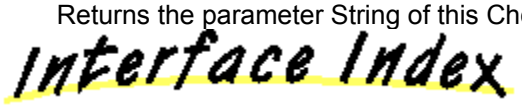
getSelectedIndex()
Returns the index of the currently selected item.



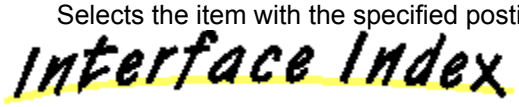
getSelectedItem()
Returns a String representation of the current choice.



paramString()
Returns the parameter String of this Choice.



select(int)
Selects the item with the specified postion.



select(String)
Selects the item with the specified String.

Choice

```
public Choice()
```

Constructs a new Choice.

addNotify

```
public synchronized void addNotify()
```

Creates the Choice's peer. This peer allows us to change the look of the Choice without changing its functionality.
Overrides:
addNotify in class Component

countItems

```
public int countItems()
```

Returns the number of items in this Choice.

See Also:

getItem

getItem

```
public String getItem(int index)
```

Returns the String at the specified index in the Choice.

Parameters:

index - the index at which to begin

See Also:

countItems

addItem

```
public synchronized void addItem(String item)
```

Adds an item to this Choice.

Parameters:

item - the item to be added

Throws: NullPointerException

If the item's value is equal to null.

getSelectedItem

```
public String getSelectedItem()
```

Returns a String representation of the current choice.

See Also:

getSelectedIndex

getSelectedIndex

```
public int getSelectedIndex()
```

Returns the index of the currently selected item.
See Also:
getSelectedItem

select

```
public synchronized void select(int pos)
```

Selects the item with the specified postion.
Parameters:
pos - the choice item position
Throws: IllegalArgumentException
If the choice item position is invalid.
See Also:
getSelectedItem, getSelectedIndex

select

```
public void select(String str)
```

Selects the item with the specified String.
Parameters:
str - the specified String
See Also:
getSelectedItem, getSelectedIndex

paramString

```
protected String paramString()
```

Returns the parameter String of this Choice.
Overrides:
paramString in class Component

Class java.awt.Color

Class java.awt.Color

```
java.lang.Object
    |
    +----java.awt.Color
```

---

public final class **Color**
extends Object

A class to encapsulate RGB Colors.

Variable Index

- **black**
  The color black.
- **blue**
  The color blue.
- **cyan**
  The color cyan.
- **darkGray**
  The color dark gray.
- **gray**
  The color gray.
- **green**
  The color green.
- **lightGray**
  The color light gray.
- **magenta**
  The color magneta.
- **orange**
  The color orange.
- **pink**
  The color pink.

Interface Index

red
The color red.

Interface Index

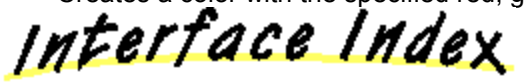
white
The color white.

yellow

The color yellow.

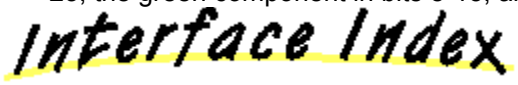
Color(int, int, int)

Creates a color with the specified red, green, and blue values in the range (0 - 255).



Color(int)

Creates a color with the specified combined RGB value consisting of the red component in bits 16-23, the green component in bits 8-15, and the blue component in bits 0-7.



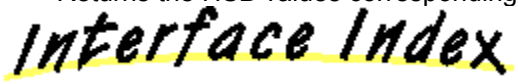
Color(float, float, float)

Creates a color with the specified red, green, and blue values in the range (0.0 - 1.0).

- **HSBtoRGB**(float, float, float)

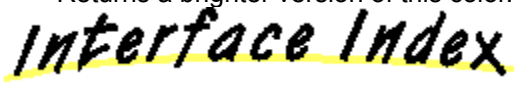
  Returns the RGB value defined by the default RGB ColorModel, of the color corresponding to the given HSB color components.
- **RGBtoHSB**(int, int, int, float[])

  Returns the HSB values corresponding to the color defined by the red, green, and blue components.



brighter()

Returns a brighter version of this color.

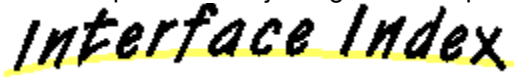


darker()

Returns a darker version of this color.

equals(Object)

Compares this object against the specified object.



getBlue()

Gets the blue component.

- **getColor**(String)

  Gets the specified Color property.
- **getColor**(String, Color)

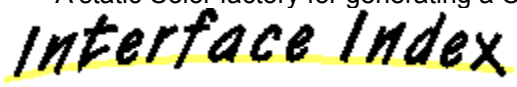
  Gets the specified Color property of the specified Color.
- **getColor**(String, int)

  Gets the specified Color property of the color value.

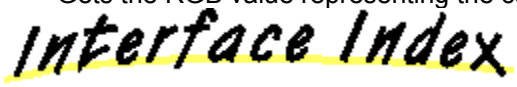
getGreen()

Gets the green component.

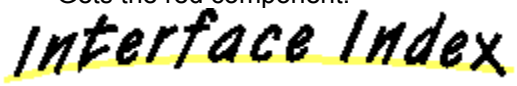
- **getHSBColor**(float, float, float)
  A static Color factory for generating a Color object from HSB values.



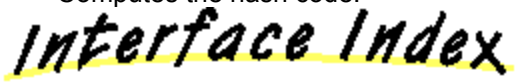
getRGB()
Gets the RGB value representing the color in the default RGB ColorModel.



getRed()
Gets the red component.



hashCode()
Computes the hash code.



toString()
Returns the String representation of this Color's values.

Variables

white

```
public final static Color white
```

The color white.

lightGray

```
public final static Color lightGray
```

The color light gray.

gray

```
public final static Color gray
```

The color gray.

darkGray

```
public final static Color darkGray
```

The color dark gray.

black

```
public final static Color black
```

The color black.

red

```
public final static Color red
```

The color red.

pink

```
public final static Color pink
```

The color pink.

orange

```
public final static Color orange
```

The color orange.

yellow

```
public final static Color yellow
```

The color yellow.

green

```
public final static Color green
```

The color green.

magenta

```
public final static Color magenta
```

The color magneta.

cyan

```
public final static Color cyan
```

The color cyan.

blue

```
public final static Color blue
```

The color blue.

Color

```
public Color(int r,
             int g,
             int b)
```

Creates a color with the specified red, green, and blue values in the range (0 - 255). The actual color used in rendering will depend on finding the best match given the color space available for a given output device.

Parameters:

r - the red component
g - the green component
b - the blue component

See Also:

getRed, getGreen, getBlue, getRGB

Color

```
public Color(int rgb)
```

Creates a color with the specified combined RGB value consisting of the red component in bits 16-

23, the green component in bits 8-15, and the blue component in bits 0-7. The actual color used in rendering will depend on finding the best match given the color space available for a given output device.

Parameters:

rgb - the combined RGB components

See Also:

getRGBdefault, getRed, getGreen, getBlue, getRGB

Color

```
public Color(float r,
             float g,
             float b)
```

Creates a color with the specified red, green, and blue values in the range (0.0 - 1.0). The actual color used in rendering will depend on finding the best match given the color space available for a given output device.

Parameters:

r - the red component
g - the red component
b - the red component

See Also:

getRed, getGreen, getBlue, getRGB

getRed

```
public int getRed()
```

Gets the red component.

See Also:

getRGB

getGreen

```
public int getGreen()
```

Gets the green component.

See Also:
getRGB

getBlue

```
public int getBlue()
```

Gets the blue component.
See Also:
getRGB

getRGB

```
public int getRGB()
```

Gets the RGB value representing the color in the default RGB ColorModel. (Bits 24-31 are 0xff, 16-23 are red, 8-15 are green, 0-7 are blue).
See Also:
getRGBdefault, getRed, getGreen, getBlue

brighter

```
public Color brighter()
```

Returns a brighter version of this color.

darker

```
public Color darker()
```

Returns a darker version of this color.

hashCode

```
public int hashCode()
```

Computes the hash code.

Overrides:
hashCode in class Object

equals

```
public boolean equals(Object obj)
```

Compares this object against the specified object.
Parameters:
obj - the object to compare with.
Returns:
true if the objects are the same; false otherwise.
Overrides:
equals in class Object

toString

```
public String toString()
```

Returns the String representation of this Color's values.
Overrides:
toString in class Object

getColor

```
public static Color getColor(String nm)
```

Gets the specified Color property.
Parameters:
nm - the name of the color property

getColor

```
public static Color getColor(String nm,
                             Color v)
```

Gets the specified Color property of the specified Color.

Parameters:
nm - the name of the color property
v - the specified color
Returns:
the new color.

getColor

```
public static Color getColor(String nm,
                             int v)
```

Gets the specified Color property of the color value.
Parameters:
nm - the name of the color property
v - the color value
Returns:
the new color.

HSBtoRGB

```
public static int HSBtoRGB(float hue,
                           float saturation,
                           float brightness)
```

Returns the RGB value defined by the default RGB ColorModel, of the color corresponding to the given HSB color components.
Parameters:
hue - the hue component of the color
saturation - the saturation of the color
brightness - the brightness of the color
See Also:
getRGBdefault, getRGB

RGBtoHSB

```
public static float[] RGBtoHSB(int r,
                               int g,
                               int b,
                               float hsbvals[])
```

Returns the HSB values corresponding to the color defined by the red, green, and blue components.
Parameters:

r - the red component of the color
g - the green component of the color
b - the blue component of the color
hsbvals - the array to be used to return the 3 HSB values, or null

Returns:

the array used to store the results [hue, saturation, brightness]

See Also:

getRGBdefault, getRGB

getHSBColor

```
public static Color getHSBColor(float h,
                                float s,
                                float b)
```

A static Color factory for generating a Color object from HSB values.

Parameters:

h - the hue component
s - the saturation of the color
b - the brightness of the color

Returns:

the Color object for the corresponding RGB color

Class java.awt.Component

Class java.awt.Component

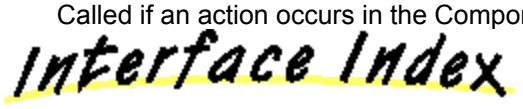
```
java.lang.Object
    |
    +----java.awt.Component
```

---

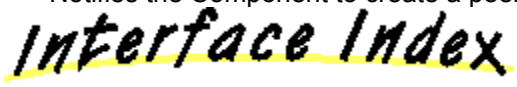
public class **Component**
extends Object
implements ImageObserver

A generic Abstract Window Toolkit component.

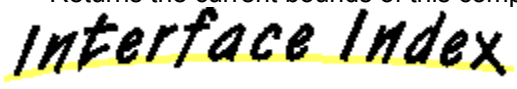
action(Event, Object)
Called if an action occurs in the Component.



addNotify()
Notifies the Component to create a peer.



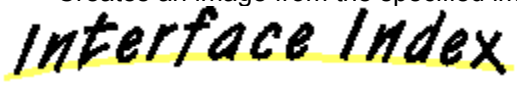
bounds()
Returns the current bounds of this component.



checkImage(Image, ImageObserver)
Returns the status of the construction of a screen representation of the specified image.

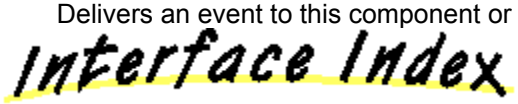
checkImage(Image, int, int, ImageObserver)
Returns the status of the construction of a scaled screen representation of the specified image.

createImage(ImageProducer)
Creates an image from the specified image producer.

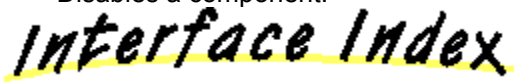


createImage(int, int)
Creates an off-screen drawable Image to be used for double buffering.

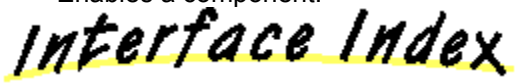
deliverEvent(Event)
Delivers an event to this component or one of its sub components.



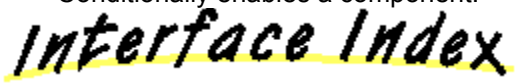
disable()
Disables a component.



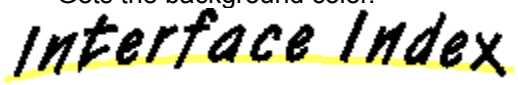
enable()
Enables a component.



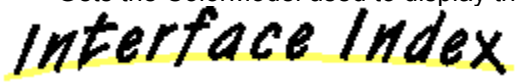
enable(boolean)
Conditionally enables a component.



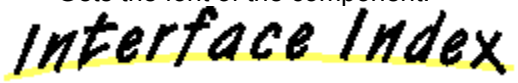
getBackground()
Gets the background color.



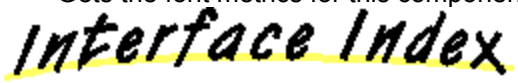
getColorModel()
Gets the ColorModel used to display the component on the output device.



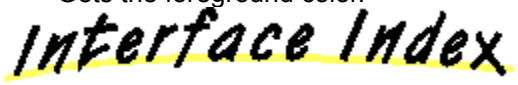
getFont()
Gets the font of the component.



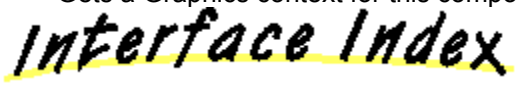
getFontMetrics(Font)
Gets the font metrics for this component.



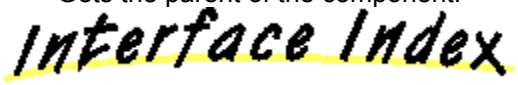
getForeground()
Gets the foreground color.



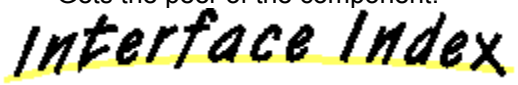
getGraphics()
Gets a Graphics context for this component.



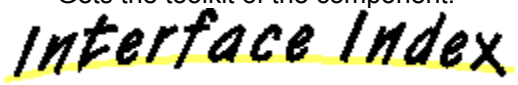
getParent()
Gets the parent of the component.



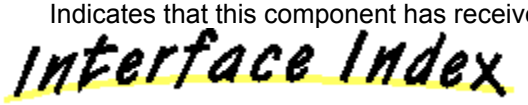
getPeer()
Gets the peer of the component.



getToolkit()
Gets the toolkit of the component.



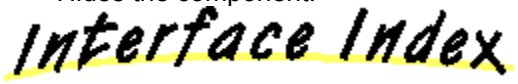
gotFocus(Event, Object)
Indicates that this component has received the input focus.



handleEvent(Event)
Handles the event.

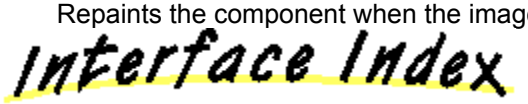
hide()

Hides the component.



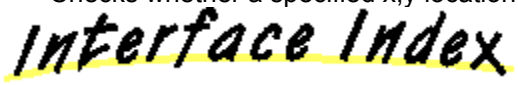
imageUpdate(Image, int, int, int, int, int)

Repaints the component when the image has changed.



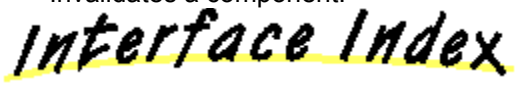
inside(int, int)

Checks whether a specified x,y location is "inside" this Component.



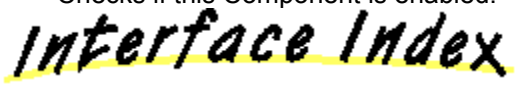
invalidate()

Invalidates a component.



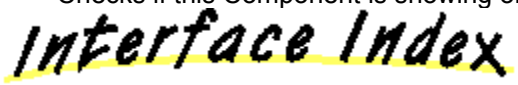
isEnabled()

Checks if this Component is enabled.



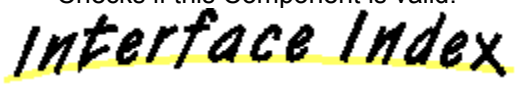
isShowing()

Checks if this Component is showing on screen.



isValid()

Checks if this Component is valid.

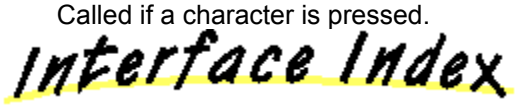


isVisible()

Checks if this Component is visible.

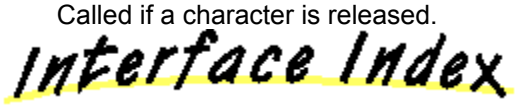
keyDown(Event, int)

Called if a character is pressed.



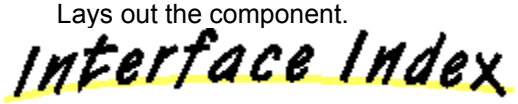
keyUp(Event, int)

Called if a character is released.



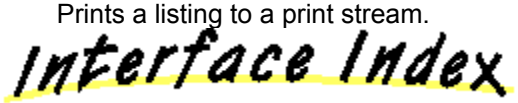
layout()

Lays out the component.



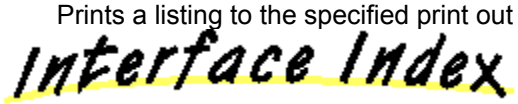
list()

Prints a listing to a print stream.



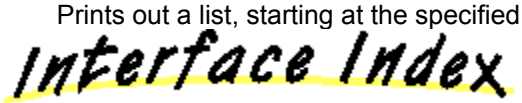
list(PrintStream)

Prints a listing to the specified print out stream.



list(PrintStream, int)

Prints out a list, starting at the specified indention, to the specified print stream.

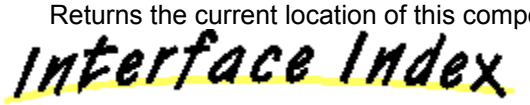


locate(int, int)

Returns the component or subcomponent that contains the x,y location.

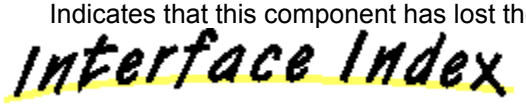
location()

Returns the current location of this component.



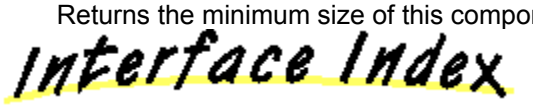
lostFocus(Event, Object)

Indicates that this component has lost the input focus.



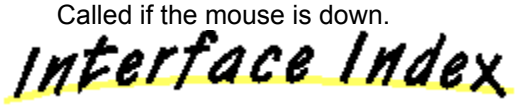
minimumSize()

Returns the minimum size of this component.



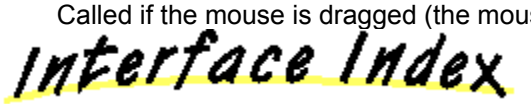
mouseDown(Event, int, int)

Called if the mouse is down.



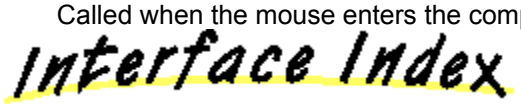
mouseDrag(Event, int, int)

Called if the mouse is dragged (the mouse button is down).



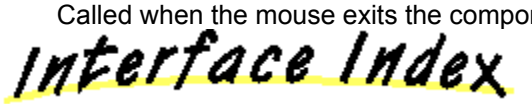
mouseEnter(Event, int, int)

Called when the mouse enters the component.



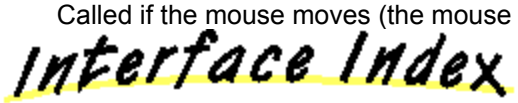
mouseExit(Event, int, int)

Called when the mouse exits the component.



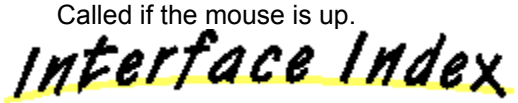
mouseMove(Event, int, int)

Called if the mouse moves (the mouse button is up).



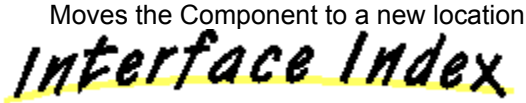
mouseUp(Event, int, int)

Called if the mouse is up.



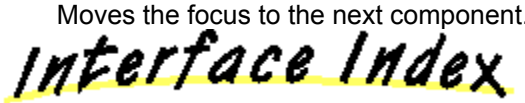
move(int, int)

Moves the Component to a new location.



nextFocus()

Moves the focus to the next component.

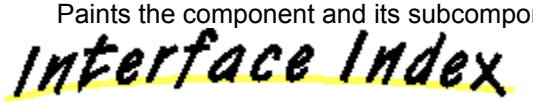


paint(Graphics)

Paints the component.

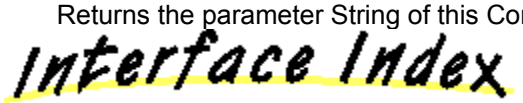
paintAll(Graphics)

Paints the component and its subcomponents.



paramString()

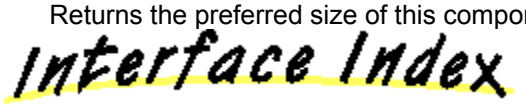
Returns the parameter String of this Component.



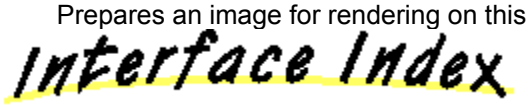
postEvent(Event)

Posts an event to this component.

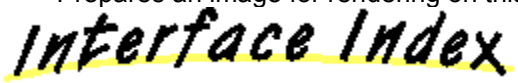
preferredSize()
Returns the preferred size of this component.



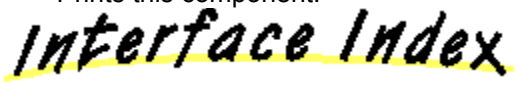
prepareImage(Image, ImageObserver)
Prepares an image for rendering on this Component.



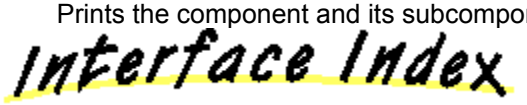
prepareImage(Image, int, int, ImageObserver)
Prepares an image for rendering on this Component at the specified width and height.



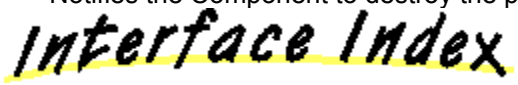
print(Graphics)
Prints this component.



printAll(Graphics)
Prints the component and its subcomponents.

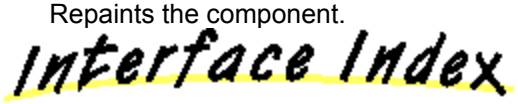


removeNotify()
Notifies the Component to destroy the peer.

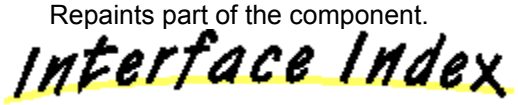


repaint()
Repaints the component.

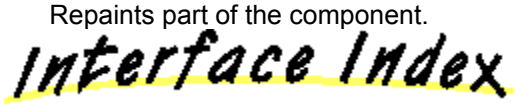
repaint(long)
Repaints the component.



repaint(int, int, int, int)
Repaints part of the component.



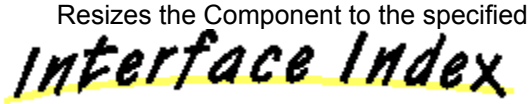
repaint(long, int, int, int, int)
Repaints part of the component.



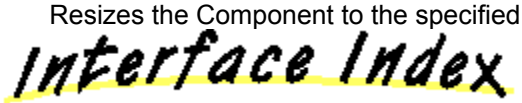
requestFocus()
Requests the input focus.

reshape(int, int, int, int)
Reshapes the Component to the specified bounding box.

resize(int, int)
Resizes the Component to the specified width and height.



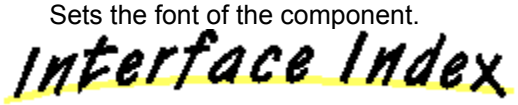
resize(Dimension)
Resizes the Component to the specified dimension.



setBackground(Color)
Sets the background color.

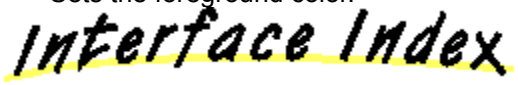
setFont(Font)

Sets the font of the component.



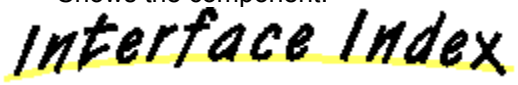
setForeground(Color)

Sets the foreground color.



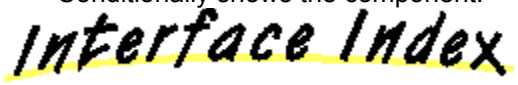
show()

Shows the component.



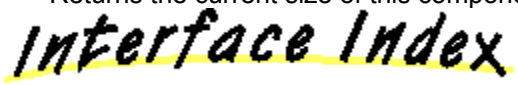
show(boolean)

Conditionally shows the component.



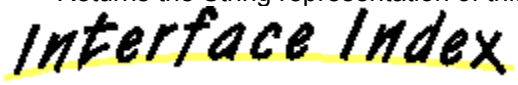
size()

Returns the current size of this component.



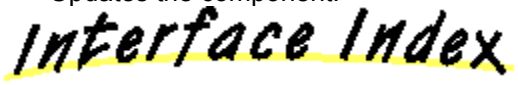
toString()

Returns the String representation of this Component's values.



update(Graphics)

Updates the component.



validate()

Validates a component.

getParent

```
public Container getParent()
```

Gets the parent of the component.

getPeer

```
public ComponentPeer getPeer()
```

Gets the peer of the component.

getToolkit

```
public Toolkit getToolkit()
```

Gets the toolkit of the component. This toolkit is used to create the peer for this component. Note that the Frame which contains a Component controls which toolkit is used so if the Component has not yet been added to a Frame or if it is later moved to a different Frame, the toolkit it uses may change.

isValid

```
public boolean isValid()
```

Checks if this Component is valid. Components are invalidated when they are first shown on the screen.

See Also:

validate, invalidate

isVisible

```
public boolean isVisible()
```

Checks if this Component is visible. Components are initially visible (with the exception of top level components such as Frame).

See Also:

show, hide

isShowing

```
public boolean isShowing()
```

Checks if this Component is showing on screen. This means that the component must be visible, and it must be in a container that is visible and showing.

See Also:

show, hide

isEnabled

```
public boolean isEnabled()
```

Checks if this Component is enabled. Components are initially enabled.

See Also:
enable, disable

location

```
public Point location()
```

Returns the current location of this component. The location will be in the parent's coordinate space.
See Also:
move

size

```
public Dimension size()
```

Returns the current size of this component.
See Also:
resize

bounds

```
public Rectangle bounds()
```

Returns the current bounds of this component.
See Also:
reshape

enable

```
public synchronized void enable()
```

Enables a component.
See Also:
isEnabled, disable

Interface Index **enable**

```
public void enable(boolean cond)
```

Conditionally enables a component.
Parameters:
cond - if true, enables component; disables otherwise.
See Also:
enable, disable

Interface Index **disable**

```
public synchronized void disable()
```

Disables a component.
See Also:
isEnabled, enable

Interface Index **show**

```
public synchronized void show()
```

Shows the component.
See Also:
isVisible, hide

Interface Index **show**

```
public void show(boolean cond)
```

Conditionally shows the component.
Parameters:
cond - if true, it shows the component; hides otherwise.
See Also:
show, hide

Interface Index **hide**

```
public synchronized void hide()
```

Hides the component.
See Also:
isVisible, hide

Interface Index **getForeground**

```
public Color getForeground()
```

Gets the foreground color. If the component does not have a foreground color, the foreground color of its parent is returned.
See Also:
setForeground

Interface Index **setForeground**

```
public synchronized void setForeground(Color c)
```

Sets the foreground color.
Parameters:
c - the Color
See Also:
getForeground

Interface Index **getBackground**

```
public Color getBackground()
```

Gets the background color. If the component does not have a background color, the background color of its parent is returned.
See Also:
setBackground

setBackground

```
public synchronized void setBackground(Color c)
```

Sets the background color.

Parameters:

c - the Color

See Also:

getBackground

getFont

```
public Font getFont()
```

Gets the font of the component. If the component does not have a font, the font of its parent is returned.

See Also:

setFont

setFont

```
public synchronized void setFont(Font f)
```

Sets the font of the component.

Parameters:

f - the font

See Also:

getFont

getColorModel

```
public synchronized ColorModel getColorModel()
```

Gets the ColorModel used to display the component on the output device.

See Also:

ColorModel

 move

```
public void move(int x,
                 int y)
```

Moves the Component to a new location. The x and y coordinates are in the parent's coordinate space.

Parameters:

x - the x coordinate
y - the y coordinate

See Also:

location, reshape

 resize

```
public void resize(int width,
                   int height)
```

Resizes the Component to the specified width and height.

Parameters:

width - the width of the component
height - the height of the component

See Also:

size, reshape

 resize

```
public void resize(Dimension d)
```

Resizes the Component to the specified dimension.

Parameters:

d - the component dimension

See Also:

size, reshape

 reshape

```
public synchronized void reshape(int x,
                                 int y,
                                 int width,
```

```
int height)
```

Reshapes the Component to the specified bounding box.
Parameters:
x - the x coordinate
y - the y coordinate
width - the width of the component
height - the height of the component
See Also:
bounds, move, resize

preferredSize

```
public Dimension preferredSize()
```

Returns the preferred size of this component.
See Also:
minimumSize, LayoutManager

minimumSize

```
public Dimension minimumSize()
```

Returns the minimum size of this component.
See Also:
preferredSize, LayoutManager

layout

```
public void layout()
```

Lays out the component. This is usually called when the component is validated.
See Also:
validate, LayoutManager

validate

```
public void validate()
```

Validates a component.
See Also:
invalidate, layout, LayoutManager

invalidate

```
public void invalidate()
```

Invalidates a component.
See Also:
validate, layout, LayoutManager

getGraphics

```
public Graphics getGraphics()
```

Gets a Graphics context for this component. This method will return null if the component is currently not on the screen.
See Also:
paint

getFontMetrics

```
public FontMetrics getFontMetrics(Font font)
```

Gets the font metrics for this component. This will return null if the component is currently not on the screen.
Parameters:
font - the font
See Also:
getFont

paint

```
public void paint(Graphics g)
```

Paints the component.

Parameters:

g - the specified Graphics window

See Also:

update

update

```
public void update(Graphics g)
```

Updates the component. This method is called in response to a call to repaint. You can assume that the background is not cleared.

Parameters:

g - the specified Graphics window

See Also:

paint, repaint

paintAll

```
public void paintAll(Graphics g)
```

Paints the component and its subcomponents.

Parameters:

g - the specified Graphics window

See Also:

paint

repaint

```
public void repaint()
```

Repaints the component. This will result in a call to update as soon as possible.

See Also:

paint

repaint

```
public void repaint(long tm)
```

Repaints the component. This will result in a call to update within *tm* milliseconds.

Parameters:

tm - maximum time in milliseconds before update

See Also:

paint

repaint

```
public void repaint(int x,
                    int y,
                    int width,
                    int height)
```

Repaints part of the component. This will result in a call to update as soon as possible.

Parameters:

x - the x coordinate
y - the y coordinate
width - the width
height - the height

See Also:

repaint

repaint

```
public void repaint(long tm,
                    int x,
                    int y,
                    int width,
                    int height)
```

Repaints part of the component. This will result in a call to update width *tm* millseconds.

Parameters:

tm - maximum time in milliseconds before update
x - the x coordinate
y - the y coordinate
width - the width
height - the height

See Also:

repaint

 print

```
public void print(Graphics g)
```

Prints this component. The default implementation of this method calls paint.

Parameters:

g - the specified Graphics window

See Also:

paint

 printAll

```
public void printAll(Graphics g)
```

Prints the component and its subcomponents.

Parameters:

g - the specified Graphics window

See Also:

print

 imageUpdate

```
public boolean imageUpdate(Image img,
                           int flags,
                           int x,
                           int y,
                           int w,
                           int h)
```

Repaints the component when the image has changed.

Returns:

true if image has changed; false otherwise.

 createImage

```
public Image createImage(ImageProducer producer)
```

Creates an image from the specified image producer.

Parameters:

producer - the image producer

Interface Index **createImage**

```
public Image createImage(int width,
                         int height)
```

Creates an off-screen drawable Image to be used for double buffering.

Parameters:

width - the specified width
height - the specified height

Interface Index **prepareImage**

```
public boolean prepareImage(Image image,
                            ImageObserver observer)
```

Prepares an image for rendering on this Component. The image data is downloaded asynchronously in another thread and the appropriate screen representation of the image is generated.

Parameters:

image - the Image to prepare a screen representation for
observer - the ImageObserver object to be notified as the image is being prepared

Returns:

true if the image has already been fully prepared

See Also:

ImageObserver

Interface Index **prepareImage**

```
public boolean prepareImage(Image image,
                            int width,
                            int height,
                            ImageObserver observer)
```

Prepares an image for rendering on this Component at the specified width and height. The image data is downloaded asynchronously in another thread and an appropriately scaled screen representation of the image is generated.

Parameters:

image - the Image to prepare a screen representation for
width - the width of the desired screen representation
height - the height of the desired screen representation
observer - the ImageObserver object to be notified as the image is being prepared

Returns:
true if the image has already been fully prepared
See Also:
ImageObserver

Interface Index **checkImage**

```
public int checkImage(Image image,
                      ImageObserver observer)
```

Returns the status of the construction of a screen representation of the specified image. This method does not cause the image to begin loading. Use the prepareImage method to force the loading of an image.
Parameters:
image - the Image to check the status of
observer - the ImageObserver object to be notified as the image is being prepared
Returns:
the boolean OR of the ImageObserver flags for the data that is currently available
See Also:
ImageObserver, prepareImage

Interface Index **checkImage**

```
public int checkImage(Image image,
                      int width,
                      int height,
                      ImageObserver observer)
```

Returns the status of the construction of a scaled screen representation of the specified image. This method does not cause the image to begin loading, use the prepareImage method to force the loading of an image.
Parameters:
image - the Image to check the status of
width - the width of the scaled version to check the status of
height - the height of the scaled version to check the status of
observer - the ImageObserver object to be notified as the image is being prepared
Returns:
the boolean OR of the ImageObserver flags for the data that is currently available
See Also:
ImageObserver, prepareImage

Interface Index **inside**

```
public synchronized boolean inside(int x,
                                   int y)
```

Checks whether a specified x,y location is "inside" this Component. By default, x and y are inside an Component if they fall within the bounding box of that Component.

Parameters:

x - the x coordinate
y - the y coordinate

See Also:

locate

locate

```
public Component locate(int x,
                        int y)
```

Returns the component or subcomponent that contains the x,y location.

Parameters:

x - the x coordinate
y - the y coordinate

See Also:

inside

deliverEvent

```
public void deliverEvent(Event e)
```

Delivers an event to this component or one of its sub components.

Parameters:

e - the event

See Also:

handleEvent, postEvent

postEvent

```
public boolean postEvent(Event e)
```

Posts an event to this component. This will result in a call to handleEvent. If handleEvent returns false the event is passed on to the parent of this component.

Parameters:

e - the event

See Also:
handleEvent, deliverEvent

handleEvent

```
public boolean handleEvent(Event evt)
```

Handles the event. Returns true if the event is handled and should not be passed to the parent of this component. The default event handler calls some helper methods to make life easier on the programmer.

Parameters:
evt - the event

See Also:
mouseEnter, mouseExit, mouseMove, mouseDown, mouseDrag , mouseUp, keyDown, action

mouseDown

```
public boolean mouseDown(Event evt,
                         int x,
                         int y)
```

Called if the mouse is down.

Parameters:
evt - the event
x - the x coordinate
y - the y coordinate

See Also:
handleEvent

mouseDrag

```
public boolean mouseDrag(Event evt,
                         int x,
                         int y)
```

Called if the mouse is dragged (the mouse button is down).

Parameters:
evt - the event
x - the x coordinate
y - the y coordinate

See Also:
handleEvent

mouseUp

```
public boolean mouseUp(Event evt,
                       int x,
                       int y)
```

Called if the mouse is up.

Parameters:

evt - the event
x - the x coordinate
y - the y coordinate

See Also:

handleEvent

mouseMove

```
public boolean mouseMove(Event evt,
                         int x,
                         int y)
```

Called if the mouse moves (the mouse button is up).

Parameters:

evt - the event
x - the x coordinate
y - the y coordinate

See Also:

handleEvent

mouseEnter

```
public boolean mouseEnter(Event evt,
                          int x,
                          int y)
```

Called when the mouse enters the component.

Parameters:

evt - the event
x - the x coordinate
y - the y coordinate

See Also:

handleEvent

mouseExit

```
public boolean mouseExit(Event evt,
                         int x,
                         int y)
```

Called when the mouse exits the component.

Parameters:

evt - the event
x - the x coordinate
y - the y coordinate

See Also:

handleEvent

keyDown

```
public boolean keyDown(Event evt,
                       int key)
```

Called if a character is pressed.

Parameters:

evt - the event
key - the key that's pressed

See Also:

handleEvent

keyUp

```
public boolean keyUp(Event evt,
                     int key)
```

Called if a character is released.

Parameters:

evt - the event
key - the key that's released

See Also:

handleEvent

action

```
public boolean action(Event evt,
                      Object what)
```

Called if an action occurs in the Component.

Parameters:

evt - the event

what - the action that's occuring

See Also:

handleEvent

addNotify

```
public void addNotify()
```

Notifies the Component to create a peer.

See Also:

getPeer, removeNotify

removeNotify

```
public synchronized void removeNotify()
```

Notifies the Component to destroy the peer.

See Also:

getPeer, addNotify

gotFocus

```
public boolean gotFocus(Event evt,
                        Object what)
```

Indicates that this component has received the input focus.

See Also:

requestFocus, lostFocus

lostFocus

```
public boolean lostFocus(Event evt,
                         Object what)
```

Indicates that this component has lost the input focus.

See Also:

requestFocus, gotFocus

requestFocus

```
public void requestFocus()
```

Requests the input focus. The gotFocus() method will be called if this method is successful.

See Also:

gotFocus

nextFocus

```
public void nextFocus()
```

Moves the focus to the next component.

See Also:

requestFocus, gotFocus

paramString

```
protected String paramString()
```

Returns the parameter String of this Component.

toString

```
public String toString()
```

Returns the String representation of this Component's values.

Overrides:

toString in class Object

list

```
public void list()
```

Prints a listing to a print stream.

list

```
public void list(PrintStream out)
```

Prints a listing to the specified print out stream.

Parameters:

out - the Stream name

list

```
public void list(PrintStream out,
                 int indent)
```

Prints out a list, starting at the specified indention, to the specified print stream.

Parameters:

out - the Stream name

indent - the start of the list

Class java.awt.Container

Class java.awt.Container

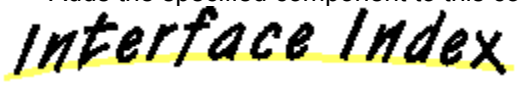
```
java.lang.Object
    |
    +----java.awt.Component
              |
              +----java.awt.Container
```

---

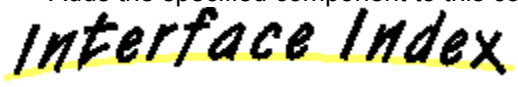
public class **Container**
extends Component

A generic Abstract Window Toolkit(AWT) container object is a component that can contain other AWT components.

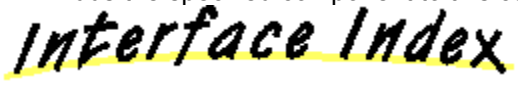
add(Component)
Adds the specified component to this container.



add(Component, int)
Adds the specified component to this container at the given position.



add(String, Component)
Adds the specified component to this container.



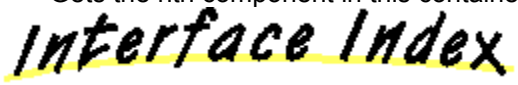
addNotify()
Notifies the container to create a peer.

countComponents()
Returns the number of components in this panel.

deliverEvent(Event)
Delivers an event.

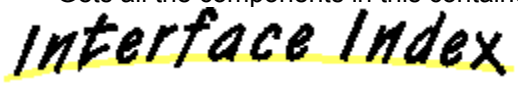
getComponent(int)

Gets the nth component in this container.



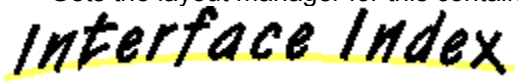
getComponents()

Gets all the components in this container.



getLayout()

Gets the layout manager for this container.

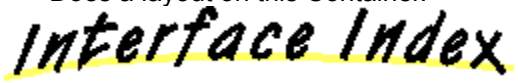


insets()

Returns the insets of the container.

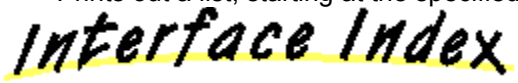
layout()

Does a layout on this Container.



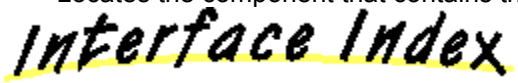
list(PrintStream, int)

Prints out a list, starting at the specified indention, to the specified out stream.



locate(int, int)

Locates the component that contains the x,y position.

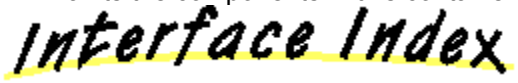


minimumSize()

Returns the minimum size of this container.

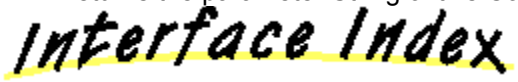
paintComponents(Graphics)

Paints the components in this container.



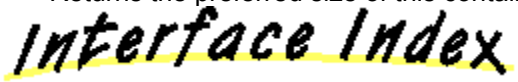
paramString()

Returns the parameter String of this Container.



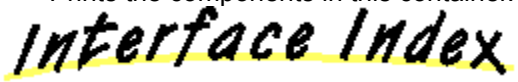
preferredSize()

Returns the preferred size of this container.



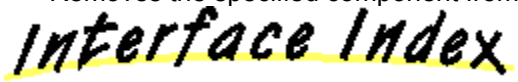
printComponents(Graphics)

Prints the components in this container.



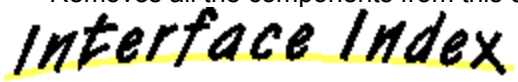
remove(Component)

Removes the specified component from this container.



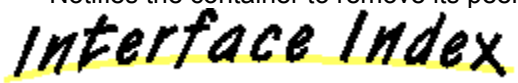
removeAll()

Removes all the components from this container.



removeNotify()

Notifies the container to remove its peer.



setLayout(LayoutManager)

Sets the layout manager for this container.

validate()
Validates this Container and all of the components contained within it.

countComponents

```
public int countComponents()
```

Returns the number of components in this panel.
See Also:
getComponent

getComponent

```
public synchronized Component getComponent(int n)
```

Gets the nth component in this container.
Parameters:
n - the number of the component to get
Throws: ArrayIndexOutOfBoundsException
If the nth value does not exist.

getComponents

```
public synchronized Component[] getComponents()
```

Gets all the components in this container.

insets

```
public Insets insets()
```

Returns the insets of the container. The insets indicate the size of the border of the container. A Frame, for example, will have a top inset that corresponds to the height of the Frame's title bar.
See Also:
LayoutManager

add

```
public Component add(Component comp)
```

Adds the specified component to this container.
Parameters:
comp - the component to be added

add

```
public synchronized Component add(Component comp,
                                  int pos)
```

Adds the specified component to this container at the given position.
Parameters:
comp - the component to be added
pos - the position at which to insert the component. -1 means insert at the end.
See Also:
remove

add

```
public synchronized Component add(String name,
                                  Component comp)
```

Adds the specified component to this container. The component is also added to the layout manager of this container using the name specified .
Parameters:
name - the component name
comp - the component to be added
See Also:
remove, LayoutManager

remove

```
public synchronized void remove(Component comp)
```

Removes the specified component from this container.
Parameters:
comp - the component to be removed
See Also:
add

removeAll

```
public synchronized void removeAll()
```

Removes all the components from this container.
See Also:
add, remove

getLayout

```
public LayoutManager getLayout()
```

Gets the layout manager for this container.
See Also:
layout, setLayout

setLayout

```
public void setLayout(LayoutManager mgr)
```

Sets the layout manager for this container.
Parameters:
mgr - the specified layout manager
See Also:
layout, getLayout

layout

```
public synchronized void layout()
```

Does a layout on this Container.
Overrides:
layout in class Component
See Also:
setLayout

validate

```
public synchronized void validate()
```

Validates this Container and all of the components contained within it.
Overrides:
validate in class Component
See Also:
validate, invalidate

preferredSize

```
public synchronized Dimension preferredSize()
```

Returns the preferred size of this container.
Overrides:
preferredSize in class Component
See Also:
minimumSize

minimumSize

```
public synchronized Dimension minimumSize()
```

Returns the minimum size of this container.
Overrides:
minimumSize in class Component
See Also:
preferredSize

paintComponents

```
public void paintComponents(Graphics g)
```

Paints the components in this container.
Parameters:
g - the specified Graphics window
See Also:
paint, paintAll

printComponents

```
public void printComponents(Graphics g)
```

Prints the components in this container.
Parameters:
g - the specified Graphics window
See Also:
print, printAll

deliverEvent

```
public void deliverEvent(Event e)
```

Delivers an event. The appropriate component is located and the event is delivered to it.
Parameters:
e - the event
Overrides:
deliverEvent in class Component
See Also:
handleEvent, postEvent

locate

```
public Component locate(int x,
                        int y)
```

Locates the component that contains the x,y position.
Parameters:
x - the x coordinate
y - the y coordinate
Returns:
null if the component is not within the x and y coordinates; returns the component otherwise.

Overrides:
locate in class Component
See Also:
inside

addNotify

```
public synchronized void addNotify()
```

Notifies the container to create a peer. It will also notify the components contained in this container.
Overrides:
addNotify in class Component
See Also:
removeNotify

removeNotify

```
public synchronized void removeNotify()
```

Notifies the container to remove its peer. It will also notify the components contained in this container.
Overrides:
removeNotify in class Component
See Also:
addNotify

paramString

```
protected String paramString()
```

Returns the parameter String of this Container.
Overrides:
paramString in class Component

list

```
public void list(PrintStream out,
                 int indent)
```

Prints out a list, starting at the specified indention, to the specified out stream.

Parameters:

out - the Stream name
indent - the start of the list

Overrides:

list in class Component

Class java.awt.Dialog

Class java.awt.Dialog

```
java.lang.Object
    |
    +----java.awt.Component
            |
            +----java.awt.Container
                    |
                    +----java.awt.Window
                            |
                            +----java.awt.Dialog
```

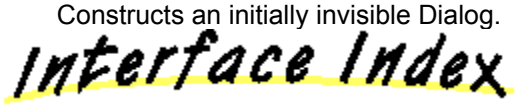
---

public class **Dialog**
extends Window

A class that produces a dialog - a window that takes input from the user. The default layout for a dialog is BorderLayout.

Dialog(Frame, boolean)

Constructs an initially invisible Dialog.

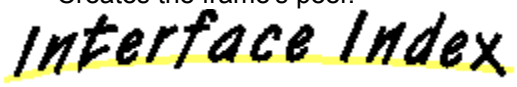


Dialog(Frame, String, boolean)

Constructs an initially invisible Dialog with a title.

addNotify()

Creates the frame's peer.



getTitle()

Gets the title of the Dialog.

Interface Index **isModal**()
Returns true if the Dialog is modal.
Interface Index **isResizable**()
Returns true if the user can resize the frame.
Interface Index **paramString**()
Returns the parameter String of this Dialog.
Interface Index **setResizable**(boolean)
Sets the resizable flag.
Interface Index **setTitle**(String)
Sets the title of the Dialog.

Interface Index

Interface Index **Dialog**

```
public Dialog(Frame parent,
              boolean modal)
```

Constructs an initially invisible Dialog. A modal Dialog grabs all the input from the user.

Parameters:

parent - the owner of the dialog
modal - if true, dialog blocks input to other windows when shown

See Also:

resize, show

Interface Index **Dialog**

```
public Dialog(Frame parent,
              String title,
              boolean modal)
```

Constructs an initially invisible Dialog with a title. A modal Dialog grabs all the input from the user.

Parameters:

parent - the owner of the dialog
title - the title of the dialog
modal - if true, dialog blocks input to other windows when shown

See Also:

resize, show

Interface Index

addNotify

```
public synchronized void addNotify()
```

Creates the frame's peer. The peer allows us to change the appearance of the frame without changing its functionality.

Overrides:

addNotify in class Window

Interface Index

isModal

```
public boolean isModal()
```

Returns true if the Dialog is modal. A modal Dialog grabs all the input from the user.

Interface Index

getTitle

```
public String getTitle()
```

Gets the title of the Dialog.

See Also:

setTitle

Interface Index

setTitle

```
public void setTitle(String title)
```

Sets the title of the Dialog.

Parameters:

title - the new title being given to the Dialog

See Also:

getTitle

isResizable

```
public boolean isResizable()
```

Returns true if the user can resize the frame.

setResizable

```
public void setResizable(boolean resizable)
```

Sets the resizable flag.

Parameters:

resizable - true if resizable; false otherwise

paramString

```
protected String paramString()
```

Returns the parameter String of this Dialog.

Overrides:

paramString in class Container

Class java.awt.Dimension

Class java.awt.Dimension

```
java.lang.Object
    |
    +----java.awt.Dimension
```

---

public class **Dimension**
extends Object

A class to encapsulate a width and a height Dimension.

- **height**
  The height dimension.
- **width**
  The width dimension.

Dimension()
Constructs a Dimension with a 0 width and 0 height.

Dimension(Dimension)
Constructs a Dimension and initializes it to the specified value.

Dimension(int, int)
Constructs a Dimension and initializes it to the specified width and specified height.

Interface Index **toString**()

Returns the String representation of this Dimension's values.

Variables

Interface Index **width**

```
public int width
```

The width dimension.

Interface Index **height**

```
public int height
```

The height dimension.

Interface Index

Interface Index **Dimension**

```
public Dimension()
```

Constructs a Dimension with a 0 width and 0 height.

Interface Index **Dimension**

```
public Dimension(Dimension d)
```

Constructs a Dimension and initializes it to the specified value.

Parameters:

d - the specified dimension for the width and height values

Interface Index **Dimension**

```
public Dimension(int width,
                 int height)
```

Constructs a Dimension and initializes it to the specified width and specified height.

Parameters:

width - the specified width dimension
height - the specified height dimension

toString

```
public String toString()
```

Returns the String representation of this Dimension's values.

Overrides:

toString in class Object

Class java.awt.Event

Class java.awt.Event

```
java.lang.Object
    |
    +----java.awt.Event
```

---

public class **Event**
extends Object

Event is a platform-independent class that encapsulates events from the local Graphical User Interface(GUI) platform.

ACTION_EVENT

An action event.

ALT_MASK

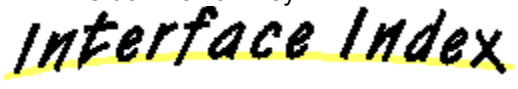
The alt modifier constant.

CTRL_MASK

The control modifier constant.

DOWN

The down arrow key.

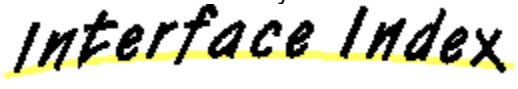


END

The end key.

F1

The F1 function key.

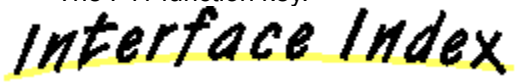


F10

The F10 function key.

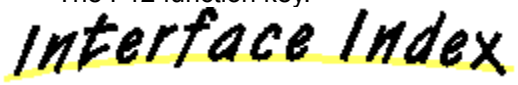
F11

The F11 function key.



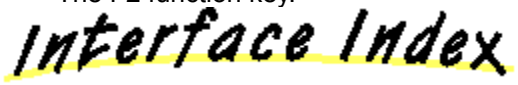
F12

The F12 function key.



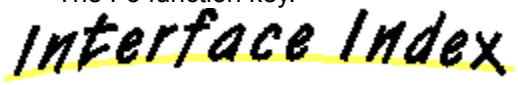
F2

The F2 function key.



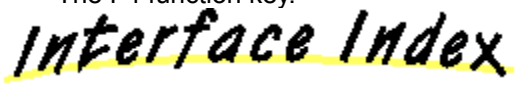
F3

The F3 function key.



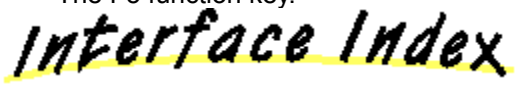
F4

The F4 function key.



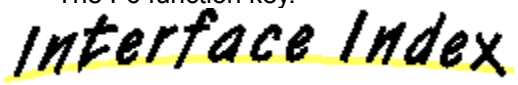
F5

The F5 function key.



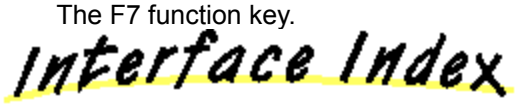
F6

The F6 function key.



F7

The F7 function key.



F8

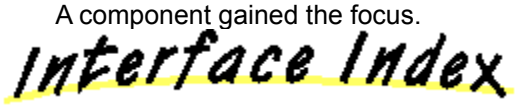
The F8 function key.

F9

The F9 function key.

GOT_FOCUS

A component gained the focus.

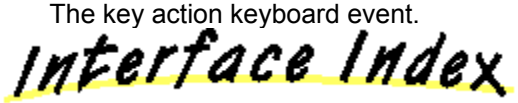


HOME

The home key.

KEY_ACTION

The key action keyboard event.



KEY_ACTION_RELEASE

The key action keyboard event.

KEY_PRESS

The key press keyboard event.

Interface Index

KEY_RELEASE

The key release keyboard event.

Interface Index

LEFT

The left arrow key.

Interface Index

LIST_DESELECT

Interface Index

LIST_SELECT

Interface Index

LOAD_FILE

A file loading event.

Interface Index

LOST_FOCUS

A component lost the focus.

Interface Index

META_MASK

The meta modifier constant.

Interface Index

MOUSE_DOWN

The mouse down event.

Interface Index

MOUSE_DRAG

The mouse drag event.

Interface Index

MOUSE_ENTER

The mouse enter event.

Interface Index

MOUSE_EXIT

The mouse exit event.

Interface Index

MOUSE_MOVE

The mouse move event.

Interface Index

MOUSE_UP

The mouse up event.

Interface Index

PGDN

The page down key.

Interface Index

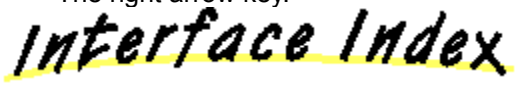
PGUP

The page up key.

Interface Index

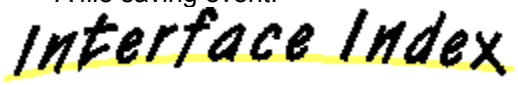
RIGHT

The right arrow key.



SAVE_FILE

A file saving event.



SCROLL_ABSOLUTE

The absolute scroll event.

SCROLL_LINE_DOWN

The line down scroll event.

SCROLL_LINE_UP

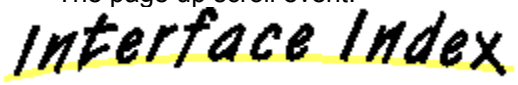
The line up scroll event.

SCROLL_PAGE_DOWN

The page down scroll event.

SCROLL_PAGE_UP

The page up scroll event.

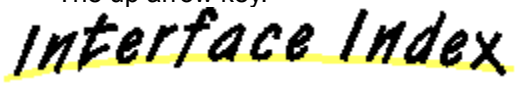


SHIFT_MASK

The shift modifier constant.

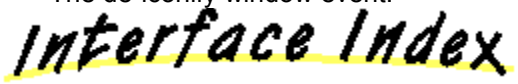
UP

The up arrow key.



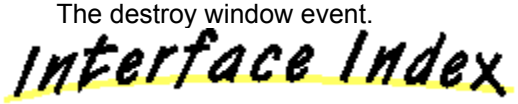
WINDOW_DEICONIFY

The de-iconify window event.



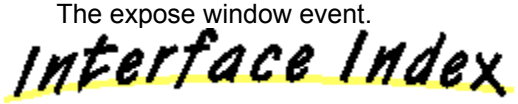
WINDOW_DESTROY

The destroy window event.



WINDOW_EXPOSE

The expose window event.



WINDOW_ICONIFY

The iconify window event.

WINDOW_MOVED

The move window event.

- **arg**
  An arbitraty argument.
- **clickCount**
  The number of consecutive clicks.

evt

The next event.

Interface Index **id**

The type of this event.

Interface Index **key**

The key that was pressed in a keyboard event.

Interface Index **modifiers**

The state of the modifier keys.

Interface Index **target**

The target component.

Interface Index **when**

The time stamp.

Interface Index **x**

The x coordinate of the event.

Interface Index **y**

The y coordinate of the event.

Interface Index

Interface Index **Event**(Object, long, int, int, int, int, int, Object)

Constructs an event with the specified target component, time stamp, event type, x and y coordinates, keyboard key, state of the modifier keys and argument.

Interface Index **Event**(Object, long, int, int, int, int, int)

Constructs an event with the specified target component, time stamp, event type, x and y coordinates, keyboard key, state of the modifier keys and an argument set to null.

Interface Index **Event**(Object, int, Object)

Constructs an event with the specified target component, event type, and argument.

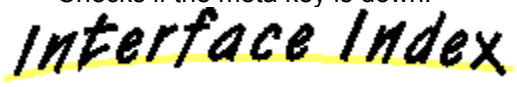
Interface Index

Interface Index **controlDown**()

Checks if the control key is down.

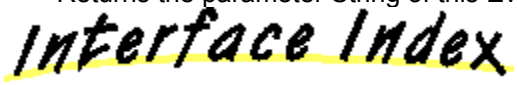
Interface Index **metaDown**()

Checks if the meta key is down.



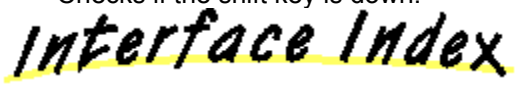
paramString()

Returns the parameter String of this Event.



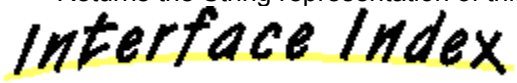
shiftDown()

Checks if the shift key is down.



toString()

Returns the String representation of this Event's values.



translate(int, int)

Translates an event relative to the given component.

SHIFT_MASK

```
public final static int SHIFT_MASK
```

The shift modifier constant.

CTRL_MASK

```
public final static int CTRL_MASK
```

The control modifier constant.

META_MASK

```
public final static int META_MASK
```

The meta modifier constant.

ALT_MASK

```
public final static int ALT_MASK
```

The alt modifier constant.

Interface Index HOME

```
public final static int HOME
```

The home key.

Interface Index END

```
public final static int END
```

The end key.

Interface Index PGUP

```
public final static int PGUP
```

The page up key.

Interface Index PGDN

```
public final static int PGDN
```

The page down key.

Interface Index UP

```
public final static int UP
```

The up arrow key.

Interface Index DOWN

```
public final static int DOWN
```

The down arrow key.

Interface Index LEFT

```
public final static int LEFT
```

The left arrow key.


```
public final static int RIGHT
```

The right arrow key.


```
public final static int F1
```

The F1 function key.


```
public final static int F2
```

The F2 function key.


```
public final static int F3
```

The F3 function key.


```
public final static int F4
```

The F4 function key.


```
public final static int F5
```

The F5 function key.

Interface Index F6

```
public final static int F6
```

The F6 function key.

Interface Index F7

```
public final static int F7
```

The F7 function key.

Interface Index F8

```
public final static int F8
```

The F8 function key.

Interface Index F9

```
public final static int F9
```

The F9 function key.

Interface Index F10

```
public final static int F10
```

The F10 function key.

Interface Index F11

```
public final static int F11
```

The F11 function key.

Interface Index F12

```
public final static int F12
```

The F12 function key.

WINDOW_DESTROY

```
public final static int WINDOW_DESTROY
```

The destroy window event.

WINDOW_EXPOSE

```
public final static int WINDOW_EXPOSE
```

The expose window event.

WINDOW_ICONIFY

```
public final static int WINDOW_ICONIFY
```

The iconify window event.

WINDOW_DEICONIFY

```
public final static int WINDOW_DEICONIFY
```

The de-iconify window event.

WINDOW_MOVED

```
public final static int WINDOW_MOVED
```

The move window event.

KEY_PRESS

```
public final static int KEY_PRESS
```

The key press keyboard event.

KEY_RELEASE

```
public final static int KEY_RELEASE
```

The key release keyboard event.

KEY_ACTION

```
public final static int KEY_ACTION
```

The key action keyboard event.

KEY_ACTION_RELEASE

```
public final static int KEY_ACTION_RELEASE
```

The key action keyboard event.

MOUSE_DOWN

```
public final static int MOUSE_DOWN
```

The mouse down event.

MOUSE_UP

```
public final static int MOUSE_UP
```

The mouse up event.

MOUSE_MOVE

```
public final static int MOUSE_MOVE
```

The mouse move event.

Interface Index **MOUSE_ENTER**

```
public final static int MOUSE_ENTER
```

The mouse enter event.

Interface Index **MOUSE_EXIT**

```
public final static int MOUSE_EXIT
```

The mouse exit event.

Interface Index **MOUSE_DRAG**

```
public final static int MOUSE_DRAG
```

The mouse drag event.

Interface Index **SCROLL_LINE_UP**

```
public final static int SCROLL_LINE_UP
```

The line up scroll event.

Interface Index **SCROLL_LINE_DOWN**

```
public final static int SCROLL_LINE_DOWN
```

The line down scroll event.

Interface Index **SCROLL_PAGE_UP**

```
public final static int SCROLL_PAGE_UP
```

The page up scroll event.

Interface Index **SCROLL_PAGE_DOWN**

```
public final static int SCROLL_PAGE_DOWN
```

The page down scroll event.

Interface Index SCROLL_ABSOLUTE

```
public final static int SCROLL_ABSOLUTE
```

The absolute scroll event.

Interface Index LIST_SELECT

```
public final static int LIST_SELECT
```

Interface Index LIST_DESELECT

```
public final static int LIST_DESELECT
```

Interface Index ACTION_EVENT

```
public final static int ACTION_EVENT
```

An action event.

Interface Index LOAD_FILE

```
public final static int LOAD_FILE
```

A file loading event.

Interface Index SAVE_FILE

```
public final static int SAVE_FILE
```

A file saving event.

Interface Index GOT_FOCUS

```
public final static int GOT_FOCUS
```

A component gained the focus.

LOST_FOCUS

```
public final static int LOST_FOCUS
```

A component lost the focus.

target

```
public Object target
```

The target component.

when

```
public long when
```

The time stamp.

id

```
public int id
```

The type of this event.

x

```
public int x
```

The x coordinate of the event.

y

```
public int y
```

The y coordinate of the event.

key

```
public int key
```

The key that was pressed in a keyboard event.

modifiers

```
public int modifiers
```

The state of the modifier keys.

clickCount

```
public int clickCount
```

The number of consecutive clicks. This field is relevant only for MOUSE_DOWN events. If the field isn't set it will be 0. Otherwise, it will be 1 for single-clicks, 2 for double-clicks, and so on.

arg

```
public Object arg
```

An arbitraty argument.

evt

```
public Event evt
```

The next event. Used when putting events into a linked list.

Event

```
public Event(Object target,
```

```
                long when,
                int id,
                int x,
                int y,
                int key,
                int modifiers,
                Object arg)
```

Constructs an event with the specified target component, time stamp, event type, x and y coordinates, keyboard key, state of the modifier keys and argument.

Parameters:

target - the target component
when - the time stamp
id - the event type
x - the x coordinate
y - the y coordinate
key - the key pressed in a keyboard event
modifiers - the state of the modifier keys
arg - the specified argument

Event

```
public Event(Object target,
                long when,
                int id,
                int x,
                int y,
                int key,
                int modifiers)
```

Constructs an event with the specified target component, time stamp, event type, x and y coordinates, keyboard key, state of the modifier keys and an argument set to null.

Parameters:

target - the target component
when - the time stamp
id - the event type
x - the x coordinate
y - the y coordinate
key - the key pressed in a keyboard event
modifiers - the state of the modifier keys

Event

```
public Event(Object target,
                int id,
                Object arg)
```

Constructs an event with the specified target component, event type, and argument.

Parameters:

target - the target component
id - the event type
arg - the specified argument

Interface Index

translate

```
public void translate(int x,
                      int y)
```

Translates an event relative to the given component. This involves at a minimum translating the coordinates so they make sense within the given component. It may also involve translating a region in the case of an expose event.

Parameters:

x - the x coordinate
y - the y coordinate

Interface Index

shiftDown

```
public boolean shiftDown()
```

Checks if the shift key is down.

See Also:

modifiers, controlDown, metaDown

Interface Index

controlDown

```
public boolean controlDown()
```

Checks if the control key is down.

See Also:

modifiers, shiftDown, metaDown

metaDown

```
public boolean metaDown()
```

Checks if the meta key is down.

See Also:

modifiers, shiftDown, controlDown

paramString

```
protected String paramString()
```

Returns the parameter String of this Event.

toString

```
public String toString()
```

Returns the String representation of this Event's values.

Overrides:

toString in class Object

Class java.awt.FileDialog

Class java.awt.FileDialog

```
java.lang.Object
    |
    +----java.awt.Component
            |
            +----java.awt.Container
                    |
                    +----java.awt.Window
                            |
                            +----java.awt.Dialog
                                    |
                                    +----java.awt.FileDialog
```

---

public class **FileDialog**
extends Dialog

The File Dialog class displays a file selection dialog. It is a modal dialog and will block the calling thread when the show method is called to display it, until the user has chosen a file.

See Also:

show

LOAD

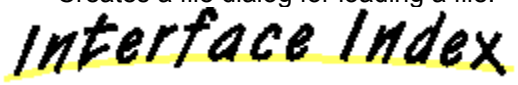
The file load variable.

SAVE

The file save variable.

FileDialog(Frame, String)

Creates a file dialog for loading a file.

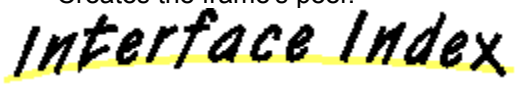


FileDialog(Frame, String, int)

Creates a file dialog with the specified title and mode.

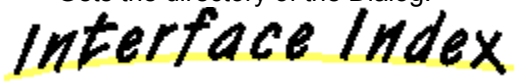
addNotify()

Creates the frame's peer.



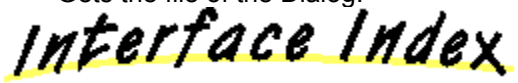
getDirectory()

Gets the directory of the Dialog.



getFile()

Gets the file of the Dialog.

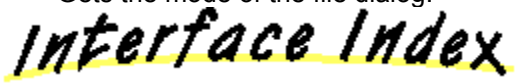


getFilenameFilter()

Gets the filter.

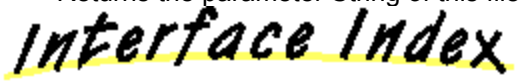
getMode()

Gets the mode of the file dialog.



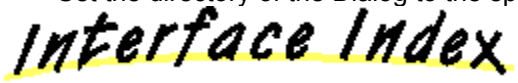
paramString()

Returns the parameter String of this file dialog.



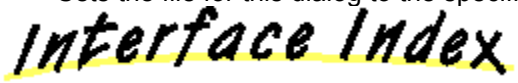
setDirectory(String)

Set the directory of the Dialog to the specified directory.



setFile(String)

Sets the file for this dialog to the specified file.



setFilenameFilter(FilenameFilter)

Sets the filter for this dialog to the specified filter.

LOAD

```
public final static int LOAD
```

The file load variable.

SAVE

```
public final static int SAVE
```

The file save variable.

FileDialog

```
public FileDialog(Frame parent,
                  String title)
```

Creates a file dialog for loading a file.
Parameters:
parent - the owner of the dialog
title - the title of the Dialog

FileDialog

```
public FileDialog(Frame parent,
                  String title,
                  int mode)
```

Creates a file dialog with the specified title and mode.
Parameters:
parent - the owner of the dialog
title - the title of the Dialog
mode - the mode of the Dialog

addNotify

```
public synchronized void addNotify()
```

Creates the frame's peer. The peer allows us to change the look of the file dialog without changing its functionality.
Overrides:
addNotify in class Dialog

getMode

```
public int getMode()
```

Gets the mode of the file dialog.

getDirectory

```
public String getDirectory()
```

Gets the directory of the Dialog.

setDirectory

```
public void setDirectory(String dir)
```

Set the directory of the Dialog to the specified directory.
Parameters:
dir - the specific directory

getFile

```
public String getFile()
```

Gets the file of the Dialog.

setFile

```
public void setFile(String file)
```

Sets the file for this dialog to the specified file. This will become the default file if set before the dialog is shown.

Parameters:
file - the file being set

getFilenameFilter

```
public FilenameFilter getFilenameFilter()
```

Gets the filter.

setFilenameFilter

```
public void setFilenameFilter(FilenameFilter filter)
```

Sets the filter for this dialog to the specified filter.
Parameters:
filter - the specified filter

paramString

```
protected String paramString()
```

Returns the parameter String of this file dialog. Parameter String.
Overrides:
paramString in class Dialog

Class java.awt.FlowLayout

Class java.awt.FlowLayout

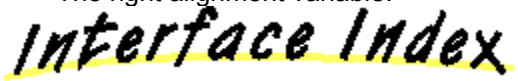
```
java.lang.Object
    |
    +----java.awt.FlowLayout
```

---

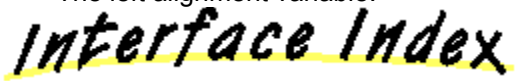
public class **FlowLayout**
extends Object
implements LayoutManager

Flow layout is used to layout buttons in a panel. It will arrange buttons left to right until no more buttons fit on the same line. Each line is centered.

CENTER
The right alignment variable.



LEFT
The left alignment variable.



RIGHT
The right alignment variable.

FlowLayout()
Constructs a new Flow Layout with a centered alignment.

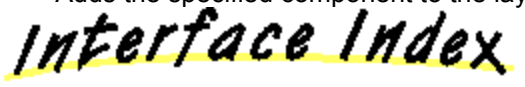
FlowLayout(int)
Constructs a new Flow Layout with the specified alignment.

FlowLayout(int, int, int)

Constructs a new Flow Layout with the specified alignment and gap values.

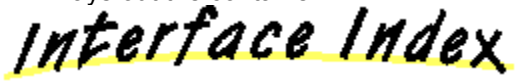
addLayoutComponent(String, Component)

Adds the specified component to the layout.



layoutContainer(Container)

Lays out the container.



minimumLayoutSize(Container)

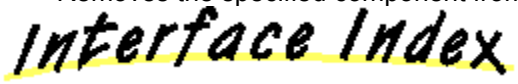
Returns the minimum dimensions needed to layout the components contained in the specified target container.

preferredLayoutSize(Container)

Returns the preferred dimensions for this layout given the components in the specified target container.

removeLayoutComponent(Component)

Removes the specified component from the layout.



toString()

Returns the String representation of this FlowLayout's values.

LEFT

```
public final static int LEFT
```

The left alignment variable.

CENTER

```
public final static int CENTER
```

The right alignment variable.

RIGHT

```
public final static int RIGHT
```

The right alignment variable.

FlowLayout

```
public FlowLayout()
```

Constructs a new Flow Layout with a centered alignment.

FlowLayout

```
public FlowLayout(int align)
```

Constructs a new Flow Layout with the specified alignment.

Parameters:

align - the alignment value

FlowLayout

```
public FlowLayout(int align,
                  int hgap,
                  int vgap)
```

Constructs a new Flow Layout with the specified alignment and gap values.

Parameters:

align - the alignment value
hgap - the horizontal gap variable
vgap - the vertical gap variable

addLayoutComponent

```
public void addLayoutComponent(String name,
                               Component comp)
```

Adds the specified component to the layout. Not used by this class.

Parameters:

name - the name of the component
comp - the the component to be added

removeLayoutComponent

```
public void removeLayoutComponent(Component comp)
```

Removes the specified component from the layout. Not used by this class.

Parameters:

comp - the component to remove

preferredLayoutSize

```
public Dimension preferredLayoutSize(Container target)
```

Returns the preferred dimensions for this layout given the components in the specified target container.

Parameters:

target - the component which needs to be laid out

See Also:

Container, minimumLayoutSize

minimumLayoutSize

```
public Dimension minimumLayoutSize(Container target)
```

Returns the minimum dimensions needed to layout the components contained in the specified target container.

Parameters:

target - the component which needs to be laid out

See Also:

preferredLayoutSize

layoutContainer

```
public void layoutContainer(Container target)
```

Lays out the container. This method will actually reshape the components in the target in order to satisfy the constraints of the BorderLayout object.

Parameters:

target - the specified component being laid out.

See Also:

Container

toString

```
public String toString()
```

Returns the String representation of this FlowLayout's values.

Overrides:

toString in class Object

Class java.awt.Font

Class java.awt.Font

```
java.lang.Object
    |
    +----java.awt.Font
```

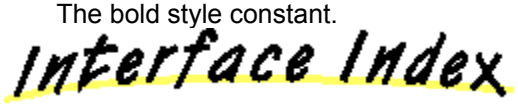
---

public class **Font**
extends Object

A class that produces font objects.

BOLD

The bold style constant.



ITALIC

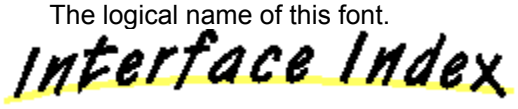
The italicized style constant.

PLAIN

The plain style constant.

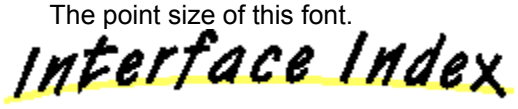
name

The logical name of this font.



size

The point size of this font.

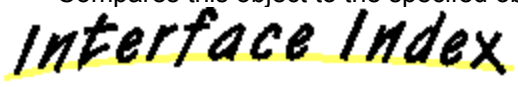


style

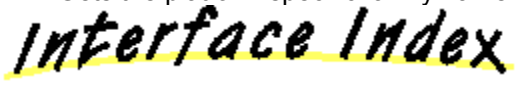
The style of the font.

Font(String, int, int)
Creates a new font with the specified name, style and point size.

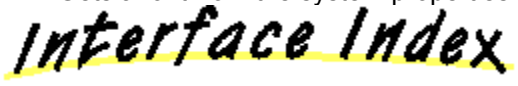
equals(Object)
Compares this object to the specifed object.



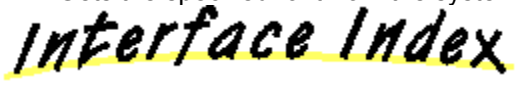
getFamily()
Gets the platform specific family name of the font.



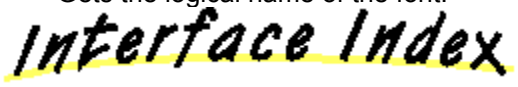
getFont(String)
Gets a font from the system properties list.



getFont(String, Font)
Gets the specified font from the system properties list.

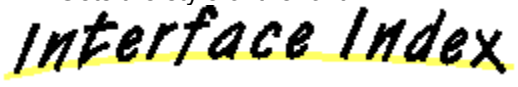


getName()
Gets the logical name of the font.



getSize()
Gets the point size of the font.

getStyle()
Gets the style of the font.



hashCode()
Returns a hashcode for this font.

isBold()
Returns true if the font is bold.

isItalic()
Returns true if the font is italic.

isPlain()
Returns true if the font is plain.

toString()
Converts this object to a String representation.

Interface Index

Interface Index PLAIN

```
public final static int PLAIN
```

The plain style constant. This can be combined with the other style constants for mixed styles.

Interface Index BOLD

```
public final static int BOLD
```

The bold style constant. This can be combined with the other style constants for mixed styles.

Interface Index ITALIC

```
public final static int ITALIC
```

The italicized style constant. This can be combined with the other style constants for mixed styles.

Interface Index name

```
protected String name
```

The logical name of this font.

Interface Index style

```
protected int style
```

The style of the font. This is the sum of the constants PLAIN, BOLD, or ITALIC.

Interface Index size

```
protected int size
```

The point size of this font.

Interface Index

Interface Index **Font**

```
public Font(String name,
            int style,
            int size)
```

Creates a new font with the specified name, style and point size.

Parameters:

name - the font name
style - the constant style used
size - the point size of the font

See Also:

getFontList

Interface Index

Interface Index **getFamily**

```
public String getFamily()
```

Gets the platform specific family name of the font. Use getName to get the logical name of the font.

See Also:

getName

Interface Index **getName**

```
public String getName()
```

Gets the logical name of the font.

See Also:

getFamily

Interface Index **getStyle**

```
public int getStyle()
```

Gets the style of the font.
See Also:
isPlain, isBold, isItalic

Interface Index **getSize**

```
public int getSize()
```

Gets the point size of the font.

Interface Index **isPlain**

```
public boolean isPlain()
```

Returns true if the font is plain.
See Also:
getStyle

Interface Index **isBold**

```
public boolean isBold()
```

Returns true if the font is bold.
See Also:
getStyle

Interface Index **isItalic**

```
public boolean isItalic()
```

Returns true if the font is italic.
See Also:
getStyle

getFont

```
public static Font getFont(String nm)
```

Gets a font from the system properties list.

Parameters:

nm - the property name

getFont

```
public static Font getFont(String nm,
                           Font font)
```

Gets the specified font from the system properties list.

Parameters:

nm - the property name

font - a default font to return if property 'nm' is not defined

hashCode

```
public int hashCode()
```

Returns a hashcode for this font.

Overrides:

hashCode in class Object

equals

```
public boolean equals(Object obj)
```

Compares this object to the specifed object.

Parameters:

obj - the object to compare with

Returns:

true if the objects are the same; false otherwise.

Overrides:

equals in class Object

toString

```
public String toString()
```

Converts this object to a String representation.

Overrides:

toString in class Object

Class java.awt.FontMetrics

Class java.awt.FontMetrics

```
java.lang.Object
    |
    +----java.awt.FontMetrics
```

---

public class **FontMetrics**
extends Object

A font metrics object. Note that the implementations of these methods are inefficient, they are usually overridden with more efficient toolkit specific implementations.

font
The actual font.

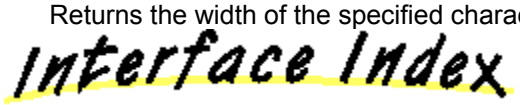
FontMetrics(Font)
Creates a new FontMetrics object with the specified font.

bytesWidth(byte[], int, int)
Returns the width of the specified array of bytes in this Font.

charWidth(int)
Returns the width of the specified character in this Font.

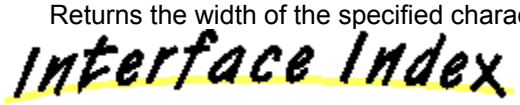
charWidth(char)

Returns the width of the specified character in this Font.



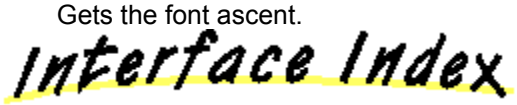
charsWidth(char[], int, int)

Returns the width of the specified character array in this Font.



getAscent()

Gets the font ascent.



getDescent()

Gets the font descent.

getFont()

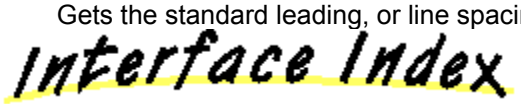
Gets the font.

getHeight()

Gets the total height of the font.

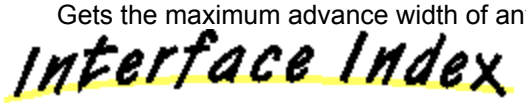
getLeading()

Gets the standard leading, or line spacing, for the font.



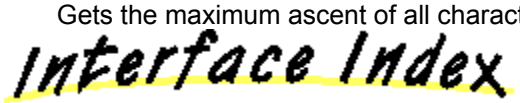
getMaxAdvance()

Gets the maximum advance width of any character in this Font.



getMaxAscent()

Gets the maximum ascent of all characters in this Font.

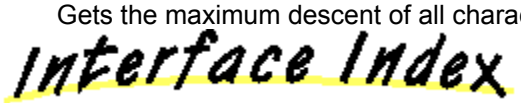


getMaxDecent()

For backward compatibility only.

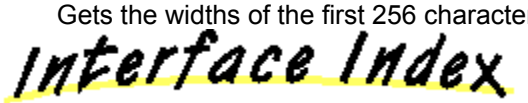
getMaxDescent()

Gets the maximum descent of all characters.



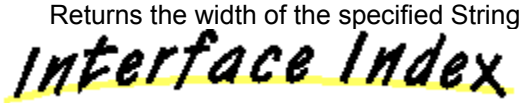
getWidths()

Gets the widths of the first 256 characters in the Font.



stringWidth(String)

Returns the width of the specified String in this Font.



toString()

Returns the String representation of this FontMetric's values.

font

```
protected Font font
```

The actual font.

See Also:

getFont

FontMetrics

```
protected FontMetrics(Font font)
```

Creates a new FontMetrics object with the specified font.

Parameters:

font - the font

See Also:

Font

getFont

```
public Font getFont()
```

Gets the font.

getLeading

```
public int getLeading()
```

Gets the standard leading, or line spacing, for the font. This is the logical amount of space to be reserved between the descent of one line of text and the ascent of the next line. The height metric is calculated to include this extra space.

getAscent

```
public int getAscent()
```

Gets the font ascent. The font ascent is the distance from the base line to the top of the characters.

See Also:

getMaxAscent

getDescent

```
public int getDescent()
```

Gets the font descent. The font descent is the distance from the base line to the bottom of the characters.

See Also:

getMaxDescent

getHeight

```
public int getHeight()
```

Gets the total height of the font. This is the distance between the baseline of adjacent lines of text. It is the sum of the leading + ascent + descent.

getMaxAscent

```
public int getMaxAscent()
```

Gets the maximum ascent of all characters in this Font. No character will extend further above the baseline than this metric.

See Also:

getAscent

getMaxDescent

```
public int getMaxDescent()
```

Gets the maximum descent of all characters. No character will descend futher below the baseline than this metric.

See Also:

getDescent

getMaxDecent

```
public int getMaxDecent()
```

For backward compatibility only.

See Also:

getMaxDescent

getMaxAdvance

```
public int getMaxAdvance()
```

Gets the maximum advance width of any character in this Font.

Returns:

-1 if the max advance is not known.

charWidth

```
public int charWidth(int ch)
```

Returns the width of the specified character in this Font.

Parameters:

ch - the specified font

See Also:

stringWidth

charWidth

```
public int charWidth(char ch)
```

Returns the width of the specified character in this Font.

Parameters:
ch - the specified font
See Also:
stringWidth

stringWidth

```
public int stringWidth(String str)
```

Returns the width of the specified String in this Font.
Parameters:
str - the String to be checked
See Also:
charsWidth, bytesWidth

charsWidth

```
public int charsWidth(char data[],
                      int off,
                      int len)
```

Returns the width of the specified character array in this Font.
Parameters:
data - the data to be checked
off - the start offset of the data
len - the maximum number of bytes checked
See Also:
stringWidth, bytesWidth

bytesWidth

```
public int bytesWidth(byte data[],
                      int off,
                      int len)
```

Returns the width of the specified array of bytes in this Font.
Parameters:
data - the data to be checked
off - the start offset of the data
len - the maximum number of bytes checked
See Also:
stringWidth, charsWidth

 getWidths

```
public int[] getWidths()
```

Gets the widths of the first 256 characters in the Font.

 toString

```
public String toString()
```

Returns the String representation of this FontMetric's values.
Overrides:
toString in class Object

Class java.awt.Frame

Class java.awt.Frame

```
java.lang.Object
    |
    +----java.awt.Component
            |
            +----java.awt.Container
                    |
                    +----java.awt.Window
                            |
                            +----java.awt.Frame
```

---

public class **Frame**
extends Window
implements MenuContainer

A Frame is a top-level window with a title. The default layout for a frame is BorderLayout.

CROSSHAIR_CURSOR

DEFAULT_CURSOR

E_RESIZE_CURSOR

HAND_CURSOR

MOVE_CURSOR

NE_RESIZE_CURSOR

NW_RESIZE_CURSOR

N_RESIZE_CURSOR

SE_RESIZE_CURSOR

SW_RESIZE_CURSOR

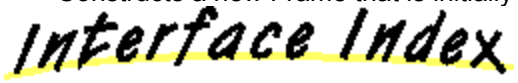
S_RESIZE_CURSOR

TEXT_CURSOR

WAIT_CURSOR

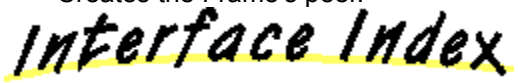
W_RESIZE_CURSOR

Frame()
Constructs a new Frame that is initially invisible.

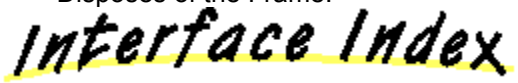


Frame(String)
Constructs a new, initially invisible Frame with the specified title.

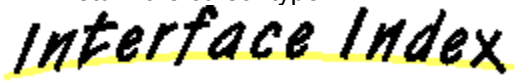
addNotify()
Creates the Frame's peer.



dispose()
Disposes of the Frame.



getCursorType()
Return the cursor type



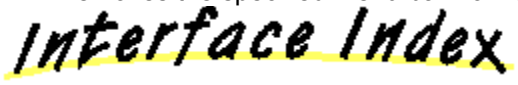
getIconImage()
Returns the icon image for this Frame.

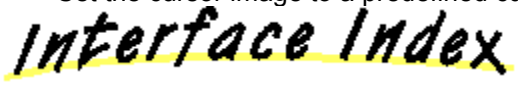
getMenuBar()
Gets the menu bar for this Frame.

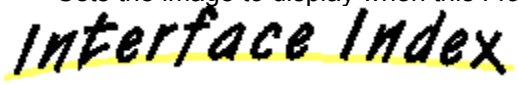
getTitle()
Gets the title of the Frame.


isResizable()
Returns true if the user can resize the Frame.

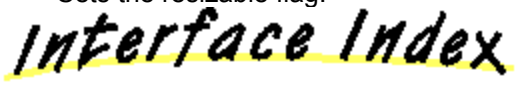

paramString()
Returns the parameter String of this Frame.


remove(MenuComponent)
Removes the specified menu bar from this Frame.


setCursor(int)
Set the cursor image to a predefined cursor.


setIconImage(Image)
Sets the image to display when this Frame is iconized.


setMenuBar(MenuBar)
Sets the menubar for this Frame to the specified menubar.


setResizable(boolean)
Sets the resizable flag.


setTitle(String)
Sets the title for this Frame to the specified title.

DEFAULT_CURSOR

```
public final static int DEFAULT_CURSOR
```

CROSSHAIR_CURSOR

```
public final static int CROSSHAIR_CURSOR
```

TEXT_CURSOR

```
public final static int TEXT_CURSOR
```


```
public final static int WAIT_CURSOR
```


```
public final static int SW_RESIZE_CURSOR
```


```
public final static int SE_RESIZE_CURSOR
```


```
public final static int NW_RESIZE_CURSOR
```


```
public final static int NE_RESIZE_CURSOR
```


```
public final static int N_RESIZE_CURSOR
```


```
public final static int S_RESIZE_CURSOR
```


```
public final static int W_RESIZE_CURSOR
```

Interface Index **E_RESIZE_CURSOR**

```
public final static int E_RESIZE_CURSOR
```

Interface Index **HAND_CURSOR**

```
public final static int HAND_CURSOR
```

Interface Index **MOVE_CURSOR**

```
public final static int MOVE_CURSOR
```

Interface Index

Interface Index **Frame**

```
public Frame()
```

Constructs a new Frame that is initially invisible.
See Also:
resize, show

Interface Index **Frame**

```
public Frame(String title)
```

Constructs a new, initially invisible Frame with the specified title.
Parameters:
title - te specified title
See Also:
resize, show

Interface Index

addNotify

```
public synchronized void addNotify()
```

Creates the Frame's peer. The peer allows us to change the look of the Frame without changing its functionality.

Overrides:

addNotify in class Window

getTitle

```
public String getTitle()
```

Gets the title of the Frame.

See Also:

setTitle

setTitle

```
public void setTitle(String title)
```

Sets the title for this Frame to the specified title.

Parameters:

title - the specified title of this Frame

See Also:

getTitle

getIconImage

```
public Image getIconImage()
```

Returns the icon image for this Frame.

setIconImage

```
public void setIconImage(Image image)
```

Sets the image to display when this Frame is iconized. Note that not all platforms support the concept of iconizing a window.

Parameters:

image - the icon image to be displayed

Interface Index

getMenuBar

```
public MenuBar getMenuBar()
```

Gets the menu bar for this Frame.

Interface Index

setMenuBar

```
public synchronized void setMenuBar(MenuBar mb)
```

Sets the menubar for this Frame to the specified menubar.

Parameters:

mb - the menubar being set

Interface Index

remove

```
public synchronized void remove(MenuComponent m)
```

Removes the specified menu bar from this Frame.

Interface Index

dispose

```
public synchronized void dispose()
```

Disposes of the Frame. This method must be called to release the resources that are used for the frame.

Overrides:

dispose in class Window

Interface Index

isResizable

```
public boolean isResizable()
```

Returns true if the user can resize the Frame.

setResizable

```
public void setResizable(boolean resizable)
```

Sets the resizable flag.
Parameters:
resizable - true if resizable; false otherwise.

setCursor

```
public void setCursor(int cursorType)
```

Set the cursor image to a predefined cursor.
Parameters:
cursorType - one of the cursor constants defined above.

getCursorType

```
public int getCursorType()
```

Return the cursor type

paramString

```
protected String paramString()
```

Returns the parameter String of this Frame.
Overrides:
paramString in class Container

Class java.awt.Graphics

Class java.awt.Graphics

```
java.lang.Object
    |
    +----java.awt.Graphics
```

---

public class **Graphics**
extends Object

Graphics is the abstract base class for all graphic contexts for various devices.

Graphics()

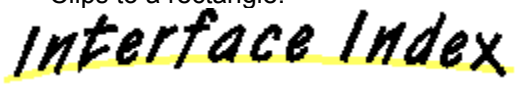
Constructs a new Graphics Object.

clearRect(int, int, int, int)

Clears the specified rectangle by filling it with the current background color of the current drawing surface.

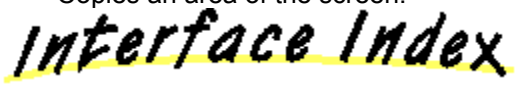
clipRect(int, int, int, int)

Clips to a rectangle.



copyArea(int, int, int, int, int, int)

Copies an area of the screen.

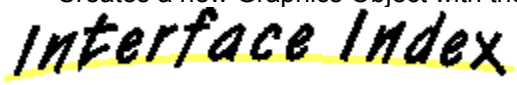


create()

Creates a new Graphics Object that is a copy of the original Graphics Object.

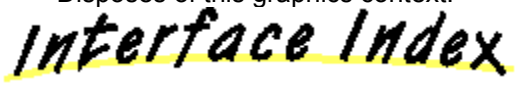
create(int, int, int, int)

Creates a new Graphics Object with the specified parameters, based on the original Graphics Object.



dispose()

Disposes of this graphics context.

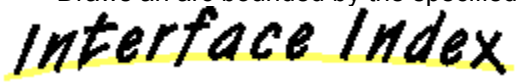


draw3DRect(int, int, int, int, boolean)

Draws a highlighted 3-D rectangle.

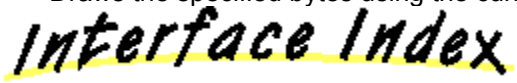
drawArc(int, int, int, int, int, int)

Draws an arc bounded by the specified rectangle from startAngle to endAngle.



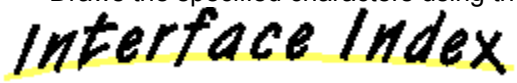
drawBytes(byte[], int, int, int, int)

Draws the specified bytes using the current font and color.



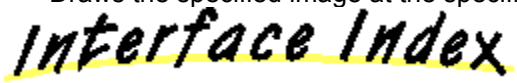
drawChars(char[], int, int, int, int)

Draws the specified characters using the current font and color.



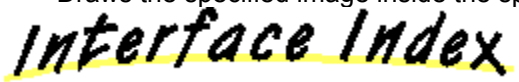
drawImage(Image, int, int, ImageObserver)

Draws the specified image at the specified coordinate (x, y).



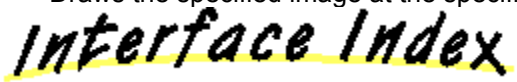
drawImage(Image, int, int, int, int, ImageObserver)

Draws the specified image inside the specified rectangle.



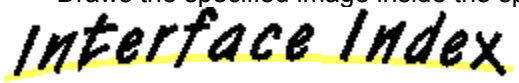
drawImage(Image, int, int, Color, ImageObserver)

Draws the specified image at the specified coordinate (x, y), with the given solid background Color.



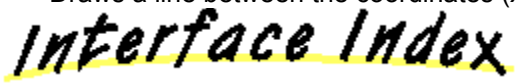
drawImage (Image, int, int, int, int, Color, ImageObserver)

Draws the specified image inside the specified rectangle, with the given solid background Color.



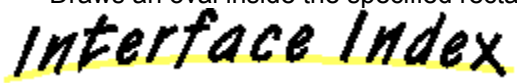
drawLine(int, int, int, int)

Draws a line between the coordinates (x1,y1) and (x2,y2).



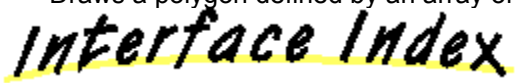
drawOval(int, int, int, int)

Draws an oval inside the specified rectangle using the current color.



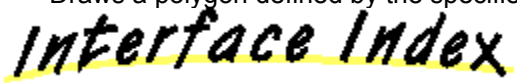
drawPolygon(int[], int[], int)

Draws a polygon defined by an array of x points and y points.



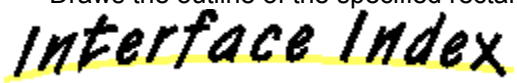
drawPolygon(Polygon)

Draws a polygon defined by the specified point.



drawRect(int, int, int, int)

Draws the outline of the specified rectangle using the current color.

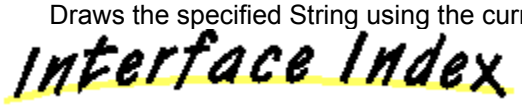


drawRoundRect(int, int, int, int, int, int)

Draws an outlined rounded corner rectangle using the current color.

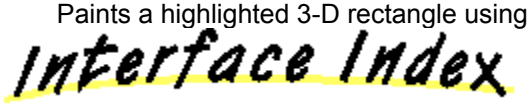
drawString(String, int, int)

Draws the specified String using the current font and color.



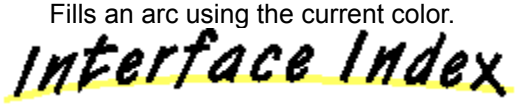
fill3DRect(int, int, int, int, boolean)

Paints a highlighted 3-D rectangle using the current color.



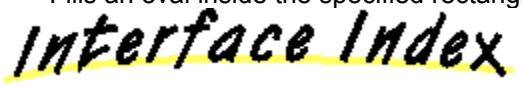
fillArc(int, int, int, int, int, int)

Fills an arc using the current color.



fillOval(int, int, int, int)

Fills an oval inside the specified rectangle using the current color.

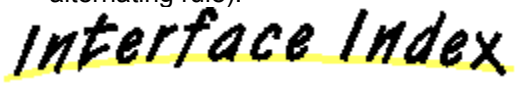


fillPolygon(int[], int[], int)

Fills a polygon with the current color using an even-odd fill rule (otherwise known as an alternating rule).

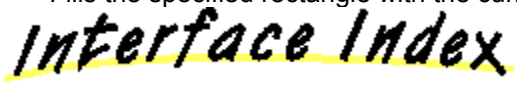
fillPolygon(Polygon)

Fills the specified polygon with the current color using an even-odd fill rule (otherwise known as an alternating rule).



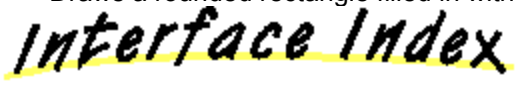
fillRect(int, int, int, int)

Fills the specified rectangle with the current color.



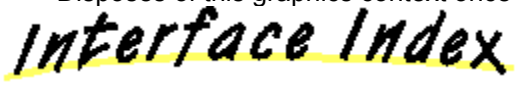
fillRoundRect(int, int, int, int, int, int)

Draws a rounded rectangle filled in with the current color.



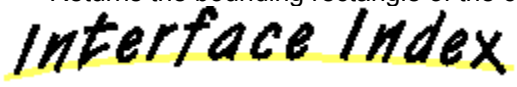
finalize()

Disposes of this graphics context once it is no longer referenced.



getClipRect()

Returns the bounding rectangle of the current clipping area.



getColor()

Gets the current color.

getFont()

Gets the current font.

getFontMetrics()

Gets the current font metrics.

getFontMetrics(Font)

Gets the current font metrics for the specified font.

Interface Index **setColor**(Color)
Sets the current color to the specified color.
Interface Index **setFont**(Font)
Sets the font for all subsequent text-drawing operations.
Interface Index **setPaintMode**()
Sets the paint mode to overwrite the destination with the current color.
Interface Index **setXORMode**(Color)
Sets the paint mode to alternate between the current color and the new specified color.
Interface Index **toString**()
Returns a String object representing this Graphic's value.
Interface Index **translate**(int, int)
Translates the specified parameters into the origin of the graphics context.

Interface Index

Interface Index **Graphics**

```
protected Graphics()
```

Constructs a new Graphics Object. Graphic contexts cannot be created directly. They must be obtained from another graphics context or created by a Component.
See Also:
getGraphics, create

Interface Index

Interface Index **create**

```
public abstract Graphics create()
```

Creates a new Graphics Object that is a copy of the original Graphics Object.

create

```
public Graphics create(int x,
                       int y,
                       int width,
                       int height)
```

Creates a new Graphics Object with the specified parameters, based on the original Graphics Object. This method translates the specified parameters, x and y, to the proper origin coordinates and then clips the Graphics Object to the area.

Parameters:

x - the x coordinate
y - the y coordinate
width - the width of the area
height - the height of the area

See Also:

translate

translate

```
public abstract void translate(int x,
                               int y)
```

Translates the specified parameters into the origin of the graphics context. All subsequent operations on this graphics context will be relative to this origin.

Parameters:

x - the x coordinate
y - the y coordinate

getColor

```
public abstract Color getColor()
```

Gets the current color.

See Also:

setColor

setColor

```
public abstract void setColor(Color c)
```

Sets the current color to the specified color. All subsequent graphics operations will use this specified color.

Parameters:

c - the color to be set

See Also:

Color, getColor

setPaintMode

```
public abstract void setPaintMode()
```

Sets the paint mode to overwrite the destination with the current color.

setXORMode

```
public abstract void setXORMode(Color c1)
```

Sets the paint mode to alternate between the current color and the new specified color. When drawing operations are performed, pixels which are the current color will be changed to the specified color and vice versa. Pixels of colors other than those two colors will be changed in an unpredictable, but reversible manner - if you draw the same figure twice then all pixels will be restored to their original values.

Parameters:

c1 - the second color

getFont

```
public abstract Font getFont()
```

Gets the current font.

See Also:

setFont

setFont

```
public abstract void setFont(Font font)
```

Sets the font for all subsequent text-drawing operations.
Parameters:
font - the specified font
See Also:
Font, getFont, drawString, drawBytes, drawChars

getFontMetrics

```
public FontMetrics getFontMetrics()
```

Gets the current font metrics.
See Also:
getFont

getFontMetrics

```
public abstract FontMetrics getFontMetrics(Font f)
```

Gets the current font metrics for the specified font.
Parameters:
f - the specified font
See Also:
getFont, getFontMetrics

getClipRect

```
public abstract Rectangle getClipRect()
```

Returns the bounding rectangle of the current clipping area.
See Also:
clipRect

clipRect

```
public abstract void clipRect(int x,
                              int y,
                              int width,
                              int height)
```

Clips to a rectangle. The resulting clipping area is the intersection of the current clipping area and the specified rectangle. Graphic operations have no effect outside of the clipping area.

Parameters:

x - the x coordinate
y - the y coordinate
width - the width of the rectangle
height - the height of the rectangle

See Also:

getClipRect

copyArea

```
public abstract void copyArea(int x,
                              int y,
                              int width,
                              int height,
                              int dx,
                              int dy)
```

Copies an area of the screen.

Parameters:

x - the x-coordinate of the source
y - the y-coordinate of the source
width - the width
height - the height
dx - the horizontal distance
dy - the vertical distance

drawLine

```
public abstract void drawLine(int x1,
                              int y1,
                              int x2,
                              int y2)
```

Draws a line between the coordinates (x1,y1) and (x2,y2). The line is drawn below and to the left of the logical coordinates.

Parameters:

x1 - the first point's x coordinate
y1 - the first point's y coordinate
x2 - the second point's x coordinate
y2 - the second point's y coordinate

Interface Index **fillRect**

```
public abstract void fillRect(int x,
                              int y,
                              int width,
                              int height)
```

Fills the specified rectangle with the current color.

Parameters:

x - the x coordinate
y - the y coordinate
width - the width of the rectangle
height - the height of the rectangle

See Also:

drawRect, clearRect

Interface Index **drawRect**

```
public void drawRect(int x,
                     int y,
                     int width,
                     int height)
```

Draws the outline of the specified rectangle using the current color. Use drawRect(x, y, width-1, height-1) to draw the outline inside the specified rectangle.

Parameters:

x - the x coordinate
y - the y coordinate
width - the width of the rectangle
height - the height of the rectangle

See Also:

fillRect, clearRect

Interface Index **clearRect**

```
public abstract void clearRect(int x,
                               int y,
                               int width,
                               int height)
```

Clears the specified rectangle by filling it with the current background color of the current drawing surface. Which drawing surface it selects depends on how the graphics context was created.

Parameters:

x - the x coordinate

y - the y coordinate
width - the width of the rectangle
height - the height of the rectangle

See Also:

fillRect, drawRect

drawRoundRect

```
public abstract void drawRoundRect(int x,
                                   int y,
                                   int width,
                                   int height,
                                   int arcWidth,
                                   int arcHeight)
```

Draws an outlined rounded corner rectangle using the current color.

Parameters:

x - the x coordinate
y - the y coordinate
width - the width of the rectangle
height - the height of the rectangle
arcWidth - the horizontal diameter of the arc at the four corners
arcHeight - the horizontal diameter of the arc at the four corners

See Also:

fillRoundRect

fillRoundRect

```
public abstract void fillRoundRect(int x,
                                   int y,
                                   int width,
                                   int height,
                                   int arcWidth,
                                   int arcHeight)
```

Draws a rounded rectangle filled in with the current color.

Parameters:

x - the x coordinate
y - the y coordinate
width - the width of the rectangle
height - the height of the rectangle
arcWidth - the horizontal diameter of the arc at the four corners
arcHeight - the horizontal diameter of the arc at the four corners

See Also:

drawRoundRect

Interface Index **draw3DRect**

```
public void draw3DRect(int x,
                       int y,
                       int width,
                       int height,
                       boolean raised)
```

Draws a highlighted 3-D rectangle.
Parameters:
x - the x coordinate
y - the y coordinate
width - the width of the rectangle
height - the height of the rectangle
raised - a boolean that states whether the rectangle is raised or not

Interface Index **fill3DRect**

```
public void fill3DRect(int x,
                       int y,
                       int width,
                       int height,
                       boolean raised)
```

Paints a highlighted 3-D rectangle using the current color.
Parameters:
x - the x coordinate
y - the y coordinate
width - the width of the rectangle
height - the height of the rectangle
raised - a boolean that states whether the rectangle is raised or not

Interface Index **drawOval**

```
public abstract void drawOval(int x,
                              int y,
                              int width,
                              int height)
```

Draws an oval inside the specified rectangle using the current color.
Parameters:
x - the x coordinate
y - the y coordinate

width - the width of the rectangle
height - the height of the rectangle

See Also:

fillOval

Interface Index **fillOval**

```
public abstract void fillOval(int x,
                              int y,
                              int width,
                              int height)
```

Fills an oval inside the specified rectangle using the current color.

Parameters:

x - the x coordinate
y - the y coordinate
width - the width of the rectangle
height - the height of the rectangle

See Also:

drawOval

Interface Index **drawArc**

```
public abstract void drawArc(int x,
                             int y,
                             int width,
                             int height,
                             int startAngle,
                             int arcAngle)
```

Draws an arc bounded by the specified rectangle from startAngle to endAngle. 0 degrees is at the 3-o'clock position.Positive arc angles indicate counter-clockwise rotations, negative arc angles are drawn clockwise.

Parameters:

x - the x coordinate
y - the y coordinate
width - the width of the rectangle
height - the height of the rectangle
startAngle - the beginning angle
arcAngle - the angle of the arc (relative to startAngle).

See Also:

fillArc

Interface Index **fillArc**

```
public abstract void fillArc(int x,
                             int y,
                             int width,
                             int height,
                             int startAngle,
                             int arcAngle)
```

Fills an arc using the current color. This generates a pie shape.

Parameters:

x - the x coordinate
y - the y coordinate
width - the width of the arc
height - the height of the arc
startAngle - the beginning angle
arcAngle - the angle of the arc (relative to startAngle).

See Also:

drawArc

Interface Index **drawPolygon**

```
public abstract void drawPolygon(int xPoints[],
                                 int yPoints[],
                                 int nPoints)
```

Draws a polygon defined by an array of x points and y points.

Parameters:

xPoints - an array of x points
yPoints - an array of y points
nPoints - the total number of points

See Also:

fillPolygon

Interface Index **drawPolygon**

```
public void drawPolygon(Polygon p)
```

Draws a polygon defined by the specified point.

Parameters:

p - the specified polygon

See Also:

fillPolygon

Interface Index **fillPolygon**

```
public abstract void fillPolygon(int xPoints[],
                                 int yPoints[],
                                 int nPoints)
```

Fills a polygon with the current color using an even-odd fill rule (otherwise known as an alternating rule).

Parameters:

xPoints - an array of x points
yPoints - an array of y points
nPoints - the total number of points

See Also:

drawPolygon

Interface Index **fillPolygon**

```
public void fillPolygon(Polygon p)
```

Fills the specified polygon with the current color using an even-odd fill rule (otherwise known as an alternating rule).

Parameters:

p - the polygon

See Also:

drawPolygon

Interface Index **drawString**

```
public abstract void drawString(String str,
                                int x,
                                int y)
```

Draws the specified String using the current font and color. The x,y position is the starting point of the baseline of the String.

Parameters:

str - the String to be drawn
x - the x coordinate
y - the y coordinate

See Also:

drawChars, drawBytes

Interface Index **drawChars**

```
public void drawChars(char data[],
                      int offset,
                      int length,
                      int x,
                      int y)
```

Draws the specified characters using the current font and color.

Parameters:

data - the array of characters to be drawn
offset - the start offset in the data
length - the number of characters to be drawn
x - the x coordinate
y - the y coordinate

See Also:

drawString, drawBytes

Interface Index **drawBytes**

```
public void drawBytes(byte data[],
                      int offset,
                      int length,
                      int x,
                      int y)
```

Draws the specified bytes using the current font and color.

Parameters:

data - the data to be drawn
offset - the start offset in the data
length - the number of bytes that are drawn
x - the x coordinate
y - the y coordinate

See Also:

drawString, drawChars

Interface Index **drawImage**

```
public abstract boolean drawImage(Image img,
                                  int x,
                                  int y,
                                  ImageObserver observer)
```

Draws the specified image at the specified coordinate (x, y). If the image is incomplete the image observer will be notified later.

Parameters:

img - the specified image to be drawn
x - the x coordinate
y - the y coordinate
observer - notifies if the image is complete or not

See Also:

Image, ImageObserver

drawImage

```
public abstract boolean drawImage(Image img,
                                  int x,
                                  int y,
                                  int width,
                                  int height,
                                  ImageObserver observer)
```

Draws the specified image inside the specified rectangle. The image is scaled if necessary. If the image is incomplete the image observer will be notified later.

Parameters:

img - the specified image to be drawn
x - the x coordinate
y - the y coordinate
width - the width of the rectangle
height - the height of the rectangle
observer - notifies if the image is complete or not

See Also:

Image, ImageObserver

drawImage

```
public abstract boolean drawImage(Image img,
                                  int x,
                                  int y,
                                  Color bgcolor,
                                  ImageObserver observer)
```

Draws the specified image at the specified coordinate (x, y), with the given solid background Color. If the image is incomplete the image observer will be notified later.

Parameters:

img - the specified image to be drawn
x - the x coordinate
y - the y coordinate
observer - notifies if the image is complete or not

See Also:

Image, ImageObserver

drawImage

```
public abstract boolean drawImage(Image img,
                                  int x,
                                  int y,
                                  int width,
                                  int height,
                                  Color bgcolor,
                                  ImageObserver observer)
```

Draws the specified image inside the specified rectangle, with the given solid background Color. The image is scaled if necessary. If the image is incomplete the image observer will be notified later.

Parameters:

img - the specified image to be drawn
x - the x coordinate
y - the y coordinate
width - the width of the rectangle
height - the height of the rectangle
observer - notifies if the image is complete or not

See Also:

Image, ImageObserver

dispose

```
public abstract void dispose()
```

Disposes of this graphics context. The Graphics context cannot be used after being disposed of.

See Also:

finalize

finalize

```
public void finalize()
```

Disposes of this graphics context once it is no longer referenced.

Overrides:

finalize in class Object

See Also:

dispose

toString

```
public String toString()
```

Returns a String object representing this Graphic's value.

Overrides:

toString in class Object

Class java.awt.GridBagConstraints

Class java.awt.GridBagConstraints

```
java.lang.Object
    |
    +----java.awt.GridBagConstraints
```

---

public class **GridBagConstraints**
extends Object
implements Cloneable

GridBagConstraints is used to specify constraints for components laid out using the GridBagLayout class.
See Also:
GridBagLayout

BOTH

CENTER

EAST

HORIZONTAL

NONE

NORTH

NORTHEAST

NORTHWEST

RELATIVE

clone()

Creates a clone of the object.


RELATIVE

```
public final static int RELATIVE
```


REMAINDER

```
public final static int REMAINDER
```


NONE

```
public final static int NONE
```


BOTH

```
public final static int BOTH
```


HORIZONTAL

```
public final static int HORIZONTAL
```


VERTICAL

```
public final static int VERTICAL
```

CENTER

```
public final static int CENTER
```

NORTH

```
public final static int NORTH
```

NORTHEAST

```
public final static int NORTHEAST
```

EAST

```
public final static int EAST
```

SOUTHEAST

```
public final static int SOUTHEAST
```

SOUTH

```
public final static int SOUTH
```

SOUTHWEST

```
public final static int SOUTHWEST
```

WEST

```
public final static int WEST
```

NORTHWEST

```
public final static int NORTHWEST
```


```
public int gridx
```


```
public int gridy
```


```
public int gridwidth
```


```
public int gridheight
```


```
public double weightx
```


```
public double weighty
```


```
public int anchor
```


```
public int fill
```

insets

```
public Insets insets
```

ipadx

```
public int ipadx
```

ipady

```
public int ipady
```

GridBagConstraints

```
public GridBagConstraints()
```

clone

```
public Object clone()
```

Creates a clone of the object.

Overrides:

clone in class Object

Class java.awt.GridBagLayout

Class java.awt.GridBagLayout

```
java.lang.Object
    |
    +----java.awt.GridBagLayout
```

---

public class **GridBagLayout**
extends Object
implements LayoutManager

GridBagLayout is a flexible layout manager that aligns components vertically and horizontally, without requiring that the components be the same size. Each GridBagLayout uses a rectangular grid of cells, with each component occupying one or more cells (called its *display area*). Each component managed by a GridBagLayout is associated with a GridBagConstraints instance that specifies how the component is laid out within its display area. How a GridBagLayout places a set of components depends on each component's GridBagConstraints and minimum size, as well as the preferred size of the components' container.

To use a GridBagLayout effectively, you must customize one or more of its components' GridBagConstraints. You customize a GridBagConstraints object by setting one or more of its instance variables:

gridx, gridy
: Specifies the cell at the upper left of the component's display area, where the upper-left-most cell has address gridx=0, gridy=0. Use GridBagConstraints.RELATIVE (the default value) to specify that the component be placed just to the right of (for gridx) or just below (for gridy) the component that was added to the container just before this component was added.

gridwidth, gridheight
: Specifies the number of cells in a row (for gridwidth) or column (for gridheight) in the component's display area. The default value is 1. Use GridBagConstraints.REMAINDER to specify that the component be the last one in its row (for gridwidth) or column (for gridheight). Use GridBagConstraints.RELATIVE to specify that the component be the next to last one in its row (for gridwidth) or column (for gridheight).

fill
: Used when the component's display area is larger than the component's requested size to determine whether (and how) to resize the component. Valid values are GridBagConstraint.NONE (the default), GridBagConstraint.HORIZONTAL (make the component wide enough to fill its display area horizontally, but don't change its height), GridBagConstraint.VERTICAL (make the component tall enough to fill its display area vertically, but don't change its width), and GridBagConstraint.BOTH (make the component fill its display area entirely).

ipadx, ipady
: Specifies the internal padding: how much to add to the minimum size of the component. The width of the component will be at least its minimum width plus ipadx*2 pixels (since

the padding applies to both sides of the component). Similarly, the height of the component will be at least the minimum height plus ipady*2 pixels.

insets

Specifies the external padding of the component -- the minimum amount of space between the component and the edges of its display area.

anchor

Used when the component is smaller than its display area to determine where (within the area) to place the component. Valid values are GridBagConstraints.CENTER (the default), GridBagConstraints.NORTH, GridBagConstraints.NORTHEAST, GridBagConstraints.EAST, GridBagConstraints.SOUTHEAST, GridBagConstraints.SOUTH, GridBagConstraints.SOUTHWEST, GridBagConstraints.WEST, and GridBagConstraints.NORTHWEST.

weightx, weighty

Used to determine how to distribute space; this is important for specifying resizing behavior. Unless you specify a weight for at least one component in a row (weightx) and column (weighty), all the components clump together in the center of their container. This is because when the weight is zero (the default), the GridBagLayout puts any extra space between its grid of cells and the edges of the container.

The following figure shows ten components (all buttons) managed by a GridBagLayout:

All the components have fill=GridBagConstraints.BOTH. In addition, the components have the following non-default constraints:

- Button1, Button2, Button3: weightx=1.0
- Button4: weightx=1.0, gridwidth=GridBagConstraints.REMAINDER
- Button5: gridwidth=GridBagConstraints.REMAINDER
- Button6: gridwidth=GridBagConstraints.RELATIVE
- Button7: gridwidth=GridBagConstraints.REMAINDER
- Button8: gridheight=2, weighty=1.0,
- Button9, Button 10: gridwidth=GridBagConstraints.REMAINDER

Here is the code that implements the example shown above:

```
import java.awt.*;
import java.util.*;
import java.applet.Applet;
public class GridBagEx1 extends Applet {
    protected void makebutton(String name,
                              GridBagLayout gridbag,
                              GridBagConstraints c) {
        Button button = new Button(name);
        gridbag.setConstraints(button, c);
        add(button);
    }
    public void init() {
        GridBagLayout gridbag = new GridBagLayout();
        GridBagConstraints c = new GridBagConstraints();
        setFont(new Font("Helvetica", Font.PLAIN, 14));
        setLayout(gridbag);
        c.fill = GridBagConstraints.BOTH;
        c.weightx = 1.0;
```

```
        makebutton("Button1", gridbag, c);
        makebutton("Button2", gridbag, c);
        makebutton("Button3", gridbag, c);
        c.gridwidth = GridBagConstraints.REMAINDER; //end row
        makebutton("Button4", gridbag, c);
        c.weightx = 0.0;                    //reset to the default
        makebutton("Button5", gridbag, c); //another row
        c.gridwidth = GridBagConstraints.RELATIVE; //next-to-last in row
        makebutton("Button6", gridbag, c);
        c.gridwidth = GridBagConstraints.REMAINDER; //end row
        makebutton("Button7", gridbag, c);
        c.gridwidth = 1;                    //reset to the default
        c.gridheight = 2;
        c.weighty = 1.0;
        makebutton("Button8", gridbag, c);
        c.weighty = 0.0;                    //reset to the default
        c.gridwidth = GridBagConstraints.REMAINDER; //end row
        c.gridheight = 1;                   //reset to the default
        makebutton("Button9", gridbag, c);
        makebutton("Button10", gridbag, c);
        resize(300, 100);
    }
    public static void main(String args[]) {
        Frame f = new Frame("GridBag Layout Example");
        GridBagEx1 ex1 = new GridBagEx1();
        ex1.init();
        f.add("Center", ex1);
        f.pack();
        f.resize(f.preferredSize());
        f.show();
    }
}
```

Interface Index

Interface Index

Interface Index

Interface Index

Interface Index

Interface Index

Interface Index

MAXGRIDSIZE

MINSIZE

PREFERREDSIZE

columnWeights

columnWidths

comptable

defaultConstraints

layoutInfo

rowHeights

rowWeights

GridBagLayout()

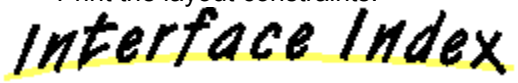
Creates a gridbag layout.

AdjustForGravity(GridBagConstraints, Rectangle)

ArrangeGrid(Container)

DumpConstraints (GridBagConstraints)

Print the layout constraints.



DumpLayoutInfo(GridBagLayoutInfo)

Print the layout information.

GetLayoutInfo(Container, int)

GetMinSize (Container, GridBagLayoutInfo)

addLayoutComponent (String, Component)

Adds the specified component with the specified name to the layout.

getConstraints(Component)

Retrieves the constraints for the specified component.

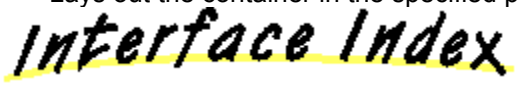
getLayoutDimensions()

getLayoutOrigin()

getLayoutWeights()

layoutContainer(Container)

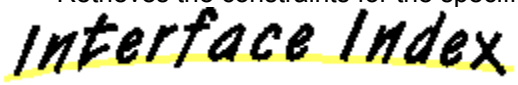
Lays out the container in the specified panel.



location(int, int)

lookupConstraints(Component)

Retrieves the constraints for the specified component.

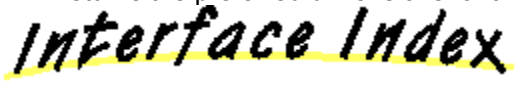


minimumLayoutSize(Container)

Returns the minimum dimensions needed to layout the components contained in the specified panel.

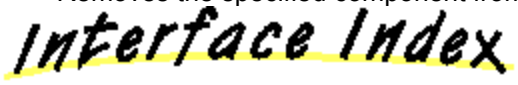
preferredLayoutSize(Container)

Returns the preferred dimensions for this layout given the components in the specified panel.



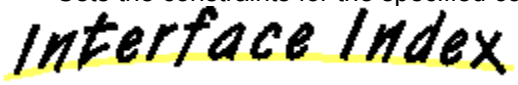
removeLayoutComponent(Component)

Removes the specified component from the layout.



setConstraints(Component, GridBagConstraints)

Sets the constraints for the specified component.



toString()

Returns the String representation of this GridLayout's values.

MAXGRIDSIZE

```
protected final static int MAXGRIDSIZE
```

MINSIZE

```
protected final static int MINSIZE
```

PREFERREDSIZE

```
protected final static int PREFERREDSIZE
```

Interface Index comptable

```
protected Hashtable comptable
```

Interface Index defaultConstraints

```
protected GridBagConstraints defaultConstraints
```

Interface Index layoutInfo

```
protected GridBagLayoutInfo layoutInfo
```

Interface Index columnWidths

```
public int columnWidths[]
```

Interface Index rowHeights

```
public int rowHeights[]
```

Interface Index columnWeights

```
public double columnWeights[]
```

Interface Index rowWeights

```
public double rowWeights[]
```

Interface Index

Interface Index GridBagLayout

```
public GridBagLayout()
```

Creates a gridbag layout.

setConstraints

```
public void setConstraints(Component comp,
                           GridBagConstraints constraints)
```

Sets the constraints for the specified component.
Parameters:
comp - the component to be modified
constraints - the constraints to be applied

getConstraints

```
public GridBagConstraints getConstraints(Component comp)
```

Retrieves the constraints for the specified component. A copy of the constraints is returned.
Parameters:
comp - the component to be queried

lookupConstraints

```
protected GridBagConstraints lookupConstraints(Component comp)
```

Retrieves the constraints for the specified component. The return value is not a copy, but is the actual constraints class used by the layout mechanism.
Parameters:
comp - the component to be queried

getLayoutOrigin

```
public Point getLayoutOrigin()
```

getLayoutDimensions

```
public int[][] getLayoutDimensions()
```

getLayoutWeights

```
public double[][] getLayoutWeights()
```

location

```
public Point location(int x,
                      int y)
```

addLayoutComponent

```
public void addLayoutComponent(String name,
                               Component comp)
```

Adds the specified component with the specified name to the layout.
Parameters:
name - the name of the component
comp - the component to be added

removeLayoutComponent

```
public void removeLayoutComponent(Component comp)
```

Removes the specified component from the layout. Does not apply.
Parameters:
comp - the component to be removed

preferredLayoutSize

```
public Dimension preferredLayoutSize(Container parent)
```

Returns the preferred dimensions for this layout given the components in the specified panel.

Parameters:

parent - the component which needs to be laid out

See Also:

minimumLayoutSize

minimumLayoutSize

```
public Dimension minimumLayoutSize(Container parent)
```

Returns the minimum dimensions needed to layout the components contained in the specified panel.

Parameters:

parent - the component which needs to be laid out

See Also:

preferredLayoutSize

layoutContainer

```
public void layoutContainer(Container parent)
```

Lays out the container in the specified panel.

Parameters:

parent - the specified component being laid out

See Also:

Container

toString

```
public String toString()
```

Returns the String representation of this GridLayout's values.

Overrides:

toString in class Object

DumpLayoutInfo

```
protected void DumpLayoutInfo(GridBagLayoutInfo s)
```

Print the layout information. Useful for debugging.

DumpConstraints

```
protected void DumpConstraints(GridBagConstraints constraints)
```

Print the layout constraints. Useful for debugging.

GetLayoutInfo

```
protected GridBagLayoutInfo GetLayoutInfo(Container parent,
                                          int sizeflag)
```

AdjustForGravity

```
protected void AdjustForGravity(GridBagConstraints constraints,
                                Rectangle r)
```

GetMinSize

```
protected Dimension GetMinSize(Container parent,
                               GridBagLayoutInfo info)
```

ArrangeGrid

```
protected void ArrangeGrid(Container parent)
```

Class java.awt.GridLayout

Class java.awt.GridLayout

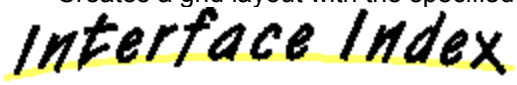
```
java.lang.Object
    |
    +----java.awt.GridLayout
```

---

public class **GridLayout**
extends Object
implements LayoutManager

A layout manager for a container that lays out grids.

GridLayout(int, int)
Creates a grid layout with the specified rows and columns.



GridLayout(int, int, int, int)
Creates a grid layout with the specified rows, columns, horizontal gap, and vertical gap.

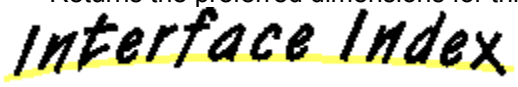
addLayoutComponent(String, Component)
Adds the specified component with the specified name to the layout.

layoutContainer(Container)
Lays out the container in the specified panel.

minimumLayoutSize(Container)
Returns the minimum dimensions needed to layout the components contained in the specified panel.

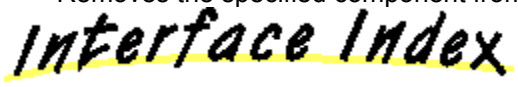
preferredLayoutSize(Container)

Returns the preferred dimensions for this layout given the components int the specified panel.



removeLayoutComponent(Component)

Removes the specified component from the layout.



toString()

Returns the String representation of this GridLayout's values.

GridLayout

```
public GridLayout(int rows,
                  int cols)
```

Creates a grid layout with the specified rows and columns.

Parameters:

rows - the rows

cols - the columns

GridLayout

```
public GridLayout(int rows,
                  int cols,
                  int hgap,
                  int vgap)
```

Creates a grid layout with the specified rows, columns, horizontal gap, and vertical gap.

Parameters:

rows - the rows; zero means 'any number.'

cols - the columns; zero means 'any number.' Only one of 'rows' and 'cols' can be zero, not both.

hgap - the horizontal gap variable

vgap - the vertical gap variable

Throws: IllegalArgumentException

If the rows and columns are invalid.

addLayoutComponent

```
public void addLayoutComponent(String name,
                               Component comp)
```

Adds the specified component with the specified name to the layout.

Parameters:

name - the name of the component
comp - the component to be added

removeLayoutComponent

```
public void removeLayoutComponent(Component comp)
```

Removes the specified component from the layout. Does not apply.

Parameters:

comp - the component to be removed

preferredLayoutSize

```
public Dimension preferredLayoutSize(Container parent)
```

Returns the preferred dimensions for this layout given the components int the specified panel.

Parameters:

parent - the component which needs to be laid out

See Also:

minimumLayoutSize

minimumLayoutSize

```
public Dimension minimumLayoutSize(Container parent)
```

Returns the minimum dimensions needed to layout the components contained in the specified panel.

Parameters:

parent - the component which needs to be laid out

See Also:

preferredLayoutSize

layoutContainer

```
public void layoutContainer(Container parent)
```

Lays out the container in the specified panel.

Parameters:

parent - the specified component being laid out

See Also:

Container

toString

```
public String toString()
```

Returns the String representation of this GridLayout's values.

Overrides:

toString in class Object

Class java.awt.image.ColorModel

Class java.awt.image.ColorModel

```
java.lang.Object
    |
    +----java.awt.image.ColorModel
```

---

public class **ColorModel**
extends Object

A class that encapsulates the methods for translating from pixel values to alpha, red, green, and blue color components for an image. This class is abstract.

See Also:

IndexColorModel, DirectColorModel

pixel_bits

ColorModel(int)

Constructs a ColorModel which describes a pixel of the specified number of bits.

getAlpha(int)

The subclass must provide a function which provides the alpha color compoment for the specified pixel.

getBlue(int)

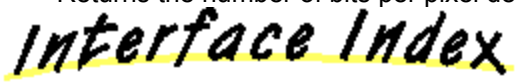
The subclass must provide a function which provides the blue color compoment for the specified pixel.

getGreen(int)

The subclass must provide a function which provides the green color compoment for the specified pixel.

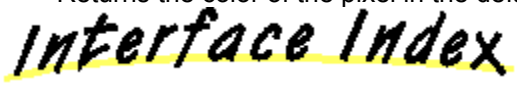
getPixelSize()

Returns the number of bits per pixel described by this ColorModel.



getRGB(int)

Returns the color of the pixel in the default RGB color model.



getRGBdefault()

Return a ColorModel which describes the default format for integer RGB values used throughout the AWT image interfaces.

getRed(int)

The subclass must provide a function which provides the red color compoment for the specified pixel.

pixel_bits

```
protected int pixel_bits
```

ColorModel

```
public ColorModel(int bits)
```

Constructs a ColorModel which describes a pixel of the specified number of bits.

 getRGBdefault

```
public static ColorModel getRGBdefault()
```

Return a ColorModel which describes the default format for integer RGB values used throughout the AWT image interfaces. The format for the RGB values is an integer with 8 bits each of alpha, red, green, and blue color components ordered correspondingly from the most significant byte to the least significant byte, as in: 0xAARRGGBB

 getPixelSize

```
public int getPixelSize()
```

Returns the number of bits per pixel described by this ColorModel.

 getRed

```
public abstract int getRed(int pixel)
```

The subclass must provide a function which provides the red color compoment for the specified pixel.
Returns:
The red color component ranging from 0 to 255

 getGreen

```
public abstract int getGreen(int pixel)
```

The subclass must provide a function which provides the green color compoment for the specified pixel.
Returns:
The green color component ranging from 0 to 255

 getBlue

```
public abstract int getBlue(int pixel)
```

The subclass must provide a function which provides the blue color compoment for the specified pixel.
Returns:

The blue color component ranging from 0 to 255

getAlpha

```
public abstract int getAlpha(int pixel)
```

The subclass must provide a function which provides the alpha color compoment for the specified pixel.

Returns:

The alpha transparency value ranging from 0 to 255

getRGB

```
public int getRGB(int pixel)
```

Returns the color of the pixel in the default RGB color model.

See Also:

getRGBdefault

Class java.awt.image.CropImageFilter

Class java.awt.image.CropImageFilter

```
java.lang.Object
    |
    +----java.awt.image.ImageFilter
              |
              +----java.awt.image.CropImageFilter
```

---

public class **CropImageFilter**
extends ImageFilter

An ImageFilter class for cropping images. This class extends the basic ImageFilter Class to extract a given rectangular region of an existing Image and provide a source for a new image containing just the extracted region. It is meant to be used in conjunction with a FilteredImageSource object to produce cropped versions of existing images.

See Also:

FilteredImageSource, ImageFilter

CropImageFilter(int, int, int, int)
Constructs a CropImageFilter that extracts the absolute rectangular region of pixels from its source Image as specified by the x, y, w, and h parameters.

setDimensions(int, int)
Override the source image's dimensions and pass the dimensions of the rectangular cropped region to the ImageConsumer.

setPixels (int, int, int, int, ColorModel, byte[], int, int)
Determine whether the delivered byte pixels intersect the region to be extracted and passes through only that subset of pixels that appear in the output region.

Interface Index

setPixels (int, int, int, int, ColorModel, int[], int, int)

Determine if the delivered int pixels intersect the region to be extracted and pass through only that subset of pixels that appear in the output region.

Interface Index

setProperties(Hashtable)

Passes along the properties from the source object after adding a property indicating the cropped region.

Interface Index

Interface Index

CropImageFilter

```
public CropImageFilter(int x,
                       int y,
                       int w,
                       int h)
```

Constructs a CropImageFilter that extracts the absolute rectangular region of pixels from its source Image as specified by the x, y, w, and h parameters.

Parameters:

x - the x location of the top of the rectangle to be extracted
y - the y location of the top of the rectangle to be extracted
w - the width of the rectangle to be extracted
h - the height of the rectangle to be extracted

Interface Index

Interface Index

setProperties

```
public void setProperties(Hashtable props)
```

Passes along the properties from the source object after adding a property indicating the cropped region.

Overrides:

setProperties in class ImageFilter

Interface Index

setDimensions

```
public void setDimensions(int w,
                          int h)
```

Override the source image's dimensions and pass the dimensions of the rectangular cropped region to the ImageConsumer.

Overrides:

setDimensions in class ImageFilter

See Also:

ImageConsumer

setPixels

```
public void setPixels(int x,
                      int y,
                      int w,
                      int h,
                      ColorModel model,
                      byte pixels[],
                      int off,
                      int scansize)
```

Determine whether the delivered byte pixels intersect the region to be extracted and passes through only that subset of pixels that appear in the output region.

Overrides:

setPixels in class ImageFilter

setPixels

```
public void setPixels(int x,
                      int y,
                      int w,
                      int h,
                      ColorModel model,
                      int pixels[],
                      int off,
                      int scansize)
```

Determine if the delivered int pixels intersect the region to be extracted and pass through only that subset of pixels that appear in the output region.

Overrides:

setPixels in class ImageFilter

Class java.awt.image.DirectColorModel

Class java.awt.image.DirectColorModel

```
java.lang.Object
    |
    +----java.awt.image.ColorModel
              |
              +----java.awt.image.DirectColorModel
```

---

public class **DirectColorModel**
extends ColorModel

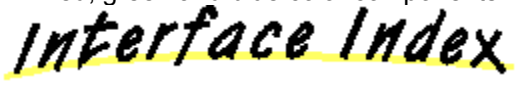
A ColorModel class that specifies a translation from pixel values to alpha, red, green, and blue color components for pixels which have the color components embedded directly in the bits of the pixel itself. This color model is similar to an X11 TrueColor visual.

Many of the methods in this class are final. This is because the underlying native graphics code makes assumptions about the layout and operation of this class and those assumptions are reflected in the implementations of the methods here that are marked final. You can subclass this class for other reaons, but you cannot override or modify the behaviour of those methods.

See Also:

ColorModel

DirectColorModel(int, int, int, int)
Constructs a DirectColorModel from the given masks specifying which bits in the pixel contain the red, green and blue color components.



DirectColorModel(int, int, int, int, int)
Constructs a DirectColorModel from the given masks specifying which bits in the pixel contain the alhpa, red, green and blue color components.

Interface Index **getAlpha**(int)
Return the alpha transparency value for the specified pixel in the range 0-255.
Interface Index **getAlphaMask**()
Returns the mask indicating which bits in a pixel contain the alpha transparency component.
Interface Index **getBlue**(int)
Returns the blue color compoment for the specified pixel in the range 0-255.
Interface Index **getBlueMask**()
Returns the mask indicating which bits in a pixel contain the blue color component.
Interface Index **getGreen**(int)
Returns the green color compoment for the specified pixel in the range 0-255.
Interface Index **getGreenMask**()
Returns the mask indicating which bits in a pixel contain the green color component.
Interface Index **getRGB**(int)
Returns the color of the pixel in the default RGB color model.
Interface Index **getRed**(int)
Returns the red color compoment for the specified pixel in the range 0-255.
Interface Index **getRedMask**()
Returns the mask indicating which bits in a pixel contain the red color component.

Interface Index

Interface Index **DirectColorModel**

```
public DirectColorModel(int bits,
                        int rmask,
                        int gmask,
                        int bmask)
```

Constructs a DirectColorModel from the given masks specifying which bits in the pixel contain the red, green and blue color components. Pixels described by this color model will all have alpha components of 255 (fully opaque). All of the bits in each mask must be contiguous and fit in the specified number of least significant bits of the integer.

Interface Index **DirectColorModel**

```
public DirectColorModel(int bits,
                        int rmask,
                        int gmask,
                        int bmask,
                        int amask)
```

Constructs a DirectColorModel from the given masks specifying which bits in the pixel contain the alhpa, red, green and blue color components. All of the bits in each mask must be contiguous and fit in the specified number of least significant bits of the integer.

getRedMask

```
public final int getRedMask()
```

Returns the mask indicating which bits in a pixel contain the red color component.

getGreenMask

```
public final int getGreenMask()
```

Returns the mask indicating which bits in a pixel contain the green color component.

getBlueMask

```
public final int getBlueMask()
```

Returns the mask indicating which bits in a pixel contain the blue color component.

getAlphaMask

```
public final int getAlphaMask()
```

Returns the mask indicating which bits in a pixel contain the alpha transparency component.

getRed

```
public final int getRed(int pixel)
```

Returns the red color compoment for the specified pixel in the range 0-255.
Overrides:
getRed in class ColorModel

getGreen

```
public final int getGreen(int pixel)
```

Returns the green color compoment for the specified pixel in the range 0-255.
Overrides:
getGreen in class ColorModel

getBlue

```
public final int getBlue(int pixel)
```

Returns the blue color compoment for the specified pixel in the range 0-255.
Overrides:
getBlue in class ColorModel

getAlpha

```
public final int getAlpha(int pixel)
```

Return the alpha transparency value for the specified pixel in the range 0-255.
Overrides:
getAlpha in class ColorModel

getRGB

```
public final int getRGB(int pixel)
```

Returns the color of the pixel in the default RGB color model.
Overrides:
getRGB in class ColorModel

See Also:
getRGBdefault

Class java.awt.image.FilteredImageSource

Class java.awt.image.FilteredImageSource

```
java.lang.Object
    |
    +----java.awt.image.FilteredImageSource
```

---

public class **FilteredImageSource**
extends Object
implements ImageProducer

This class is an implementation of the ImageProducer interface which takes an existing image and a filter object and uses them to produce image data for a new filtered version of the original image. Here is an example which filters an image by swapping the red and blue compents:

```
Image src = getImage("doc:///demo/images/duke/T1.gif");
ImageFilter colorfilter = new RedBlueSwapFilter();
Image img = createImage(new FilteredImageSource(src.getSource(),
                                                colorfilter));
```

See Also:
ImageProducer

FilteredImageSource(ImageProducer, ImageFilter)
Constructs an ImageProducer object from an existing ImageProducer and a filter object.

addConsumer(ImageConsumer)

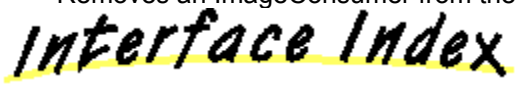
Adds an ImageConsumer to the list of consumers interested in data for this image.

isConsumer(ImageConsumer)

Determines whether an ImageConsumer is on the list of consumers currently interested in data for this image.

removeConsumer(ImageConsumer)

Removes an ImageConsumer from the list of consumers interested in data for this image.



requestTopDownLeftRightResend(ImageConsumer)

Requests that a given ImageConsumer have the image data delivered one more time in top-down, left-right order.

startProduction(ImageConsumer)

Adds an ImageConsumer to the list of consumers interested in data for this image, and immediately starts delivery of the image data through the ImageConsumer interface.

FilteredImageSource

```
public FilteredImageSource(ImageProducer orig,
                           ImageFilter imgf)
```

Constructs an ImageProducer object from an existing ImageProducer and a filter object.

See Also:

ImageFilter, createImage

addConsumer

```
public synchronized void addConsumer(ImageConsumer ic)
```

Adds an ImageConsumer to the list of consumers interested in data for this image.

See Also:

ImageConsumer

Interface Index **isConsumer**

```
public synchronized boolean isConsumer(ImageConsumer ic)
```

Determines whether an ImageConsumer is on the list of consumers currently interested in data for this image.
Returns:
true if the ImageConsumer is on the list; false otherwise
See Also:
ImageConsumer

Interface Index **removeConsumer**

```
public synchronized void removeConsumer(ImageConsumer ic)
```

Removes an ImageConsumer from the list of consumers interested in data for this image.
See Also:
ImageConsumer

Interface Index **startProduction**

```
public void startProduction(ImageConsumer ic)
```

Adds an ImageConsumer to the list of consumers interested in data for this image, and immediately starts delivery of the image data through the ImageConsumer interface.
See Also:
ImageConsumer

Interface Index **requestTopDownLeftRightResend**

```
public void requestTopDownLeftRightResend(ImageConsumer ic)
```

Requests that a given ImageConsumer have the image data delivered one more time in top-down, left-right order. The request is handed to the ImageFilter for further processing, since the ability to preserve the pixel ordering depends on the filter.
See Also:
ImageConsumer

Class java.awt.Image

Class java.awt.Image

```
java.lang.Object
    |
    +----java.awt.Image
```

---

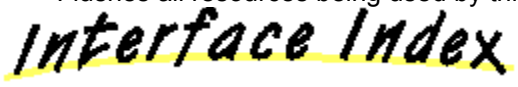
public class **Image**
extends Object

The image class is an abstract class. The image must be obtained in a platform specific way.

UndefinedProperty
The UndefinedProperty object should be returned whenever a property which was not defined for a particular image is fetched.

Image()

flush()
Flushes all resources being used by this Image object.



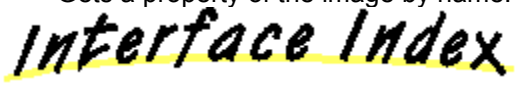
getGraphics()
Gets a graphics object to draw into this image.

getHeight(ImageObserver)

Gets the actual height of the image.

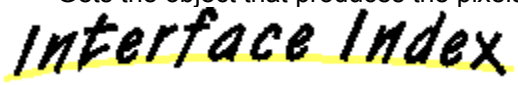
getProperty(String, ImageObserver)

Gets a property of the image by name.



getSource()

Gets the object that produces the pixels for the image.



getWidth(ImageObserver)

Gets the actual width of the image.

UndefinedProperty

```
public final static Object UndefinedProperty
```

The UndefinedProperty object should be returned whenever a property which was not defined for a particular image is fetched.

Image

```
public Image()
```

getWidth

```
public abstract int getWidth(ImageObserver observer)
```

Gets the actual width of the image. If the width is not known yet then the ImageObserver will be notified later and -1 will be returned.

See Also:

getHeight, ImageObserver

getHeight

```
public abstract int getHeight(ImageObserver observer)
```

Gets the actual height of the image. If the height is not known yet then the ImageObserver will be notified later and -1 will be returned.

See Also:

getWidth, ImageObserver

getSource

```
public abstract ImageProducer getSource()
```

Gets the object that produces the pixels for the image. This is used by the Image filtering classes and by the image conversion and scaling code.

See Also:

ImageProducer

getGraphics

```
public abstract Graphics getGraphics()
```

Gets a graphics object to draw into this image. This will only work for off-screen images.

See Also:

Graphics

getProperty

```
public abstract Object getProperty(String name,
                                   ImageObserver observer)
```

Gets a property of the image by name. Individual property names are defined by the various image formats. If a property is not defined for a particular image, this method will return the UndefinedProperty object. If the properties for this image are not yet known, then this method will return null and the ImageObserver object will be notified later. The property name "comment" should be used to store an optional comment which can be presented to the user as a description of the

image, its source, or its author.

See Also:

ImageObserver, UndefinedProperty

flush

```
public abstract void flush()
```

Flushes all resources being used by this Image object. This includes any pixel data that is being cached for rendering to the screen as well as any system resources that are being used to store data or pixels for the image. The image is reset to a state similar to when it was first created so that if it is again rendered, the image data will have to be recreated or fetched again from its source.

Interface java.awt.image.ImageConsumer

Interface java.awt.image.ImageConsumer

public interface **ImageConsumer**
extends Object

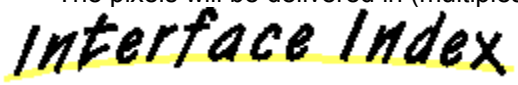
The interface for objects expressing interest in image data through the ImageProducer interfaces. When a consumer is added to an image producer, the producer delivers all of the data about the image using the method calls defined in this interface.

See Also:

ImageProducer

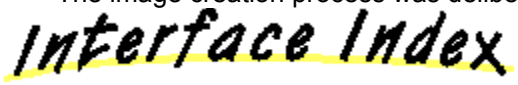
COMPLETESCANLINES

The pixels will be delivered in (multiples of) complete scanlines at a time.



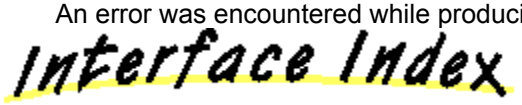
IMAGEABORTED

The image creation process was deliberately aborted.



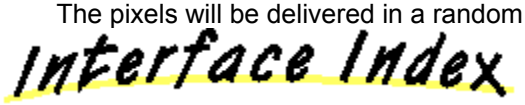
IMAGEERROR

An error was encountered while producing the image.



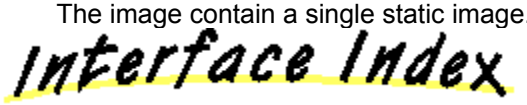
RANDOMPIXELORDER

The pixels will be delivered in a random order.



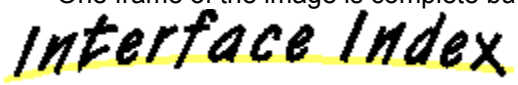
SINGLEFRAME

The image contain a single static image.



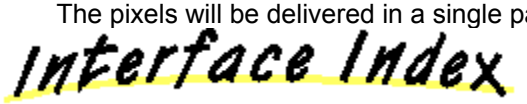
SINGLEFRAMEDONE

One frame of the image is complete but there are more frames to be delivered.



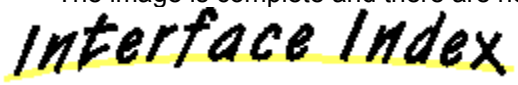
SINGLEPASS

The pixels will be delivered in a single pass.



STATICIMAGEDONE

The image is complete and there are no more pixels or frames to be delivered.

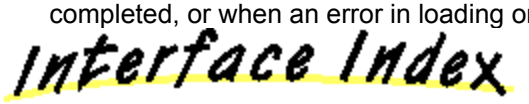


TOPDOWNLEFTRIGHT

The pixels will be delivered in top-down, left-to-right order.

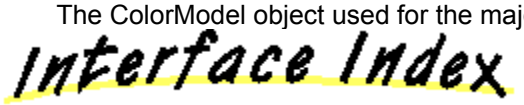
imageComplete(int)

The imageComplete method is called when the ImageProducer is finished delivering all of the pixels that the source image contains, or when a single frame of a multi-frame animation has been completed, or when an error in loading or producing the image has occured.



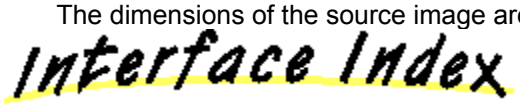
setColorModel(ColorModel)

The ColorModel object used for the majority of the pixels reported using the setPixels method calls.



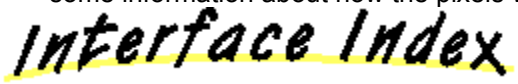
setDimensions(int, int)

The dimensions of the source image are reported using the setDimensions method call.



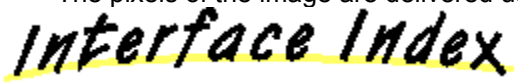
setHints(int)

The ImageProducer can deliver the pixels in any order, but the ImageConsumer may be able to scale or convert the pixels to the destination ColorModel more efficiently or with higher quality if it knows some information about how the pixels will be delivered up front.



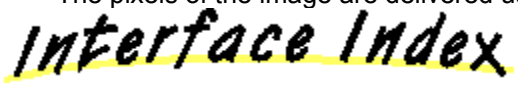
setPixels (int, int, int, int, ColorModel, byte[], int, int)

The pixels of the image are delivered using one or more calls to the setPixels method.



setPixels (int, int, int, int, ColorModel, int[], int, int)

The pixels of the image are delivered using one or more calls to the setPixels method.



setProperties(Hashtable)

Sets the extensible list of properties associated with this image.

RANDOMPIXELORDER

```
public final static int RANDOMPIXELORDER
```

The pixels will be delivered in a random order. This tells the ImageConsumer not to use any optimizations that depend on the order of pixel delivery, which should be the default assumption in the absence of any call to the setHints method.

See Also:

setHints

TOPDOWNLEFTRIGHT

```
public final static int TOPDOWNLEFTRIGHT
```

The pixels will be delivered in top-down, left-to-right order.

See Also:

setHints

COMPLETESCANLINES

```
public final static int COMPLETESCANLINES
```

The pixels will be delivered in (multiples of) complete scanlines at a time.

See Also:

setHints

SINGLEPASS

```
public final static int SINGLEPASS
```

The pixels will be delivered in a single pass. Each pixel will appear in only one call to any of the setPixels methods. An example of an image format which does not meet this criterion is a progressive JPEG image which defines pixels in multiple passes, each more refined than the previous.

See Also:

setHints

SINGLEFRAME

```
public final static int SINGLEFRAME
```

The image contain a single static image. The pixels will be defined in calls to the setPixels methods and then the imageComplete method will be called with the STATICIMAGEDONE flag after which no more image data will be delivered. An example of an image type which would not meet these criteria would be the output of a video feed, or the representation of a 3D rendering being manipulated by the user. The end of each frame in those types of images will be indicated by calling imageComplete with

the SINGLEFRAMEDONE flag.
See Also:
setHints, imageComplete

IMAGEERROR

```
public final static int IMAGEERROR
```

An error was encountered while producing the image.
See Also:
imageComplete

SINGLEFRAMEDONE

```
public final static int SINGLEFRAMEDONE
```

One frame of the image is complete but there are more frames to be delivered.
See Also:
imageComplete

STATICIMAGEDONE

```
public final static int STATICIMAGEDONE
```

The image is complete and there are no more pixels or frames to be delivered.
See Also:
imageComplete

IMAGEABORTED

```
public final static int IMAGEABORTED
```

The image creation process was deliberately aborted.
See Also:
imageComplete

Interface Index

setDimensions

```
public abstract void setDimensions(int width,
                                   int height)
```

The dimensions of the source image are reported using the setDimensions method call.

Interface Index

setProperties

```
public abstract void setProperties(Hashtable props)
```

Sets the extensible list of properties associated with this image.

Interface Index

setColorModel

```
public abstract void setColorModel(ColorModel model)
```

The ColorModel object used for the majority of the pixels reported using the setPixels method calls. Note that each set of pixels delivered using setPixels contains its own ColorModel object, so no assumption should be made that this model will be the only one used in delivering pixel values. A notable case where multiple ColorModel objects may be seen is a filtered image when for each set of pixels that it filters, the filter determines whether the pixels can be sent on untouched, using the original ColorModel, or whether the pixels should be modified (filtered) and passed on using a ColorModel more convenient for the filtering process.

See Also:

ColorModel

Interface Index

setHints

```
public abstract void setHints(int hintflags)
```

The ImageProducer can deliver the pixels in any order, but the ImageConsumer may be able to scale or convert the pixels to the destination ColorModel more efficiently or with higher quality if it knows some information about how the pixels will be delivered up front. The setHints method should be called before any calls to any of the setPixels methods with a bit mask of hints about the manner in which the pixels will be delivered. If the ImageProducer does not follow the guidelines for the indicated hint, the results are undefined.

Interface Index **setPixels**

```
public abstract void setPixels(int x,
                               int y,
                               int w,
                               int h,
                               ColorModel model,
                               byte pixels[],
                               int off,
                               int scansize)
```

The pixels of the image are delivered using one or more calls to the setPixels method. Each call specifies the location and size of the rectangle of source pixels that are contained in the array of pixels. The specified ColorModel object should be used to convert the pixels into their corresponding color and alpha components. Pixel (m,n) is stored in the pixels array at index (n * scansize + m + off). The pixels delivered using this method are all stored as bytes.

See Also:

ColorModel

Interface Index **setPixels**

```
public abstract void setPixels(int x,
                               int y,
                               int w,
                               int h,
                               ColorModel model,
                               int pixels[],
                               int off,
                               int scansize)
```

The pixels of the image are delivered using one or more calls to the setPixels method. Each call specifies the location and size of the rectangle of source pixels that are contained in the array of pixels. The specified ColorModel object should be used to convert the pixels into their corresponding color and alpha components. Pixel (m,n) is stored in the pixels array at index (n * scansize + m + off). The pixels delivered using this method are all stored as ints.

See Also:

ColorModel

Interface Index **imageComplete**

```
public abstract void imageComplete(int status)
```

The imageComplete method is called when the ImageProducer is finished delivering all of the pixels that the source image contains, or when a single frame of a multi-frame animation has been

completed, or when an error in loading or producing the image has occured. The ImageConsumer should remove itself from the list of consumers registered with the ImageProducer at this time, unless it is interested in successive frames.

See Also:

removeConsumer

Class java.awt.image.ImageFilter

Class java.awt.image.ImageFilter

```
java.lang.Object
    |
    +----java.awt.image.ImageFilter
```

---

public class **ImageFilter**
extends Object
implements ImageConsumer, Cloneable

This class implements a filter for the set of interface methods that are used to deliver data from an ImageProducer to an ImageConsumer. It is meant to be used in conjunction with a FilteredImageSource object to produce filtered versions of existing images. It is a base class that provides the calls needed to implement a "Null filter" which has no effect on the data being passed through. Filters should subclass this class and override the methods which deal with the data that needs to be filtered and modify it as necessary.

See Also:

FilteredImageSource, ImageConsumer

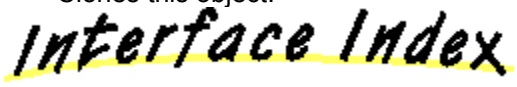
consumer

The consumer of the particular image data stream for which this instance of the ImageFilter is filtering data.

ImageFilter()

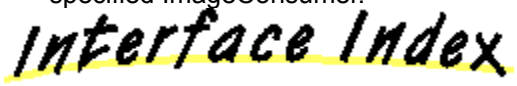
clone()

Clones this object.



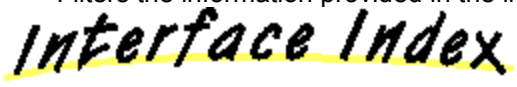
getFilterInstance(ImageConsumer)

Returns a unique instance of an ImageFilter object which will actually perform the filtering for the specified ImageConsumer.



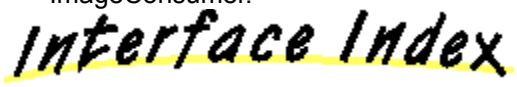
imageComplete(int)

Filters the information provided in the imageComplete method of the ImageConsumer interface.



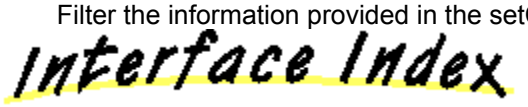
resendTopDownLeftRight(ImageProducer)

Responds to a request for a TopDownLeftRight (TDLR) ordered resend of the pixel data from an ImageConsumer.



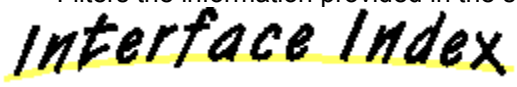
setColorModel(ColorModel)

Filter the information provided in the setColorModel method of the ImageConsumer interface.



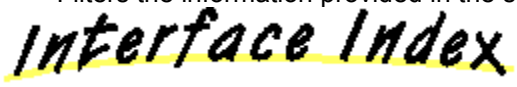
setDimensions(int, int)

Filters the information provided in the setDimensions method of the ImageConsumer interface.



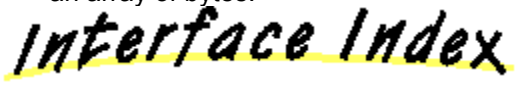
setHints(int)

Filters the information provided in the setHints method of the ImageConsumer interface.



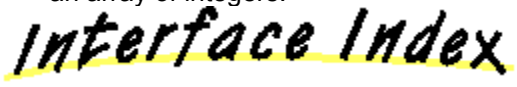
setPixels (int, int, int, int, ColorModel, byte[], int, int)

Filters the information provided in the setPixels method of the ImageConsumer interface which takes an array of bytes.



setPixels (int, int, int, int, ColorModel, int[], int, int)

Filters the information provided in the setPixels method of the ImageConsumer interface which takes an array of integers.



setProperties(Hashtable)

Passes the properties from the source object along after adding a property indicating the stream of filters it has been run through.

consumer

```
protected ImageConsumer consumer
```

The consumer of the particular image data stream for which this instance of the ImageFilter is

filtering data. It is not initialized during the constructor, but rather during the getFilterInstance() method call when the FilteredImageSource is creating a unique instance of this object for a particular image data stream.

See Also:

getFilterInstance, ImageConsumer

ImageFilter

```
public ImageFilter()
```

getFilterInstance

```
public ImageFilter getFilterInstance(ImageConsumer ic)
```

Returns a unique instance of an ImageFilter object which will actually perform the filtering for the specified ImageConsumer. The default implementation just clones this object.

setDimensions

```
public void setDimensions(int width,
                          int height)
```

Filters the information provided in the setDimensions method of the ImageConsumer interface.

See Also:

setDimensions

setProperties

```
public void setProperties(Hashtable props)
```

Passes the properties from the source object along after adding a property indicating the stream of filters it has been run through.

 setColorModel

```
public void setColorModel(ColorModel model)
```

Filter the information provided in the setColorModel method of the ImageConsumer interface.

See Also:

setColorModel

 setHints

```
public void setHints(int hints)
```

Filters the information provided in the setHints method of the ImageConsumer interface.

See Also:

setHints

 setPixels

```
public void setPixels(int x,
                      int y,
                      int w,
                      int h,
                      ColorModel model,
                      byte pixels[],
                      int off,
                      int scansize)
```

Filters the information provided in the setPixels method of the ImageConsumer interface which takes an array of bytes.

See Also:

setPixels

 setPixels

```
public void setPixels(int x,
                      int y,
                      int w,
                      int h,
                      ColorModel model,
                      int pixels[],
```

```
                    int off,
                    int scansize)
```

Filters the information provided in the setPixels method of the ImageConsumer interface which takes an array of integers.

See Also:

setPixels

imageComplete

```
public void imageComplete(int status)
```

Filters the information provided in the imageComplete method of the ImageConsumer interface.

See Also:

imageComplete

resendTopDownLeftRight

```
public void resendTopDownLeftRight(ImageProducer ip)
```

Responds to a request for a TopDownLeftRight (TDLR) ordered resend of the pixel data from an ImageConsumer. The ImageFilter can respond to this request in one of three ways.

1. If the filter can determine that it will forward the pixels in TDLR order if its upstream producer object sends them in TDLR order, then the request is automatically forwarded by default to the indicated ImageProducer using this filter as the requesting ImageConsumer, so no override is necessary.
2. If the filter can resend the pixels in the right order on its own (presumably because the generated pixels have been saved in some sort of buffer), then it can override this method and simply resend the pixels in TDLR order as specified in the ImageProducer API.
3. If the filter simply returns from this method then the request will be ignored and no resend will occur.

@see ImageProducer#requestTopDownLeftRightResend

Parameters:

ip - The ImageProducer that is feeding this instance of the filter - also the ImageProducer that the request should be forwarded to if necessary.

clone

```
public Object clone()
```

Clones this object.
Overrides:
clone in class Object

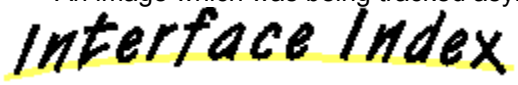
Interface java.awt.image.ImageObserver

Interface java.awt.image.ImageObserver

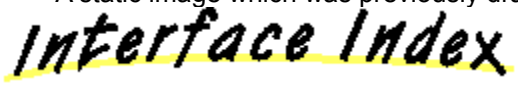
public interface **ImageObserver**
extends Object

An asynchronous update interface for receiving notifications about Image information as the Image is constructed.

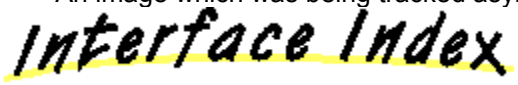
ABORT
An image which was being tracked asynchronously was aborted before production was complete.



ALLBITS
A static image which was previously drawn is now complete and can be drawn again in its final form.



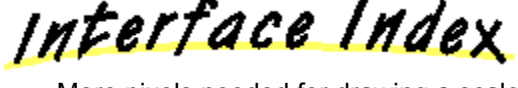
ERROR
An image which was being tracked asynchronously has encountered an error.



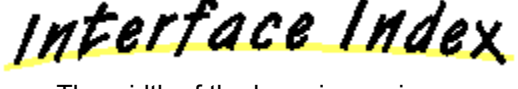
FRAMEBITS
Another complete frame of a multi-frame image which was previously drawn is now available to be drawn again.

HEIGHT
The height of the base image is now available and can be taken from the height argument to the imageUpdate callback method.

PROPERTIES
The properties of the image are now available.



SOMEBITS
More pixels needed for drawing a scaled variation of the image are available.



WIDTH
The width of the base image is now available and can be taken from the width argument to the imageUpdate callback method.

Interface Index

imageUpdate(Image, int, int, int, int, int)

This method is called when information about an image which was previously requested using an asynchronous interface becomes available.

Interface Index

Interface Index

WIDTH

```
public final static int WIDTH
```

The width of the base image is now available and can be taken from the width argument to the imageUpdate callback method.

See Also:

getWidth, imageUpdate

HEIGHT

```
public final static int HEIGHT
```

The height of the base image is now available and can be taken from the height argument to the imageUpdate callback method.

See Also:

getHeight, imageUpdate

PROPERTIES

```
public final static int PROPERTIES
```

The properties of the image are now available.

See Also:

getProperty, imageUpdate

Interface Index **SOMEBITS**

```
public final static int SOMEBITS
```

More pixels needed for drawing a scaled variation of the image are available. The bounding box of the new pixels can be taken from the x, y, width, and height arguments to the imageUpdate callback method.

See Also:

drawImage, imageUpdate

Interface Index **FRAMEBITS**

```
public final static int FRAMEBITS
```

Another complete frame of a multi-frame image which was previously drawn is now available to be drawn again. The x, y, width, and height arguments to the imageUpdate callback method should be ignored.

See Also:

drawImage, imageUpdate

Interface Index **ALLBITS**

```
public final static int ALLBITS
```

A static image which was previously drawn is now complete and can be drawn again in its final form. The x, y, width, and height arguments to the imageUpdate callback method should be ignored.

See Also:

drawImage, imageUpdate

Interface Index **ERROR**

```
public final static int ERROR
```

An image which was being tracked asynchronously has encountered an error. No further information will become available and drawing the image will fail. As a convenience, the ABORT flag will be indicated at the same time to indicate that the image production was aborted.

See Also:

imageUpdate

ABORT

```
public final static int ABORT
```

An image which was being tracked asynchronously was aborted before production was complete. No more information will become available without further action to trigger another image production sequence. If the ERROR flag was not also set in this image update, then accessing any of the data in the image will restart the production again, probably from the beginning.

See Also:

imageUpdate

imageUpdate

```
public abstract boolean imageUpdate(Image img,
                                    int infoflags,
                                    int x,
                                    int y,
                                    int width,
                                    int height)
```

This method is called when information about an image which was previously requested using an asynchronous interface becomes available. Asynchronous interfaces are method calls such as getWidth(ImageObserver) and drawImage(img, x, y, ImageObserver) which take an ImageObserver object as an argument. Those methods register the caller as interested either in information about the overall image itself (in the case of getWidth(ImageObserver)) or about an output version of an image (in the case of the drawImage(img, x, y, [w, h,] ImageObserver) call).

This method should return true if further updates are needed or false if the required information has been acquired. The image which was being tracked is passed in using the img argument. Various constants are combined to form the infoflags argument which indicates what information about the image is now available. The interpretation of the x, y, width, and height arguments depends on the contents of the infoflags argument.

See Also:

getWidth, getHeight, drawImage

Interface java.awt.image.ImageProducer

Interface java.awt.image.ImageProducer

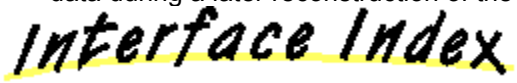
public interface **ImageProducer**
extends Object

The interface for objects which can produce the image data for Images. Each image contains an ImageProducer which is used to reconstruct the image whenever it is needed, for example, when a new size of the Image is scaled, or when the width or height of the Image is being requested.

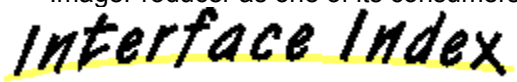
See Also:

ImageConsumer

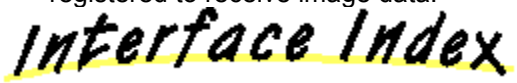
addConsumer(ImageConsumer)
This method is used to register an ImageConsumer with the ImageProducer for access to the image data during a later reconstruction of the Image.



isConsumer(ImageConsumer)
This method determines if a given ImageConsumer object is currently registered with this ImageProducer as one of its consumers.



removeConsumer(ImageConsumer)
This method removes the given ImageConsumer object from the list of consumers currently registered to receive image data.



requestTopDownLeftRightResend(ImageConsumer)
This method is used by an ImageConsumer to request that the ImageProducer attempt to resend the image data one more time in TOPDOWNLEFTRIGHT order so that higher quality conversion algorithms which depend on receiving pixels in order can be used to produce a better output version of the image.

startProduction(ImageConsumer)
This method both registers the given ImageConsumer object as a consumer and starts an immediate reconstruction of the image data which will then be delivered to this consumer and any other consumer which may have already been registered with the producer.

addConsumer

```
public abstract void addConsumer(ImageConsumer ic)
```

This method is used to register an ImageConsumer with the ImageProducer for access to the image data during a later reconstruction of the Image. The ImageProducer may, at its discretion, start delivering the image data to the consumer using the ImageConsumer interface immediately, or when the next available image reconstruction is triggered by a call to the startProduction method.

See Also:

startProduction

isConsumer

```
public abstract boolean isConsumer(ImageConsumer ic)
```

This method determines if a given ImageConsumer object is currently registered with this ImageProducer as one of its consumers.

removeConsumer

```
public abstract void removeConsumer(ImageConsumer ic)
```

This method removes the given ImageConsumer object from the list of consumers currently registered to receive image data. It is not considered an error to remove a consumer that is not currently registered. The ImageProducer should stop sending data to this consumer as soon as is feasible.

startProduction

```
public abstract void startProduction(ImageConsumer ic)
```

This method both registers the given ImageConsumer object as a consumer and starts an immediate reconstruction of the image data which will then be delivered to this consumer and any other consumer which may have already been registered with the producer. This method differs from the addConsumer method in that a reproduction of the image data should be triggered as soon as possible.

See Also:

addConsumer

requestTopDownLeftRightResend

```
public abstract void requestTopDownLeftRightResend(ImageConsumer ic)
```

This method is used by an ImageConsumer to request that the ImageProducer attempt to resend the image data one more time in TOPDOWNLEFTRIGHT order so that higher quality conversion algorithms which depend on receiving pixels in order can be used to produce a better output version of the image. The ImageProducer is free to ignore this call if it cannot resend the data in that order. If the data can be resent, then the ImageProducer should respond by executing the following minimum set of ImageConsumer method calls:

```
ic.setHints(TOPDOWNLEFTRIGHT | );
ic.setPixels(...);      // As many times as needed
ic.imageComplete();
```

See Also:

setHints

Class java.awt.image.IndexColorModel

Class java.awt.image.IndexColorModel

```
java.lang.Object
    |
    +----java.awt.image.ColorModel
              |
              +----java.awt.image.IndexColorModel
```

---

public class **IndexColorModel**
extends ColorModel

A ColorModel class that specifies a translation from pixel values to alpha, red, green, and blue color components for pixels which represent indices into a fixed colormap. An optional transparent pixel value can be supplied which indicates a completely transparent pixel, regardless of any alpha value recorded for that pixel value. This color model is similar to an X11 PseudoColor visual.

Many of the methods in this class are final. The reason for this is that the underlying native graphics code makes assumptions about the layout and operation of this class and those assumptions are reflected in the implementations of the methods here that are marked final. You can subclass this class for other reaons, but you cannot override or modify the behaviour of those methods.

See Also:

ColorModel

IndexColorModel(int, int, byte[], byte[], byte[])
Constructs an IndexColorModel from the given arrays of red, green, and blue components.

IndexColorModel (int, int, byte[], byte[], byte[], int)
Constructs an IndexColorModel from the given arrays of red, green, and blue components.

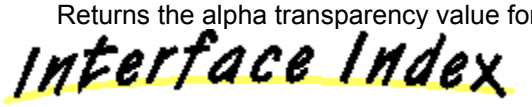
IndexColorModel (int, int, byte[], byte[], byte[], byte[])
Constructs an IndexColorModel from the given arrays of red, green, blue and alpha components.

IndexColorModel(int, int, byte[], int, boolean)
Constructs an IndexColorModel from a single arrays of packed red, green, blue and optional alpha

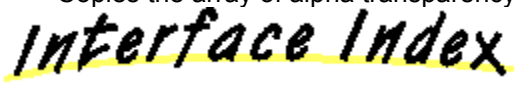
components.

IndexColorModel(int, int, byte[], int, boolean, int)
Constructs an IndexColorModel from a single arrays of packed red, green, blue and optional alpha components.

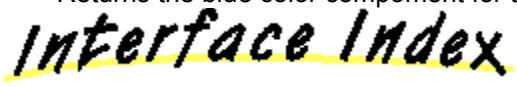
getAlpha(int)
Returns the alpha transparency value for the specified pixel in the range 0-255.



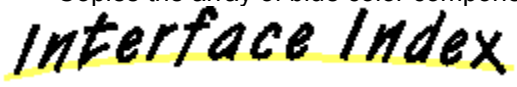
getAlphas(byte[])
Copies the array of alpha transparency values into the given array.



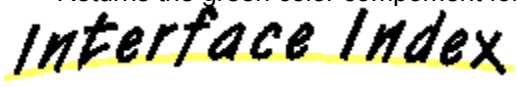
getBlue(int)
Returns the blue color compoment for the specified pixel in the range 0-255.



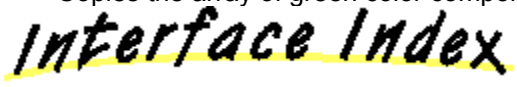
getBlues(byte[])
Copies the array of blue color components into the given array.



getGreen(int)
Returns the green color compoment for the specified pixel in the range 0-255.

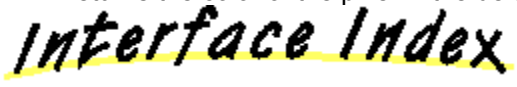


getGreens(byte[])
Copies the array of green color components into the given array.

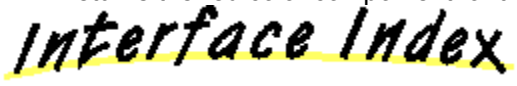


getMapSize()
Returns the size of the color component arrays in this IndexColorModel.

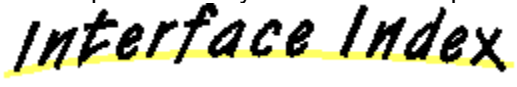
getRGB(int)
Returns the color of the pixel in the default RGB color model.



getRed(int)
Returns the red color compoment for the specified pixel in the range 0-255.



getReds(byte[])
Copies the array of red color components into the given array.



getTransparentPixel()
Returns the index of the transparent pixel in this IndexColorModel or -1 if there is no transparent pixel.

Interface Index

IndexColorModel

```
public IndexColorModel(int bits,
                       int size,
                       byte r[],
                       byte g[],
                       byte b[])
```

Constructs an IndexColorModel from the given arrays of red, green, and blue components. Pixels described by this color model will all have alpha components of 255 (fully opaque). All of the arrays specifying the color components must have at least the specified number of entries.

Parameters:

bits - The number of bits each pixel occupies.
size - The size of the color component arrays.
r - The array of red color components.
g - The array of green color components.
b - The array of blue color components.

Interface Index

IndexColorModel

```
public IndexColorModel(int bits,
                       int size,
                       byte r[],
                       byte g[],
                       byte b[],
                       int trans)
```

Constructs an IndexColorModel from the given arrays of red, green, and blue components. Pixels described by this color model will all have alpha components of 255 (fully opaque), except for the indicated transparent pixel. All of the arrays specifying the color components must have at least the specified number of entries.

Parameters:

bits - The number of bits each pixel occupies.
size - The size of the color component arrays.
r - The array of red color components.
g - The array of green color components.
b - The array of blue color components.
trans - The index of the transparent pixel.

Interface Index

IndexColorModel

```
public IndexColorModel(int bits,
                       int size,
                       byte r[],
                       byte g[],
                       byte b[],
                       byte a[])
```

Constructs an IndexColorModel from the given arrays of red, green, blue and alpha components. All of the arrays specifying the color components must have at least the specified number of entries.

Parameters:

bits - The number of bits each pixel occupies.
size - The size of the color component arrays.
r - The array of red color components.
g - The array of green color components.
b - The array of blue color components.
a - The array of alpha value components.

IndexColorModel

```
public IndexColorModel(int bits,
                       int size,
                       byte cmap[],
                       int start,
                       boolean hasalpha)
```

Constructs an IndexColorModel from a single arrays of packed red, green, blue and optional alpha components. The array must have enough values in it to fill all of the needed component arrays of the specified size.

Parameters:

bits - The number of bits each pixel occupies.
size - The size of the color component arrays.
cmap - The array of color components.
start - The starting offset of the first color component.
hasalpha - Indicates whether alpha values are contained in the cmap array.

IndexColorModel

```
public IndexColorModel(int bits,
                       int size,
                       byte cmap[],
                       int start,
                       boolean hasalpha,
                       int trans)
```

Constructs an IndexColorModel from a single arrays of packed red, green, blue and optional alpha components. The specified transparent index represents a pixel which will be considered entirely

transparent regardless of any alpha value specified for it. The array must have enough values in it to fill all of the needed component arrays of the specified size.

Parameters:

bits - The number of bits each pixel occupies.
size - The size of the color component arrays.
cmap - The array of color components.
start - The starting offset of the first color component.
hasalpha - Indicates whether alpha values are contained in the cmap array.
trans - The index of the fully transparent pixel.

getMapSize

```
public final int getMapSize()
```

Returns the size of the color component arrays in this IndexColorModel.

getTransparentPixel

```
public final int getTransparentPixel()
```

Returns the index of the transparent pixel in this IndexColorModel or -1 if there is no transparent pixel.

getReds

```
public final void getReds(byte r[])
```

Copies the array of red color components into the given array. Only the initial entries of the array as specified by getMapSize() are written.

getGreens

```
public final void getGreens(byte g[])
```

Copies the array of green color components into the given array. Only the initial entries of the array as specified by getMapSize() are written.

Interface Index **getBlues**

```
public final void getBlues(byte b[])
```

Copies the array of blue color components into the given array. Only the initial entries of the array as specified by getMapSize() will be written.

Interface Index **getAlphas**

```
public final void getAlphas(byte a[])
```

Copies the array of alpha transparency values into the given array. Only the initial entries of the array as specified by getMapSize() will be written.

Interface Index **getRed**

```
public final int getRed(int pixel)
```

Returns the red color compoment for the specified pixel in the range 0-255.
Overrides:
getRed in class ColorModel

Interface Index **getGreen**

```
public final int getGreen(int pixel)
```

Returns the green color compoment for the specified pixel in the range 0-255.
Overrides:
getGreen in class ColorModel

Interface Index **getBlue**

```
public final int getBlue(int pixel)
```

Returns the blue color compoment for the specified pixel in the range 0-255.
Overrides:
getBlue in class ColorModel

getAlpha

```
public final int getAlpha(int pixel)
```

Returns the alpha transparency value for the specified pixel in the range 0-255.

Overrides:

getAlpha in class ColorModel

getRGB

```
public final int getRGB(int pixel)
```

Returns the color of the pixel in the default RGB color model.

Overrides:

getRGB in class ColorModel

See Also:

getRGBdefault

Class java.awt.image.MemoryImageSource

Class java.awt.image.MemoryImageSource

```
java.lang.Object
    |
    +----java.awt.image.MemoryImageSource
```

---

public class **MemoryImageSource**
extends Object
implements ImageProducer

This class is an implementation of the ImageProducer interface which uses an array to produce pixel values for an Image. Here is an example which calculates a 100x100 image representing a fade from black to blue along the X axis and a fade from black to red along the Y axis:

```
        int w = 100;
        int h = 100;
        int pix[] = new int[w * h];
        int index = 0;
        for (int y = 0; y
```

See Also:

ImageProducer

Interface Index

MemoryImageSource(int, int, ColorModel, byte[], int, int)

Constructs an ImageProducer object which uses an array of bytes to produce data for an Image object.

Interface Index

MemoryImageSource(int, int, ColorModel, byte[], int, int, Hashtable)

Constructs an ImageProducer object which uses an array of bytes to produce data for an Image object.

Interface Index

MemoryImageSource(int, int, ColorModel, int[], int, int)

Constructs an ImageProducer object which uses an array of integers to produce data for an Image object.

Interface Index

MemoryImageSource(int, int, ColorModel, int[], int, int, Hashtable)

Constructs an ImageProducer object which uses an array of integers to produce data for an Image object.

Interface Index

MemoryImageSource(int, int, int[], int, int)

Constructs an ImageProducer object which uses an array of integers in the default RGB ColorModel to produce data for an Image object.

Interface Index

MemoryImageSource(int, int, int[], int, int, Hashtable)

Constructs an ImageProducer object which uses an array of integers in the default RGB ColorModel to produce data for an Image object.

Interface Index

Interface Index

addConsumer(ImageConsumer)

Adds an ImageConsumer to the list of consumers interested in data for this image.

Interface Index

isConsumer(ImageConsumer)

Determine if an ImageConsumer is on the list of consumers currently interested in data for this image.

Interface Index

removeConsumer(ImageConsumer)

Remove an ImageConsumer from the list of consumers interested in data for this image.

Interface Index

requestTopDownLeftRightResend(ImageConsumer)

Requests that a given ImageConsumer have the image data delivered one more time in top-down, left-right order.

Interface Index

startProduction(ImageConsumer)

Adds an ImageConsumer to the list of consumers interested in data for this image, and immediately start delivery of the image data through the ImageConsumer interface.

Interface Index

Interface Index

MemoryImageSource

```
public MemoryImageSource(int w,
                         int h,
                         ColorModel cm,
                         byte pix[],
                         int off,
                         int scan)
```

Constructs an ImageProducer object which uses an array of bytes to produce data for an Image object.

See Also:

createImage

Interface Index

MemoryImageSource

```
public MemoryImageSource(int w,
                         int h,
                         ColorModel cm,
                         byte pix[],
                         int off,
                         int scan,
                         Hashtable props)
```

Constructs an ImageProducer object which uses an array of bytes to produce data for an Image object.

See Also:

createImage

Interface Index

MemoryImageSource

```
public MemoryImageSource(int w,
                         int h,
                         ColorModel cm,
                         int pix[],
                         int off,
                         int scan)
```

Constructs an ImageProducer object which uses an array of integers to produce data for an Image object.

See Also:

createImage

Interface Index

MemoryImageSource

```
public MemoryImageSource(int w,
                         int h,
                         ColorModel cm,
                         int pix[],
                         int off,
```

int scan,
Hashtable props)

Constructs an ImageProducer object which uses an array of integers
to produce data for an Image object.

See Also:

createImage

Interface Index

MemoryImageSource

```
public MemoryImageSource(int w,
                         int h,
                         int pix[],
                         int off,
                         int scan)
```

Constructs an ImageProducer object which uses an array of integers
in the default RGB ColorModel to produce data for an Image object.

See Also:

createImage, getRGBdefault

Interface Index

MemoryImageSource

```
public MemoryImageSource(int w,
                         int h,
                         int pix[],
                         int off,
                         int scan,
                         Hashtable props)
```

Constructs an ImageProducer object which uses an array of integers
in the default RGB ColorModel to produce data for an Image object.

See Also:

createImage, getRGBdefault

Interface Index

Interface Index

addConsumer

```
public synchronized void addConsumer(ImageConsumer ic)
```

Adds an ImageConsumer to the list of consumers interested in
data for this image.

See Also:

ImageConsumer

Interface Index

isConsumer

```
public synchronized boolean isConsumer(ImageConsumer ic)
```

Determine if an ImageConsumer is on the list of consumers currently
interested in data for this image.

Returns:

true if the ImageConsumer is on the list; false otherwise

See Also:

ImageConsumer

Interface Index

removeConsumer

```
public synchronized void removeConsumer(ImageConsumer ic)
```

Remove an ImageConsumer from the list of consumers interested in
data for this image.

See Also:

ImageConsumer

Interface Index

startProduction

```
public void startProduction(ImageConsumer ic)
```

Adds an ImageConsumer to the list of consumers interested in
data for this image, and immediately start delivery of the
image data through the ImageConsumer interface.

See Also:

ImageConsumer

Interface Index

requestTopDownLeftRightResend

```
public void requestTopDownLeftRightResend(ImageConsumer ic)
```

Requests that a given ImageConsumer have the image data delivered
one more time in top-down, left-right order.

See Also:

ImageConsumer

Class java.awt.image.PixelGrabber

Class java.awt.image.PixelGrabber

```
java.lang.Object
   |
   +----java.awt.image.PixelGrabber
```

---

public class **PixelGrabber**
extends Object
implements ImageConsumer

The PixelGrabber class implements an ImageConsumer which can be attached to an Image or ImageProducer object to retrieve a subset of the pixels in that image. Here is an example:

```
public abstract void handlesinglepixel(int x, int y, int pixel);
public void handlepixels(Image img, int x, int y, int w, int h) {
        int[] pixels = new int[w * h];
        PixelGrabber pg = new PixelGrabber(img, x, y, w, h, pixels, 0, w);
        try {
            pg.grabPixels();
        } catch (InterruptedException e) {
            System.err.println("interrupted waiting for pixels!");
            return;
        }
        if ((pg.status() &ImageObserver.ABORT) != 0) {
            System.err.println("image fetch aborted or errored");
            return;
        }
        for (int j = 0; j
```

_

Interface Index

Interface Index

PixelGrabber(Image, int, int, int, int, int[], int, int)

Create a PixelGrabber object to grab the (x, y, w, h) rectangular section of pixels from the specified image into the given array.

Interface Index

PixelGrabber(ImageProducer, int, int, int, int, int[], int, int)

Create a PixelGrabber object to grab the (x, y, w, h) rectangular section of pixels from the image produced by the specified ImageProducer into the given array.

Interface Index

Interface Index

grabPixels()

Request the Image or ImageProducer to start delivering pixels and wait for all of the pixels in the rectangle of interest to be delivered.

Interface Index

grabPixels(long)

Request the Image or ImageProducer to start delivering pixels and wait for all of the pixels in the rectangle of interest to be delivered or until the specified timeout has elapsed.

Interface Index

imageComplete(int)

The imageComplete method is part of the ImageConsumer API which this class must implement to retrieve the pixels.

Interface Index

setColorModel(ColorModel)

The setColorModel method is part of the ImageConsumer API which this class must implement to retrieve the pixels.

Interface Index

setDimensions(int, int)

The setDimensions method is part of the ImageConsumer API which this class must implement to retrieve the pixels.

Interface Index

setHints(int)

The setHints method is part of the ImageConsumer API which this class must implement to retrieve the pixels.

Interface Index

setPixels(int, int, int, int, ColorModel, byte[], int, int)

The setPixels method is part of the ImageConsumer API which this class must implement to retrieve the pixels.

Interface Index

setPixels(int, int, int, int, ColorModel, int[], int, int)

The setPixels method is part of the ImageConsumer API which this class must implement to retrieve the pixels.

Interface Index

setProperties(Hashtable)

The setProperties method is part of the ImageConsumer API which this class must implement to retrieve the pixels.

Interface Index

status()

Return the status of the pixels.

Interface Index

PixelGrabber

```
public PixelGrabber(Image img,
                    int x,
                    int y,
                    int w,
                    int h,
                    int pix[],
                    int off,
                    int scansize)
```

Create a PixelGrabber object to grab the (x, y, w, h) rectangular section of pixels from the specified image into the given array. The pixels are stored into the array in the default RGB ColorModel. The RGB data for pixel (i, j) where (i, j) is inside the rectangle (x, y, w, h) is stored in the array at
pix[(j - y) * scansize + (i - x) + off].

Parameters:

img - the image to retrieve pixels from

x - the x coordinate of the upper left corner of the rectangle of pixels to retrieve from the image, relative to the default (unscaled) size of the image

y - the y coordinate of the upper left corner of the rectangle of pixels to retrieve from the image

w - the width of the rectangle of pixels to retrieve

h - the height of the rectangle of pixels to retrieve

pix - the array of integers which are to be used to hold the RGB pixels retrieved from the image

off - the offset into the array of where to store the first pixel

scansize - the distance from one row of pixels to the next in the array

See Also:

getRGBdefault

Interface Index

PixelGrabber

```
public PixelGrabber(ImageProducer ip,
                    int x,
                    int y,
                    int w,
                    int h,
                    int pix[],
                    int off,
                    int scansize)
```

Create a PixelGrabber object to grab the (x, y, w, h) rectangular section of pixels from the image produced by the specified ImageProducer into the given array.
The pixels are stored into the array in the default RGB ColorModel.
The RGB data for pixel (i, j) where (i, j) is inside the rectangle (x, y, w, h) is stored in the array at
pix[(j - y) * scansize + (i - x) + off].

Parameters:

img - the image to retrieve pixels from

x - the x coordinate of the upper left corner of the rectangle of pixels to retrieve from the image, relative to the default (unscaled) size of the image

y - the y coordinate of the upper left corner of the rectangle of pixels to retrieve from the image

w - the width of the rectangle of pixels to retrieve

h - the height of the rectangle of pixels to retrieve

pix - the array of integers which are to be used to hold the RGB pixels retrieved from the image

off - the offset into the array of where to store the first pixel

scansize - the distance from one row of pixels to the next in the array

See Also:

getRGBdefault

Interface Index

Interface Index

grabPixels

public boolean grabPixels() throws InterruptedException

Request the Image or ImageProducer to start delivering pixels and wait for all of the pixels in the rectangle of interest to be delivered.

Returns:

true if the pixels were successfully grabbed, false on abort, error or timeout

Throws: InterruptedException

Another thread has interrupted this thread.

Interface Index

grabPixels

public synchronized boolean grabPixels(long ms) throws InterruptedException

Request the Image or ImageProducer to start delivering pixels and wait for all of the pixels in the rectangle of interest to be delivered or until the specified timeout has elapsed.

Parameters:

ms - the number of milliseconds to wait for the image pixels to arrive before timing out

Returns:

true if the pixels were successfully grabbed, false on abort, error or timeout

Throws: InterruptedException

Another thread has interrupted this thread.

Interface Index

status

public synchronized int status()

Return the status of the pixels. The ImageObserver flags representing the available pixel information are returned.

Returns:

the bitwise OR of all relevant ImageObserver flags

See Also:

ImageObserver

Interface Index

setDimensions

```
public void setDimensions(int width,
                          int height)
```

The setDimensions method is part of the ImageConsumer API which
this class must implement to retrieve the pixels.

Interface Index

setHints

```
public void setHints(int hints)
```

The setHints method is part of the ImageConsumer API which this class must implement to retrieve the pixels.

Interface Index

setProperties

```
public void setProperties(Hashtable props)
```

The setProperties method is part of the ImageConsumer API which
this class must implement to retrieve the pixels.

Interface Index

setColorModel

```
public void setColorModel(ColorModel model)
```

The setColorModel method is part of the ImageConsumer API which
this class must implement to retrieve the pixels.

Interface Index

setPixels

```
public void setPixels(int srcX,
                      int srcY,
                      int srcW,
                      int srcH,
                      ColorModel model,
                      byte pixels[],
                      int srcOff,
                      int srcScan)
```

The setPixels method is part of the ImageConsumer API which
this class must implement to retrieve the pixels.

setPixels

```
public void setPixels(int srcX,
                      int srcY,
                      int srcW,
                      int srcH,
                      ColorModel model,
                      int pixels[],
                      int srcOff,
                      int srcScan)
```

The setPixels method is part of the ImageConsumer API which this class must implement to retrieve the pixels.

Interface Index

imageComplete

```
public synchronized void imageComplete(int status)
```

The imageComplete method is part of the ImageConsumer API which
this class must implement to retrieve the pixels.

Class java.awt.image.RGBImageFilter

Class java.awt.image.RGBImageFilter

```
java.lang.Object
   |
   +----java.awt.image.ImageFilter
           |
           +----java.awt.image.RGBImageFilter
```

---

public class **RGBImageFilter**
extends ImageFilter

This class provides an easy way to create an ImageFilter which modifies the pixels of an image in the default RGB ColorModel. It is meant to be used in conjunction with a FilteredImageSource object to produce filtered versions of existing images. It is an abstract class that provides the calls needed to channel all of the pixel data through a single method which converts pixels one at a time in the default RGB ColorModel regardless of the ColorModel being used by the ImageProducer. The only method which needs to be defined to create a useable image filter is the filterRGB method. Here is an example of a definition of a filter which swaps the red and blue components of an image:

```
class RedBlueSwapFilter extends RGBImageFilter {
    public RedBlueSwapFilter() {
        // The filter's operation does not depend on the
        // pixel's location, so IndexColorModels can be
        // filtered directly.
        canFilterIndexColorModel = true;
    }
    public int filterRGB(int x, int y, int rgb) {
        return ((rgb &0xff00ff00)
                | ((rgb &0xff0000) >> 16)
                | ((rgb &0xff)
```

See Also:

FilteredImageSource, ImageFilter, getRGBdefault

Interface Index

Interface Index

canFilterIndexColorModel

This boolean indicates whether or not it is acceptable to apply the color filtering of the filterRGB method to the color table entries of an IndexColorModel object in lieu of pixel by pixel filtering.

Interface Index

newmodel

Interface Index

origmodel

Interface Index

Interface Index

RGBImageFilter()

Interface Index

Interface Index

filterIndexColorModel(IndexColorModel)

Filters an IndexColorModel object by running each entry in its color tables through the filterRGB function that RGBImageFilter subclasses must provide.

Interface Index

filterRGB(int, int, int)

Subclasses must specify a method to convert a single input pixel in the default RGB ColorModel to a single output pixel.

Interface Index

filterRGBPixels(int, int, int, int, int[], int, int)

Filters a buffer of pixels in the default RGB ColorModel by passing them one by one through the filterRGB method.

Interface Index

setColorModel(ColorModel)

If the ColorModel is an IndexColorModel, and the subclass has set the canFilterIndexColorModel flag to true, we substitute a filtered version of the color model here and wherever that original ColorModel object appears in the setPixels methods.

Interface Index

setPixels(int, int, int, int, ColorModel, byte[], int, int)

If the ColorModel object is the same one that has already been converted, then simply passes the pixels through with the converted ColorModel.

Interface Index

setPixels(int, int, int, int, ColorModel, int[], int, int)

If the ColorModel object is the same one that has already been converted, then simply passes the pixels through with the converted ColorModel, otherwise converts the buffer of integer pixels to the default RGB ColorModel and passes the converted buffer to the filterRGBPixels method to be converted one by one.

Interface Index

substituteColorModel(ColorModel, ColorModel)

Registers two ColorModel objects for substitution.

Interface Index

Interface Index

origmodel

protected ColorModel origmodel

Interface Index

newmodel

protected ColorModel newmodel

Interface Index

canFilterIndexColorModel

protected boolean canFilterIndexColorModel

This boolean indicates whether or not it is acceptable to apply the color filtering of the filterRGB method to the color table entries of an IndexColorModel object in lieu of pixel by pixel filtering. Subclasses should set this variable to true in their constructor if their filterRGB method does not depend on the coordinate of the pixel being filtered.

See Also:

substituteColorModel, filterRGB, IndexColorModel

Interface Index

Interface Index

RGBImageFilter

```
public RGBImageFilter()
```

Interface Index

Interface Index

setColorModel

public void setColorModel(ColorModel model)

If the ColorModel is an IndexColorModel, and the subclass has set the canFilterIndexColorModel flag to true, we substitute a filtered version of the color model here and wherever that original ColorModel object appears in the setPixels methods. Otherwise overrides the default ColorModel used by the ImageProducer and specifies the default RGB ColorModel instead.

Overrides:

setColorModel in class ImageFilter

See Also:

ImageConsumer, getRGBdefault

Interface Index

substituteColorModel

public void substituteColorModel(ColorModel oldcm,
ColorModel newcm)

Registers two ColorModel objects for substitution. If the oldcm is encountered during any of the setPixels methods, the newcm is substituted and the pixels passed through untouched (but with the new ColorModel object).

Parameters:

oldcm - the ColorModel object to be replaced on the fly

newcm - the ColorModel object to replace oldcm on the fly

Interface Index

filterIndexColorModel

```
public IndexColorModel filterIndexColorModel(IndexColorModel icm)
```

Filters an IndexColorModel object by running each entry in its color tables through the filterRGB function that RGBImageFilter subclasses must provide. Uses coordinates of -1 to indicate that a color table entry is being filtered rather than an actual pixel value.

Parameters:

icm - the IndexColorModel object to be filtered

Returns:

a new IndexColorModel representing the filtered colors

Interface Index

filterRGBPixels

```
public void filterRGBPixels(int x,
                            int y,
                            int w,
                            int h,
                            int pixels[],
                            int off,
                            int scansize)
```

Filters a buffer of pixels in the default RGB ColorModel by passing
them one by one through the filterRGB method.

See Also:

getRGBdefault, filterRGB

Interface Index

setPixels

```
public void setPixels(int x,
                      int y,
                      int w,
                      int h,
                      ColorModel model,
                      byte pixels[],
                      int off,
                      int scansize)
```

If the ColorModel object is the same one that has already been converted, then simply passes the pixels through with the
converted ColorModel. Otherwise converts the buffer of byte pixels to the default RGB ColorModel and passes the converted
buffer to the filterRGBPixels method to be converted one by one.

Overrides:

setPixels in class ImageFilter

See Also:

getRGBdefault, filterRGBPixels

Interface Index

setPixels

```
public void setPixels(int x,
                      int y,
                      int w,
                      int h,
                      ColorModel model,
                      int pixels[],
                      int off,
                      int scansize)
```

If the ColorModel object is the same one that has already
been converted, then simply passes the pixels through with the
converted ColorModel, otherwise converts the buffer of integer
pixels to the default RGB ColorModel and passes the converted
buffer to the filterRGBPixels method to be converted one by one.
Converts a buffer of integer pixels to the default RGB ColorModel
and passes the converted buffer to the filterRGBPixels method.

Overrides:

setPixels in class ImageFilter

See Also:

getRGBdefault, filterRGBPixels

Interface Index

filterRGB

```
public abstract int filterRGB(int x,
```

```
        int y,
        int rgb)
```

Subclasses must specify a method to convert a single input pixel
in the default RGB ColorModel to a single output pixel.

See Also:

getRGBdefault, filterRGBPixels

Class java.awt.Insets

Class java.awt.Insets

```
java.lang.Object
    |
    +----java.awt.Insets
```

---

public class **Insets**
extends Object
implements Cloneable

The insets of a container. This class is used to layout containers.
See Also:
LayoutManager, Container

bottom

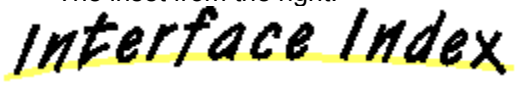
The inset from the bottom.

left

The inset from the left.

right

The inset from the right.



top

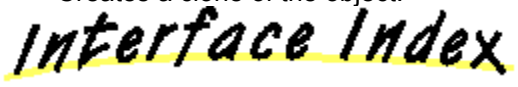
The inset from the top.

Insets(int, int, int, int)

Constructs and initializes a new Inset with the specified top, left, bottom, and right insets.

clone()

Creates a clone of the object.



toString()

Returns a String object representing this Inset's values.

top

```
public int top
```

The inset from the top.

left

```
public int left
```

The inset from the left.

bottom

```
public int bottom
```

The inset from the bottom.

right

```
public int right
```

The inset from the right.

Interface Index

Interface Index **Insets**

```
public Insets(int top,
              int left,
              int bottom,
              int right)
```

Constructs and initializes a new Inset with the specified top, left, bottom, and right insets.
Parameters:
top - the inset from the top
left - the inset from the left
bottom - the inset from the bottom
right - the inset from the right

Interface Index

Interface Index **toString**

```
public String toString()
```

Returns a String object representing this Inset's values.
Overrides:
toString in class Object

Interface Index **clone**

```
public Object clone()
```

Creates a clone of the object.
Overrides:
clone in class Object

Class java.awt.Label

Class java.awt.Label

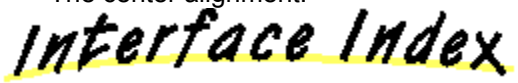
```
java.lang.Object
    |
    +----java.awt.Component
            |
            +----java.awt.Label
```

---

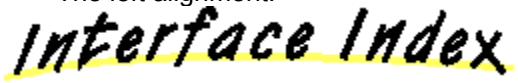
public class **Label**
extends Component

A component that displays a single line of read-only text.

CENTER
The center alignment.

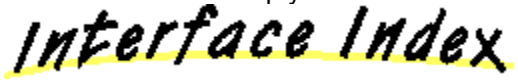


LEFT
The left alignment.



RIGHT
The right alignment.

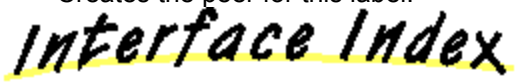
Label()
Constructs an empty label.



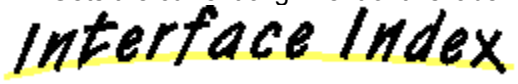
Label(String)
Constructs a new label with the specified String of text.

Label(String, int)
Constructs a new label with the specified String of text and the specified alignment.

addNotify()
Creates the peer for this label.

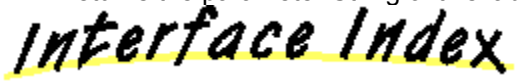


getAlignment()
Gets the current alignment of this label.

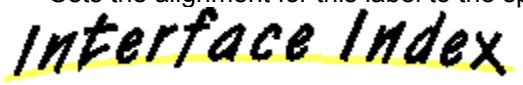


getText()
Gets the text of this label.

paramString()
Returns the parameter String of this label.



setAlignment(int)
Sets the alignment for this label to the specified alignment.



setText(String)
Sets the text for this label to the specified text.

LEFT

```
public final static int LEFT
```

The left alignment.

CENTER

```
public final static int CENTER
```

The center alignment.

Interface Index

RIGHT

```
public final static int RIGHT
```

The right alignment.

Interface Index

Interface Index

Label

```
public Label()
```

Constructs an empty label.

Interface Index

Label

```
public Label(String label)
```

Constructs a new label with the specified String of text.

Parameters:

label - the text that makes up the label

Interface Index

Label

```
public Label(String label,
             int alignment)
```

Constructs a new label with the specified String of text and the specified alignment.

Parameters:

label - the String that makes up the label

alignment - the alignment value

Interface Index

addNotify

```
public synchronized void addNotify()
```

Creates the peer for this label. The peer allows us to modify the appearance of the label without changing its functionality.

Overrides:

addNotify in class Component

getAlignment

```
public int getAlignment()
```

Gets the current alignment of this label.

See Also:

setAlignment

setAlignment

```
public void setAlignment(int alignment)
```

Sets the alignment for this label to the specified alignment.

Parameters:

alignment - the alignment value

Throws: IllegalArgumentException

If an improper alignment was given.

See Also:

getAlignment

getText

```
public String getText()
```

Gets the text of this label.

See Also:

setText

setText

```
public void setText(String label)
```

Sets the text for this label to the specified text.

Parameters:

label - the text that makes up the label

See Also:

getText

paramString

```
protected String paramString()
```

Returns the parameter String of this label.

Overrides:

paramString in class Component

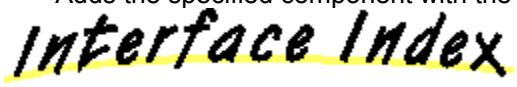
Interface java.awt.LayoutManager

Interface java.awt.LayoutManager

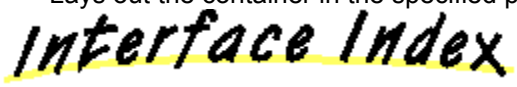
public interface **LayoutManager**
extends Object

Defines the interface for classes that know how to layout Containers.
See Also:
Container

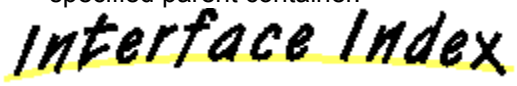
addLayoutComponent(String, Component)
Adds the specified component with the specified name to the layout.



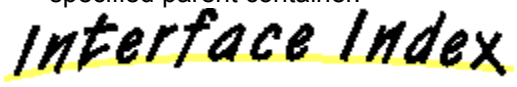
layoutContainer(Container)
Lays out the container in the specified panel.



minimumLayoutSize(Container)
Calculates the minimum size dimensions for the specified panel given the components in the specified parent container.



preferredLayoutSize(Container)
Calculates the preferred size dimensions for the specified panel given the components in the specified parent container.



removeLayoutComponent(Component)
Removes the specified component from the layout.

addLayoutComponent

```
public abstract void addLayoutComponent(String name,
                                        Component comp)
```

Adds the specified component with the specified name to the layout.
Parameters:
name - the component name
comp - the component to be added

removeLayoutComponent

```
public abstract void removeLayoutComponent(Component comp)
```

Removes the specified component from the layout.
Parameters:
comp - the component ot be removed

preferredLayoutSize

```
public abstract Dimension preferredLayoutSize(Container parent)
```

Calculates the preferred size dimensions for the specified panel given the components in the specified parent container.
Parameters:
parent - the component to be laid out
See Also:
minimumLayoutSize

minimumLayoutSize

```
public abstract Dimension minimumLayoutSize(Container parent)
```

Calculates the minimum size dimensions for the specified panel given the components in the specified parent container.
Parameters:
parent - the component to be laid out
See Also:
preferredLayoutSize

layoutContainer

```
public abstract void layoutContainer(Container parent)
```

Lays out the container in the specified panel.

Parameters:

parent - the component which needs to be laid out

Class java.awt.List

Class java.awt.List

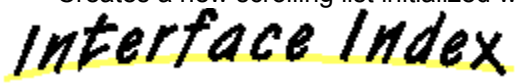
```
java.lang.Object
    |
    +----java.awt.Component
            |
            +----java.awt.List
```

---

public class **List**
extends Component

A scrolling list of text items.

List()
Creates a new scrolling list initialized with no visible Lines or multiple selections.



List(int, boolean)
Creates a new scrolling list initialized with the specified number of visible lines and a boolean stating whether multiple selections are allowed or not.

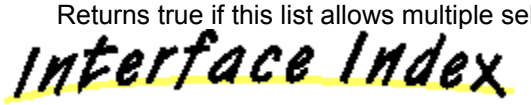
addItem(String)
Adds the specified item to the end of scrolling list.

addItem(String, int)
Adds the specified item to the end of scrolling list.

addNotify()
Creates the peer for the list.

allowsMultipleSelections()

Returns true if this list allows multiple selections.



clear()

Clears the list.

countItems()

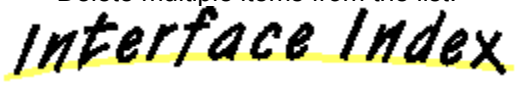
Returns the number of items in the list.

delItem(int)

Delete an item from the list.

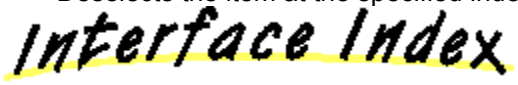
delItems(int, int)

Delete multiple items from the list.



deselect(int)

Deselects the item at the specified index.



getItem(int)

Gets the item associated with the specified index.

getRows()

Returns the number of visible lines in this list.

getSelectedIndex()

Get the selected item on the list or -1 if no item is selected.

getSelectedIndexes()

Returns the selected indexes on the list.

getSelectedItem()

Returns the selected item on the list or null if no item is selected.

getSelectedItems()

Returns the selected items on the list.

getVisibleIndex()

Gets the index of the item that was last made visible by the method makeVisible.

isSelected(int)

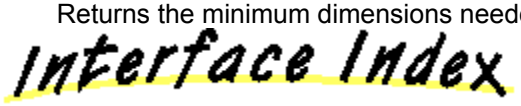
Returns true if the item at the specified index has been selected; false otherwise.

makeVisible(int)

Forces the item at the specified index to be visible.

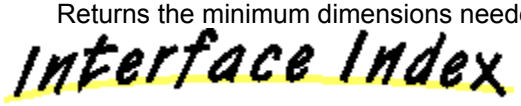
minimumSize(int)

Returns the minimum dimensions needed for the amount of rows in the list.



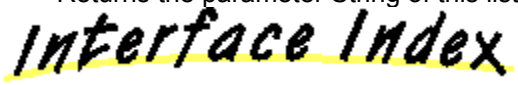
minimumSize()

Returns the minimum dimensions needed for the list.



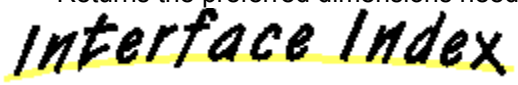
paramString()

Returns the parameter String of this list.



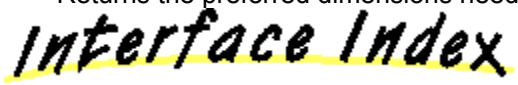
preferredSize(int)

Returns the preferred dimensions needed for the list with the specified amount of rows.



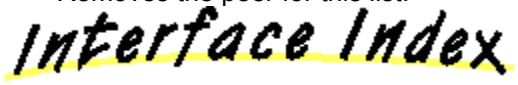
preferredSize()

Returns the preferred dimensions needed for the list.



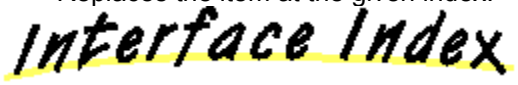
removeNotify()

Removes the peer for this list.



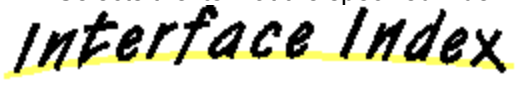
replaceItem(String, int)

Replaces the item at the given index.



select(int)

Selects the item at the specified index.



setMultipleSelections(boolean)

Sets whether this list should allow multiple selections or not.

List

```
public List()
```

Creates a new scrolling list initialized with no visible Lines or multiple selections.

List

```
public List(int rows,
            boolean multipleSelections)
```

Creates a new scrolling list initialized with the specified number of visible lines and a boolean stating

whether multiple selections are allowed or not.
Parameters:
rows - the number of items to show.
multipleSelections - if true then multiple selections are allowed.

addNotify

```
public synchronized void addNotify()
```

Creates the peer for the list. The peer allows us to modify the list's appearance without changing its functionality.
Overrides:
addNotify in class Component

removeNotify

```
public synchronized void removeNotify()
```

Removes the peer for this list. The peer allows us to modify the list's appearance without changing its functionality.
Overrides:
removeNotify in class Component

countItems

```
public int countItems()
```

Returns the number of items in the list.
See Also:
getItem

getItem

```
public String getItem(int index)
```

Gets the item associated with the specified index.

Parameters:

index - the position of the item

See Also:

countItems

addItem

```
public synchronized void addItem(String item)
```

Adds the specified item to the end of scrolling list.

Parameters:

item - the item to be added

addItem

```
public synchronized void addItem(String item,
                                 int index)
```

Adds the specified item to the end of scrolling list.

Parameters:

item - the item to be added

index - the position at which to put in the item. The index is zero-based. If index is -1 then the item is added to the end. If index is greater than the number of items in the list, the item gets added at the end.

replaceItem

```
public synchronized void replaceItem(String newValue,
                                     int index)
```

Replaces the item at the given index.

Parameters:

newValue - the new value to replace the existing item

index - the position of the item to replace

clear

```
public synchronized void clear()
```

Clears the list.

See Also:

delItem, delItems

delItem

```
public synchronized void delItem(int position)
```

Delete an item from the list.

delItems

```
public synchronized void delItems(int start,
                                  int end)
```

Delete multiple items from the list.

getSelectedIndex

```
public synchronized int getSelectedIndex()
```

Get the selected item on the list or -1 if no item is selected.

See Also:

select, deselect, isSelected

getSelectedIndexes

```
public synchronized int[] getSelectedIndexes()
```

Returns the selected indexes on the list.

See Also:

select, deselect, isSelected

 getSelectedItem

```
public synchronized String getSelectedItem()
```

Returns the selected item on the list or null if no item is selected.

See Also:

select, deselect, isSelected

 getSelectedItems

```
public synchronized String[] getSelectedItems()
```

Returns the selected items on the list.

See Also:

select, deselect, isSelected

 select

```
public synchronized void select(int index)
```

Selects the item at the specified index.

Parameters:

index - the position of the item to select

See Also:

getSelectedItem, deselect, isSelected

 deselect

```
public synchronized void deselect(int index)
```

Deselects the item at the specified index.

Parameters:

index - the position of the item to deselect

See Also:

select, getSelectedItem, isSelected

isSelected

```
public synchronized boolean isSelected(int index)
```

Returns true if the item at the specified index has been selected; false otherwise.

Parameters:

index - the item to be checked

See Also:

select, deselect, isSelected

getRows

```
public int getRows()
```

Returns the number of visible lines in this list.

allowsMultipleSelections

```
public boolean allowsMultipleSelections()
```

Returns true if this list allows multiple selections.

See Also:

setMultipleSelections

setMultipleSelections

```
public void setMultipleSelections(boolean v)
```

Sets whether this list should allow multiple selections or not.

Parameters:

v - the boolean to allow multiple selections

See Also:

allowsMultipleSelections

getVisibleIndex

```
public int getVisibleIndex()
```

Gets the index of the item that was last made visible by the method makeVisible.

makeVisible

```
public void makeVisible(int index)
```

Forces the item at the specified index to be visible.

Parameters:

index - the position of the item

See Also:

getVisibleIndex

preferredSize

```
public Dimension preferredSize(int rows)
```

Returns the preferred dimensions needed for the list with the specified amount of rows.

Parameters:

rows - amount of rows in list.

preferredSize

```
public Dimension preferredSize()
```

Returns the preferred dimensions needed for the list.

Returns:

the preferred size with the specified number of rows if the row size is greater than 0.

Overrides:

preferredSize in class Component

minimumSize

```
public Dimension minimumSize(int rows)
```

Returns the minimum dimensions needed for the amount of rows in the list.

Parameters:

rows - minimum amount of rows in the list

minimumSize

```
public Dimension minimumSize()
```

Returns the minimum dimensions needed for the list.

Returns:

the preferred size with the specified number of rows if the row size is greater than zero.

Overrides:

minimumSize in class Component

paramString

```
protected String paramString()
```

Returns the parameter String of this list.

Overrides:

paramString in class Component

Class java.awt.MediaTracker

Class java.awt.MediaTracker

```
java.lang.Object
   |
   +----java.awt.MediaTracker
```

---

public class **MediaTracker**
extends Object

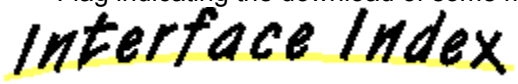
A utility class to track the status of a number of media objects. Media objects could include images as well as audio clips, though currently only images are supported. To use it, simply create an instance and then call addImage() for each image to be tracked. Each image can be assigned a unique ID for indentification purposes. The IDs control the priority order in which the images are fetched as well as identifying unique subsets of the images that can be waited on independently. Here is an example:

```
import java.applet.Applet;
import java.awt.Color;
import java.awt.Image;
import java.awt.Graphics;
import java.awt.MediaTracker;
public class ImageBlaster extends Applet implements Runnable {
        MediaTracker tracker;
        Image bg;
        Image anim[] = new Image[5];
        int index;
        Thread animator;
        // Get the images for the background (id == 0) and the animation
        // frames (id == 1) and add them to the MediaTracker
        public void init() {
            tracker = new MediaTracker(this);
            bg = getImage(getDocumentBase(), "images/background.gif");
            tracker.addImage(bg, 0);
            for (int i = 0; i = anim.length) {
                        index = 0;
                    }
                }
                repaint();
            }
        }
        // The background image fills our frame so we don't need to clear
        // the applet on repaints, just call the paint method.
        public void update(Graphics g) {
            paint(g);
        }
```

```
        // Paint a large red rectangle if there are any errors loading the
        // images.  Otherwise always paint the background so that it appears
        // incrementally as it is loading.  Finally, only paint the current
        // animation frame if all of the frames (id == 1) are done loading
        // so that we don't get partial animations.
        public void paint(Graphics g) {
            if ((tracker.statusAll() &MediaTracker.ERRORED) != 0) {
                g.setColor(Color.red);
                g.fillRect(0, 0, size().width, size().height);
                return;
            }
            g.drawImage(bg, 0, 0, this);
            if ((tracker.statusID(1) &MediaTracker.COMPLETE) != 0) {
                g.drawImage(anim[index], 10, 10, this);
            }
        }
}
```

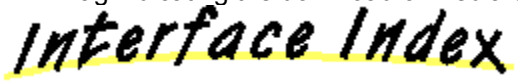
ABORTED

Flag indicating the download of some media was aborted.



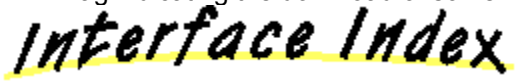
COMPLETE

Flag indicating the download of media completed successfully.



ERRORED

Flag indicating the download of some media encountered an error.



LOADING

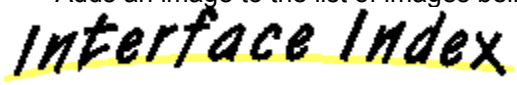
Flag indicating some media is currently being loaded.

MediaTracker(Component)

Creates a Media tracker to track images for a given Component.

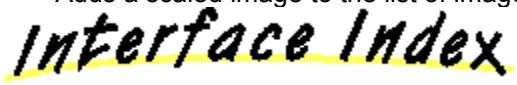
addImage(Image, int)

Adds an image to the list of images being tracked.



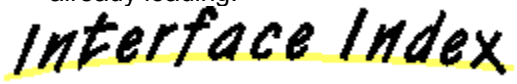
addImage(Image, int, int, int)

Adds a scaled image to the list of images being tracked.



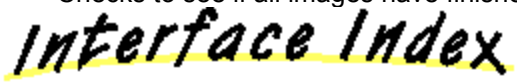
checkAll()

Checks to see if all images have finished loading but does not start loading the images if they are not already loading.



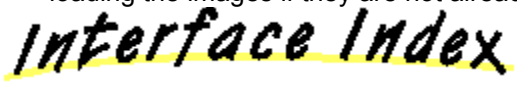
checkAll(boolean)

Checks to see if all images have finished loading.



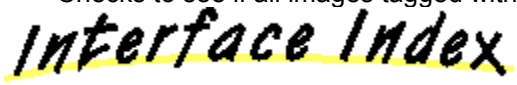
checkID(int)

Checks to see if all images tagged with the indicated ID have finished loading, but does not start loading the images if they are not already loading.



checkID(int, boolean)

Checks to see if all images tagged with the indicated ID have finished loading.

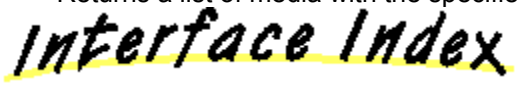


getErrorsAny()

Returns a list of all media that have encountered an error.

getErrorsID(int)

Returns a list of media with the specified ID that have encountered an error.



isErrorAny()

Checks the error status of all of the images.

isErrorID(int)

Checks the error status of all of the images with the specified ID.

statusAll(boolean)

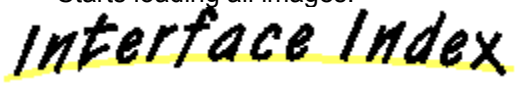
Returns the boolean OR of the status of all of the media being tracked.

statusID(int, boolean)

Returns the boolean OR of the status of all of the media with a given ID.

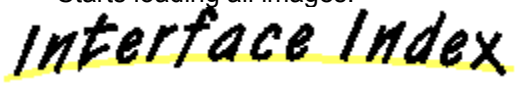
waitForAll()

Starts loading all images.



waitForAll(long)

Starts loading all images.



waitForID(int)

Starts loading all images with the specified ID and waits until they have finished loading or receive an error.

Interface Index **waitForID**(int, long)

Starts loading all images with the specified ID.

Interface Index

Interface Index **LOADING**

```
public final static int LOADING
```

Flag indicating some media is currently being loaded.

See Also:

statusAll, statusID

Interface Index **ABORTED**

```
public final static int ABORTED
```

Flag indicating the download of some media was aborted.

See Also:

statusAll, statusID

Interface Index **ERRORED**

```
public final static int ERRORED
```

Flag indicating the download of some media encountered an error.

See Also:

statusAll, statusID

Interface Index **COMPLETE**

```
public final static int COMPLETE
```

Flag indicating the download of media completed successfully.

See Also:

statusAll, statusID

Interface Index

Interface Index **MediaTracker**

```
public MediaTracker(Component comp)
```

Creates a Media tracker to track images for a given Component.
Parameters:
comp - the component on which the images will eventually be drawn

Interface Index

Interface Index **addImage**

```
public void addImage(Image image,
                     int id)
```

Adds an image to the list of images being tracked. The image will eventually be rendered at its default (unscaled) size.
Parameters:
image - the image to be tracked
id - the identifier used to later track this image

Interface Index **addImage**

```
public synchronized void addImage(Image image,
                                  int id,
                                  int w,
                                  int h)
```

Adds a scaled image to the list of images being tracked. The image will eventually be rendered at the indicated size.
Parameters:
image - the image to be tracked
id - the identifier used to later track this image

w - the width that the image will be rendered at
h - the height that the image will be rendered at

Interface Index **checkAll**

```
public boolean checkAll()
```

Checks to see if all images have finished loading but does not start loading the images if they are not already loading. If there is an error while loading or scaling an image then that image is considered "complete." Use isErrorAny() or isErrorID() to check for errors.

Returns:

true if all images have finished loading, were aborted or encountered an error

See Also:

checkAll, checkID, isErrorAny, isErrorID

Interface Index **checkAll**

```
public synchronized boolean checkAll(boolean load)
```

Checks to see if all images have finished loading. If load is true, starts loading any images that are not yet being loaded. If there is an error while loading or scaling an image then that image is considered "complete." Use isErrorAny() or isErrorID() to check for errors.

Parameters:

load - start loading the images if this parameter is true

Returns:

true if all images have finished loading, were aborted or encountered an error

See Also:

isErrorAny, isErrorID, checkID, checkAll

Interface Index **isErrorAny**

```
public synchronized boolean isErrorAny()
```

Checks the error status of all of the images.

Returns:

true if any of the images had an error during loading

See Also:

isErrorID, getErrorsAny

getErrorsAny

```
public synchronized Object[] getErrorsAny()
```

Returns a list of all media that have encountered an error.

Returns:

an array of media objects or null if there are no errors

See Also:

isErrorAny, getErrorsID

waitForAll

```
public void waitForAll() throws InterruptedException
```

Starts loading all images. Waits until they have finished loading, are aborted, or it receives an error. If there is an error while loading or scaling an image then that image is considered "complete." Use isErrorAny() or statusAll() to check for errors.

Throws: InterruptedException

Another thread has interrupted this thread.

See Also:

waitForID, waitForAll, isErrorAny, isErrorID

waitForAll

```
public synchronized boolean waitForAll(long ms) throws InterruptedException
```

Starts loading all images. Waits until they have finished loading, are aborted, it receives an error, or until the specified timeout has elapsed. If there is an error while loading or scaling an image then that image is considered "complete." Use isErrorAny() or statusAll() to check for errors.

Parameters:

ms - the length of time to wait for the loading to complete

Returns:

true if all images were successfully loaded

Throws: InterruptedException

Another thread has interrupted this thread.

See Also:

waitForID, waitForAll, isErrorAny, isErrorID

statusAll

```
public int statusAll(boolean load)
```

Returns the boolean OR of the status of all of the media being tracked.

Parameters:

load - specifies whether to start the media loading

See Also:

statusID, LOADING, ABORTED, ERRORED, COMPLETE

checkID

```
public boolean checkID(int id)
```

Checks to see if all images tagged with the indicated ID have finished loading, but does not start loading the images if they are not already loading. If there is an error while loading or scaling an image then that image is considered "complete." Use isErrorAny() or isErrorID() to check for errors.

Parameters:

id - the identifier used to determine which images to check

Returns:

true if all tagged images have finished loading, were aborted, or an error occurred.

See Also:

checkID, checkAll, isErrorAny, isErrorID

checkID

```
public synchronized boolean checkID(int id,
                                    boolean load)
```

Checks to see if all images tagged with the indicated ID have finished loading. If load is true, starts loading any images with that ID that are not yet being loaded. If there is an error while loading or scaling an image then that image is considered "complete." Use isErrorAny() or isErrorID() to check for errors.

Parameters:

id - the identifier used to determine which images to check

load - start loading the images if this parameter is true

Returns:

true if all tagged images have finished loading, were aborted, or an error occurred

See Also:

checkID, checkAll, isErrorAny, isErrorID

isErrorID

```
public synchronized boolean isErrorID(int id)
```

Checks the error status of all of the images with the specified ID.

Parameters:

id - the identifier used to determine which images to check

Returns:

true if any of the tagged images had an error during loading

See Also:

isErrorAny, getErrorsID

getErrorsID

```
public synchronized Object[] getErrorsID(int id)
```

Returns a list of media with the specified ID that have encountered an error.

Parameters:

id - the identifier used to determine which images to return

Returns:

an array of media objects or null if there are no errors

See Also:

isErrorID, getErrorsAny

waitForID

```
public void waitForID(int id) throws InterruptedException
```

Starts loading all images with the specified ID and waits until they have finished loading or receive an error. If there is an error while loading or scaling an image then that image is considered "complete." Use statusID() or isErrorID() to check for errors.

Parameters:

id - the identifier used to determine which images to wait for

Throws: InterruptedException

Another thread has interrupted this thread.

See Also:

waitForAll, waitForID, isErrorAny, isErrorID

waitForID

```
public synchronized boolean waitForID(int id,
                                      long ms) throws InterruptedException
```

Starts loading all images with the specified ID. Waits until they have finished loading, an error occurs,

or the specified timeout has elapsed. If there is an error while loading or scaling an image then that image is considered "complete." Use statusID or isErrorID to check for errors.

Parameters:

id - the identifier used to determine which images to wait for

ms - the length of time to wait for the loading to complete

Throws: InterruptedException

Another thread has interrupted this thread.

See Also:

waitForAll, waitForID, isErrorAny, isErrorID

statusID

```
public int statusID(int id,
                    boolean load)
```

Returns the boolean OR of the status of all of the media with a given ID.

Parameters:

id - the identifier used to determine which images to check

load - specifies whether to start the media loading

See Also:

statusAll, LOADING, ABORTED, ERRORED, COMPLETE

Class java.awt.Menu

Class java.awt.Menu

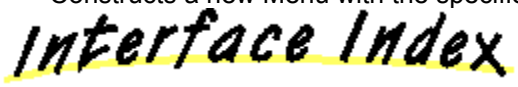
```
java.lang.Object
    |
    +----java.awt.MenuComponent
            |
            +----java.awt.MenuItem
                    |
                    +----java.awt.Menu
```

---

public class **Menu**
extends MenuItem
implements MenuContainer

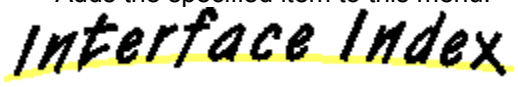
A Menu that is a component of a menu bar.

Menu(String)
Constructs a new Menu with the specified label.



Menu(String, boolean)
Constructs a new Menu with the specified label.

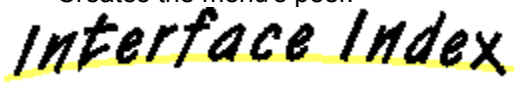
add(MenuItem)
Adds the specified item to this menu.



add(String)
Adds an item with with the specified label to this menu.

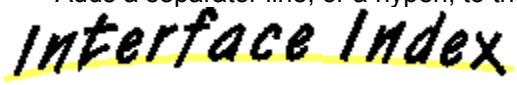
addNotify()

Creates the menu's peer.



addSeparator()

Adds a separator line, or a hypen, to the menu at the current position.

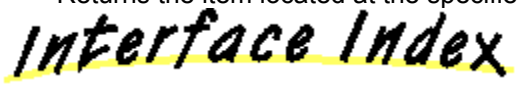


countItems()

Returns the number of elements in this menu.

getItem(int)

Returns the item located at the specified index of this menu.



isTearOff()

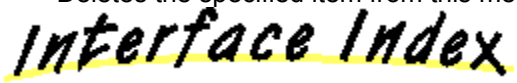
Returns true if this is a tear-off menu.

remove(int)

Deletes the item from this menu at the specified index.

remove(MenuComponent)

Deletes the specified item from this menu.



removeNotify()

Removes the menu's peer.

Menu

```
public Menu(String label)
```

Constructs a new Menu with the specified label. This menu can not be torn off - the menu will still appear on screen after the the mouse button has been released.

Parameters:

label - the label to be added to this menu

Menu

```
public Menu(String label,
            boolean tearOff)
```

Constructs a new Menu with the specified label. If tearOff is true, the menu can be torn off - the menu will still appear on screen after the the mouse button has been released.

Parameters:
label - the label to be added to this menu
tearOff - the boolean indicating whether or not the menu will be able to be torn off.

addNotify

```
public synchronized void addNotify()
```

Creates the menu's peer. The peer allows us to modify the appearance of the menu without changing its functionality.
Overrides:
addNotify in class MenuItem

removeNotify

```
public synchronized void removeNotify()
```

Removes the menu's peer. The peer allows us to modify the appearance of the menu without changing its functionality.
Overrides:
removeNotify in class MenuComponent

isTearOff

```
public boolean isTearOff()
```

Returns true if this is a tear-off menu.

countItems

```
public int countItems()
```

Returns the number of elements in this menu.

getItem

```
public MenuItem getItem(int index)
```

Returns the item located at the specified index of this menu.
Parameters:
index - the position of the item to be returned

add

```
public synchronized MenuItem add(MenuItem mi)
```

Adds the specified item to this menu.
Parameters:
mi - the item to be added

add

```
public void add(String label)
```

Adds an item with with the specified label to this menu.
Parameters:
label - the text on the item

addSeparator

```
public void addSeparator()
```

Adds a separator line, or a hypen, to the menu at the current position.

remove

```
public synchronized void remove(int index)
```

Deletes the item from this menu at the specified index.
Parameters:
index - the position of the item to be removed

remove

```
public synchronized void remove(MenuComponent item)
```

Deletes the specified item from this menu.

Parameters:

item - the item to be removed from the menu

Class java.awt.MenuBar

Class java.awt.MenuBar

```
java.lang.Object
    |
    +----java.awt.MenuComponent
            |
            +----java.awt.MenuBar
```

---

public class **MenuBar**
extends MenuComponent
implements MenuContainer

A class that encapsulates the platform's concept of a menu bar bound to a Frame. In order to associate the MenuBar with an actual Frame, the Frame.setMenuBar() method should be called.

See Also:

setMenuBar

MenuBar()

Creates a new menu bar.

add(Menu)

Adds the specified menu to the menu bar.

addNotify()

Creates the menu bar's peer.

countMenus()

Counts the number of menus on the menu bar.

Interface Index **getHelpMenu**()
Gets the help menu on the menu bar.
Interface Index **getMenu**(int)
Gets the specified menu.
Interface Index **remove**(int)
Removes the menu located at the specified index from the menu bar.
Interface Index **remove**(MenuComponent)
Removes the specified menu from the menu bar.
Interface Index **removeNotify**()
Removes the menu bar's peer.
Interface Index **setHelpMenu**(Menu)
Sets the help menu to the specified menu on the menu bar.

Interface Index

Interface Index **MenuBar**

```
public MenuBar()
```

Creates a new menu bar.

Interface Index

Interface Index **addNotify**

```
public synchronized void addNotify()
```

Creates the menu bar's peer. The peer allows us to change the appearance of the menu bar without changing any of the menu bar's functionality.

Interface Index **removeNotify**

```
public void removeNotify()
```

Removes the menu bar's peer. The peer allows us to change the appearance of the menu bar without changing any of the menu bar's functionality.
Overrides:
removeNotify in class MenuComponent

getHelpMenu

```
public Menu getHelpMenu()
```

Gets the help menu on the menu bar.

setHelpMenu

```
public synchronized void setHelpMenu(Menu m)
```

Sets the help menu to the specified menu on the menu bar.
Parameters:
m - the menu to be set

add

```
public synchronized Menu add(Menu m)
```

Adds the specified menu to the menu bar.
Parameters:
m - the menu to be added to the menu bar

remove

```
public synchronized void remove(int index)
```

Removes the menu located at the specified index from the menu bar.
Parameters:
index - the position of the menu to be removed

remove

```
public synchronized void remove(MenuComponent m)
```

Removes the specified menu from the menu bar.

Parameters:

m - the menu to be removed

countMenus

```
public int countMenus()
```

Counts the number of menus on the menu bar.

getMenu

```
public Menu getMenu(int i)
```

Gets the specified menu.

Parameters:

i - the menu to be returned

Class java.awt.MenuComponent

Class java.awt.MenuComponent

```
java.lang.Object
    |
    +----java.awt.MenuComponent
```

---

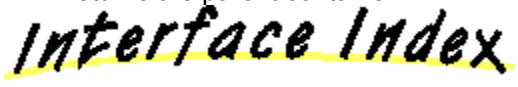
public class **MenuComponent**
extends Object

The super class of all menu related components.

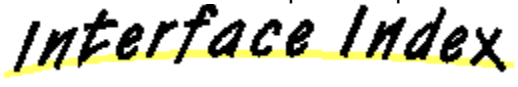
MenuComponent()

getFont()
Gets the font used for this MenuItem.

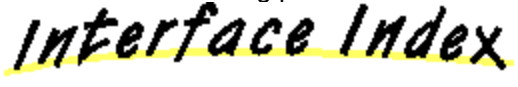
getParent()
Returns the parent container.



getPeer()
Gets the MenuComponent's peer.



paramString()
Returns the String parameter of this MenuComponent.



postEvent(Event)
Posts the specified event to the menu.

removeNotify()

Removes the menu component's peer.

setFont(Font)

Sets the font to be used for this MenuItem to the specified font.

toString()

Returns the String representation of this MenuComponent's values.

MenuComponent

```
public MenuComponent()
```

getParent

```
public MenuContainer getParent()
```

Returns the parent container.

getPeer

```
public MenuComponentPeer getPeer()
```

Gets the MenuComponent's peer. The peer allows us to modify the appearance of the menu component without changing the functionality of the menu component.

getFont

```
public Font getFont()
```

Gets the font used for this MenuItem.

Returns:

the font if one is used; null otherwise.

setFont

```
public void setFont(Font f)
```

Sets the font to be used for this MenuItem to the specified font.
Parameters:
f - the font to be set

removeNotify

```
public void removeNotify()
```

Removes the menu component's peer. The peer allows us to modify the appearance of the menu component without changing the functionality of the menu component.

postEvent

```
public boolean postEvent(Event evt)
```

Posts the specified event to the menu.
Parameters:
evt - the event which is to take place

paramString

```
protected String paramString()
```

Returns the String parameter of this MenuComponent.

toString

```
public String toString()
```

Returns the String representation of this MenuComponent's values.
Overrides:
toString in class Object

Interface java.awt.MenuContainer

Interface java.awt.MenuContainer

public interface **MenuContainer**
extends Object

The super class of all menu related containers.

getFont()

postEvent(Event)

remove(MenuComponent)

getFont

```
public abstract Font getFont()
```

postEvent

```
public abstract boolean postEvent(Event evt)
```

remove

```
public abstract void remove(MenuComponent comp)
```

Class java.awt.MenuItem

Class java.awt.MenuItem

```
java.lang.Object
    |
    +----java.awt.MenuComponent
             |
             +----java.awt.MenuItem
```

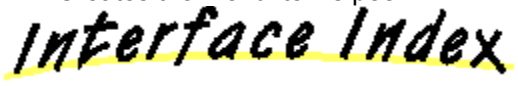
---

public class **MenuItem**
extends MenuComponent

A String item that represents a choice in a menu.

MenuItem(String)
Constructs a new MenuItem with the specified label.

addNotify()
Creates the menu item's peer.



disable()
Makes this menu item unselectable by the user.

enable()
Makes this menu item selectable by the user.

enable(boolean)
Conditionally enables a component.

Interface Index **getLabel**()
Gets the label for this menu item.
Interface Index **isEnabled**()
Checks whether the menu item is enabled.
Interface Index **paramString**()
Returns the String parameter of the menu item.
Interface Index **setLabel**(String)
Sets the label to be the specified label.

Interface Index

Interface Index **MenuItem**

```
public MenuItem(String label)
```

Constructs a new MenuItem with the specified label.

Parameters:

label - the label for this menu item. Note that "-" is reserved to mean a separator between menu items.

Interface Index

Interface Index **addNotify**

```
public synchronized void addNotify()
```

Creates the menu item's peer. The peer allows us to modify the appearance of the menu item without changing its functionality.

Interface Index **getLabel**

```
public String getLabel()
```

Gets the label for this menu item.

setLabel

```
public void setLabel(String label)
```

Sets the label to be the specified label.
Parameters:
label - the label for this menu item

isEnabled

```
public boolean isEnabled()
```

Checks whether the menu item is enabled.

enable

```
public void enable()
```

Makes this menu item selectable by the user.

enable

```
public void enable(boolean cond)
```

Conditionally enables a component.
Parameters:
cond - enabled if true; disabled otherwise.
See Also:
enable, disable

disable

```
public void disable()
```

Makes this menu item unselectable by the user.

paramString

```
public String paramString()
```

Returns the String parameter of the menu item.

Overrides:

paramString in class MenuComponent

Class java.awt.Panel

Class java.awt.Panel

```
java.lang.Object
    |
    +----java.awt.Component
            |
            +----java.awt.Container
                    |
                    +----java.awt.Panel
```

---

public class **Panel**
extends Container

A Panel Container class. This produces a generic container.

Panel()

Creates a new panel.

addNotify()

Creates the Panel's peer.

Panel

```
public Panel()
```

Creates a new panel. The default layout for all panels is FlowLayout.

addNotify

```
public synchronized void addNotify()
```

Creates the Panel's peer. The peer allows you to modify the appearance of the panel without changing its functionality.

Overrides:

addNotify in class Container

Interface java.awt.peer.ButtonPeer

Interface java.awt.peer.ButtonPeer

public interface **ButtonPeer**
extends Object
extends ComponentPeer

Interface Index

Interface Index **setLabel**(String)

Interface Index

Interface Index **setLabel**

```
public abstract void setLabel(String label)
```

Interface java.awt.peer.CanvasPeer

Interface java.awt.peer.CanvasPeer

public interface **CanvasPeer**
extends Object
extends ComponentPeer

Interface java.awt.peer.CheckboxMenuItemPeer

Interface java.awt.peer.CheckboxMenuItemPeer

public interface **CheckboxMenuItemPeer**
extends Object
extends MenuItemPeer

setState(boolean)

setState

```
public abstract void setState(boolean t)
```

Interface java.awt.peer.CheckboxPeer

Interface java.awt.peer.CheckboxPeer

public interface **CheckboxPeer**
extends Object
extends ComponentPeer

setState

```
public abstract void setState(boolean state)
```

setCheckboxGroup

```
public abstract void setCheckboxGroup(CheckboxGroup g)
```

setLabel

```
public abstract void setLabel(String label)
```

Interface java.awt.peer.ChoicePeer

Interface java.awt.peer.ChoicePeer

public interface **ChoicePeer**
extends Object
extends ComponentPeer

addItem(String, int)

select(int)

addItem

```
public abstract void addItem(String item,
                             int index)
```

select

```
public abstract void select(int index)
```

Interface java.awt.peer.ComponentPeer

Interface java.awt.peer.ComponentPeer

public interface **ComponentPeer**
extends Object

checkImage(Image, int, int, ImageObserver)

createImage(ImageProducer)

createImage(int, int)

disable()

dispose()

enable()

getColorModel()

getFontMetrics(Font)

getGraphics()

getToolkit()

handleEvent(Event)

hide()

minimumSize()

Interface Index
Interface Index
Interface Index
Interface Index
Interface Index
Interface Index
Interface Index
Interface Index
Interface Index
Interface Index
Interface Index

nextFocus()

paint(Graphics)

preferredSize()

prepareImage (Image, int, int, ImageObserver)

print(Graphics)

repaint (long, int, int, int, int)

requestFocus()

reshape(int, int, int, int)

setBackground(Color)

setFont(Font)

setForeground(Color)

show()

Interface Index
Interface Index

show

```
public abstract void show()
```

Interface Index

hide

```
public abstract void hide()
```

Interface Index

enable

```
public abstract void enable()
```


```
public abstract void disable()
```


```
public abstract void paint(Graphics g)
```


```
public abstract void repaint(long tm,
                             int x,
                             int y,
                             int width,
                             int height)
```


```
public abstract void print(Graphics g)
```


```
public abstract void reshape(int x,
                             int y,
                             int width,
                             int height)
```


```
public abstract boolean handleEvent(Event e)
```


```
public abstract Dimension minimumSize()
```

Interface Index **preferredSize**

```
public abstract Dimension preferredSize()
```

Interface Index **getColorModel**

```
public abstract ColorModel getColorModel()
```

Interface Index **getToolkit**

```
public abstract Toolkit getToolkit()
```

Interface Index **getGraphics**

```
public abstract Graphics getGraphics()
```

Interface Index **getFontMetrics**

```
public abstract FontMetrics getFontMetrics(Font font)
```

Interface Index **dispose**

```
public abstract void dispose()
```

Interface Index **setForeground**

```
public abstract void setForeground(Color c)
```

Interface Index **setBackground**

```
public abstract void setBackground(Color c)
```

Interface Index **setFont**

```
public abstract void setFont(Font f)
```

Interface Index **requestFocus**

```
public abstract void requestFocus()
```

Interface Index **nextFocus**

```
public abstract void nextFocus()
```

Interface Index **createImage**

```
public abstract Image createImage(ImageProducer producer)
```

Interface Index **createImage**

```
public abstract Image createImage(int width,
                                  int height)
```

Interface Index **prepareImage**

```
public abstract boolean prepareImage(Image img,
                                     int w,
                                     int h,
                                     ImageObserver o)
```

Interface Index **checkImage**

```
public abstract int checkImage(Image img,
                               int w,
                               int h,
                               ImageObserver o)
```

Interface java.awt.peer.ContainerPeer

Interface java.awt.peer.ContainerPeer

public interface **ContainerPeer**
extends Object
extends ComponentPeer

insets()

insets

```
public abstract Insets insets()
```

Interface java.awt.peer.DialogPeer

Interface java.awt.peer.DialogPeer

public interface **DialogPeer**
extends Object
extends WindowPeer

setResizable(boolean)

setTitle(String)

setTitle

```
public abstract void setTitle(String title)
```

setResizable

```
public abstract void setResizable(boolean resizeable)
```

Interface java.awt.peer.FileDialogPeer

Interface java.awt.peer.FileDialogPeer

public interface **FileDialogPeer**
extends Object
extends DialogPeer

setDirectory(String)

setFile(String)

setFilenameFilter (FilenameFilter)

setFile

```
public abstract void setFile(String file)
```

setDirectory

```
public abstract void setDirectory(String dir)
```

setFilenameFilter

```
public abstract void setFilenameFilter(FilenameFilter filter)
```

Interface java.awt.peer.FramePeer

Interface java.awt.peer.FramePeer

public interface **FramePeer**
extends Object
extends WindowPeer

setCursor(int)

setIconImage(Image)

setMenuBar(MenuBar)

setResizable(boolean)

setTitle(String)

setTitle

```
public abstract void setTitle(String title)
```

setIconImage

```
public abstract void setIconImage(Image im)
```

setMenuBar

public abstract void setMenuBar(MenuBar mb)

public abstract void setResizable(boolean resizeable)

public abstract void setCursor(int cursorType)

Interface java.awt.peer.LabelPeer

Interface java.awt.peer.LabelPeer

public interface **LabelPeer**
extends Object
extends ComponentPeer

setAlignment(int)

setText(String)

setText

```
public abstract void setText(String label)
```

setAlignment

```
public abstract void setAlignment(int alignment)
```

Interface java.awt.peer.ListPeer

Interface java.awt.peer.ListPeer

public interface **ListPeer**
extends Object
extends ComponentPeer

addItem(String, int)

clear()

delItems(int, int)

deselect(int)

getSelectedIndexes()

makeVisible(int)

minimumSize(int)

preferredSize(int)

select(int)

setMultipleSelections(boolean)

getSelectedIndexes

```
public abstract int[] getSelectedIndexes()
```

addItem

```
public abstract void addItem(String item,
                             int index)
```

delItems

```
public abstract void delItems(int start,
                              int end)
```

clear

```
public abstract void clear()
```

select

```
public abstract void select(int index)
```

deselect

```
public abstract void deselect(int index)
```

makeVisible

```
public abstract void makeVisible(int index)
```

setMultipleSelections

```
public abstract void setMultipleSelections(boolean v)
```

preferredSize

public abstract Dimension preferredSize(int v)

minimumSize

public abstract Dimension minimumSize(int v)

Interface java.awt.peer.MenuBarPeer

Interface java.awt.peer.MenuBarPeer

public interface **MenuBarPeer**
extends Object
extends MenuComponentPeer

addHelpMenu(Menu)

addMenu(Menu)

delMenu(int)

addMenu

```
public abstract void addMenu(Menu m)
```

 delMenu

```
public abstract void delMenu(int index)
```

 addHelpMenu

```
public abstract void addHelpMenu(Menu m)
```

Interface java.awt.peer.MenuComponentPeer

Interface java.awt.peer.MenuComponentPeer

public interface **MenuComponentPeer**
extends Object

dispose()

dispose

```
public abstract void dispose()
```

Interface java.awt.peer.MenuItemPeer

Interface java.awt.peer.MenuItemPeer

public interface **MenuItemPeer**
extends Object
extends MenuComponentPeer

disable()

enable()

setLabel(String)

setLabel

```
public abstract void setLabel(String label)
```

enable

```
public abstract void enable()
```

disable

```
public abstract void disable()
```

Interface java.awt.peer.MenuPeer

Interface java.awt.peer.MenuPeer

public interface **MenuPeer**
extends Object
extends MenuItemPeer

addItem(MenuItem)

addSeparator()

delItem(int)

addSeparator

```
public abstract void addSeparator()
```

 **addItem**

```
public abstract void addItem(MenuItem item)
```

 **delItem**

```
public abstract void delItem(int index)
```

Interface java.awt.peer.PanelPeer

Interface java.awt.peer.PanelPeer

public interface **PanelPeer**
extends Object
extends ContainerPeer

Interface java.awt.peer.ScrollbarPeer

Interface java.awt.peer.ScrollbarPeer

public interface **ScrollbarPeer**
extends Object
extends ComponentPeer




setLineIncrement(int)

setPageIncrement(int)

setValue(int)

setValues(int, int, int, int)

 **setValue**

```
public abstract void setValue(int value)
```

 **setValues**

```
public abstract void setValues(int value,
                               int visible,
                               int minimum,
                               int maximum)
```

 **setLineIncrement**

```
public abstract void setLineIncrement(int l)
```



```
public abstract void setPageIncrement(int l)
```

Interface java.awt.peer.TextAreaPeer

Interface java.awt.peer.TextAreaPeer

public interface **TextAreaPeer**
extends Object
extends TextComponentPeer

insertText(String, int)

minimumSize(int, int)

preferredSize(int, int)

replaceText(String, int, int)

insertText

```
public abstract void insertText(String txt,
                                int pos)
```

replaceText

```
public abstract void replaceText(String txt,
                                 int start,
                                 int end)
```

preferredSize

```
public abstract Dimension preferredSize(int rows,
                                        int cols)
```



```
public abstract Dimension minimumSize(int rows,
                                      int cols)
```

Interface java.awt.peer.TextComponentPeer

Interface java.awt.peer.TextComponentPeer

public interface **TextComponentPeer**
extends Object
extends ComponentPeer

getSelectionEnd()

getSelectionStart()

getText()

select(int, int)

setEditable(boolean)

setText(String)

setEditable

```
public abstract void setEditable(boolean editable)
```

getText

```
public abstract String getText()
```


```
public abstract void setText(String l)
```


```
public abstract int getSelectionStart()
```


```
public abstract int getSelectionEnd()
```


```
public abstract void select(int selStart,
                            int selEnd)
```

Interface java.awt.peer.TextFieldPeer

Interface java.awt.peer.TextFieldPeer

public interface **TextFieldPeer**
extends Object
extends TextComponentPeer

setEchoCharacter

```
public abstract void setEchoCharacter(char c)
```

preferredSize

```
public abstract Dimension preferredSize(int cols)
```

minimumSize

```
public abstract Dimension minimumSize(int cols)
```

Interface java.awt.peer.WindowPeer

Interface java.awt.peer.WindowPeer

public interface **WindowPeer**
extends Object
extends ContainerPeer

toBack()

toFront()

toFront

```
public abstract void toFront()
```

toBack

```
public abstract void toBack()
```

Class java.awt.Point

Class java.awt.Point

```
java.lang.Object
    |
    +----java.awt.Point
```

---

public class **Point**
extends Object

An x,y coordinate.

x

The x coordinate.

y

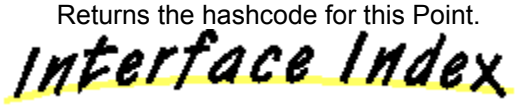
The y coordinate.

Point(int, int)
Constructs and initializes a Point from the specified x and y coordinates.

equals(Object)
Checks whether two pointers are equal.

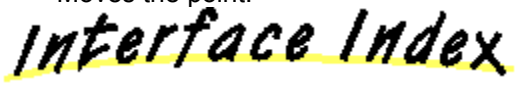
hashCode()

Returns the hashcode for this Point.



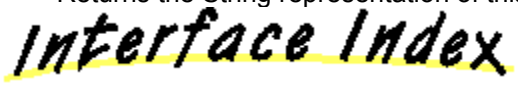
move(int, int)

Moves the point.



toString()

Returns the String representation of this Point's coordinates.



translate(int, int)

Translates the point.

x

```
public int x
```

The x coordinate.

y

```
public int y
```

The y coordinate.

Point

```
public Point(int x,
             int y)
```

Constructs and initializes a Point from the specified x and y coordinates.

Parameters:

x - the x coordinate

y - the y coordinate


move

```
public void move(int x,
                 int y)
```

Moves the point.


translate

```
public void translate(int x,
                      int y)
```

Translates the point.


hashCode

```
public int hashCode()
```

Returns the hashcode for this Point.
Overrides:
hashCode in class Object


equals

```
public boolean equals(Object obj)
```

Checks whether two pointers are equal.
Overrides:
equals in class Object


toString

```
public String toString()
```

Returns the String representation of this Point's coordinates.

Overrides:

toString in class Object

Class java.awt.Polygon

Class java.awt.Polygon

```
java.lang.Object
    |
    +----java.awt.Polygon
```

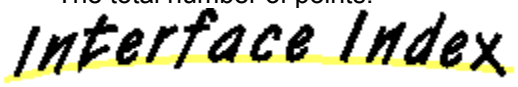
---

public class **Polygon**
extends Object

A polygon consists of a list of x and y coordinates.

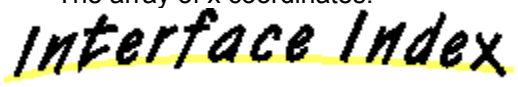
npoints

The total number of points.



xpoints

The array of x coordinates.

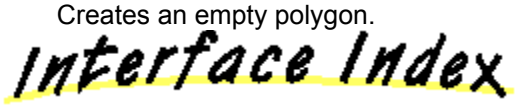


ypoints

The array of y coordinates.

Polygon()

Creates an empty polygon.

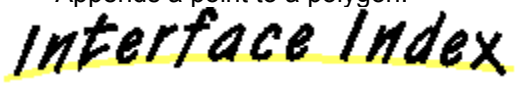


Polygon(int[], int[], int)

Constructs and initializes a Polygon from the specified parameters.

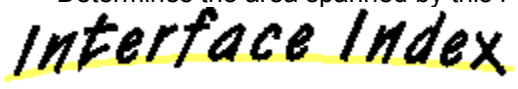
addPoint(int, int)

Appends a point to a polygon.



getBoundingBox()

Determines the area spanned by this Polygon.



inside(int, int)

Determines whether the point (x,y) is inside the Polygon.

npoints

```
public int npoints
```

The total number of points.

xpoints

```
public int xpoints[]
```

The array of x coordinates.

ypoints

```
public int ypoints[]
```

The array of y coordinates.

Polygon

```
public Polygon()
```

Creates an empty polygon.

Polygon

```
public Polygon(int xpoints[],
               int ypoints[],
               int npoints)
```

Constructs and initializes a Polygon from the specified parameters.

Parameters:

xpoints - the array of x coordinates
ypoints - the array of y coordinates
npoints - the total number of points in the Polygon

addPoint

```
public void addPoint(int x,
                     int y)
```

Appends a point to a polygon. If inside(x, y) or another operation that calculates the bounding box has already been performed, this method updates the bounds accordingly.

Parameters:

x - the x coordinate of the point
y - the y coordinate of the point

getBoundingBox

```
public Rectangle getBoundingBox()
```

Determines the area spanned by this Polygon.

Returns:

a Rectangle defining the bounds of the Polygon.

inside

```
public boolean inside(int x,
                      int y)
```

Determines whether the point (x,y) is inside the Polygon. Uses an even-odd insideness rule (also known as an alternating rule).

Parameters:

x - the X coordinate of the point to be tested

y - the Y coordinate of the point to be tested

Based on code by Hanpeter van Vliet .

Class java.awt.Rectangle

Class java.awt.Rectangle

```
java.lang.Object
    |
    +----java.awt.Rectangle
```

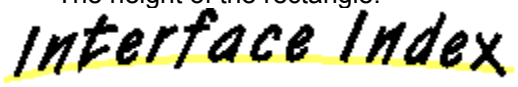
---

public class **Rectangle**
extends Object

A rectangle defined by x, y, width and height.

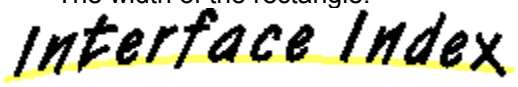
height

The height of the rectangle.



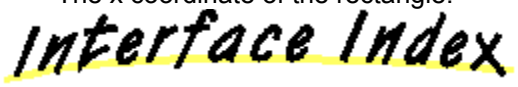
width

The width of the rectangle.



x

The x coordinate of the rectangle.



y

The y coordinate of the rectangle.

Rectangle()

Constructs a new rectangle.

Rectangle(int, int, int, int)

Constructs and initializes a rectangle with the specified parameters.

Rectangle(int, int)

Constructs a rectangle and initializes it with the specified width and height parameters.

Rectangle(Point, Dimension)

Constructs a rectangle and initializes it to a specified point and dimension.

Rectangle(Point)

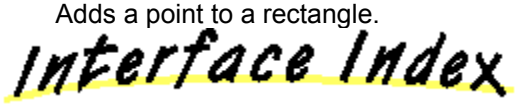
Constructs a rectangle and initializes it to the specified point.

Rectangle(Dimension)

Constructs a rectangle and initializes it to the specified width and height.

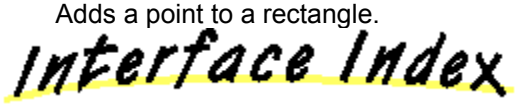
add(int, int)

Adds a point to a rectangle.



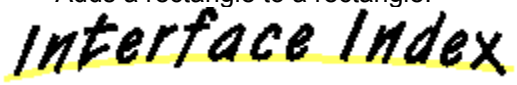
add(Point)

Adds a point to a rectangle.



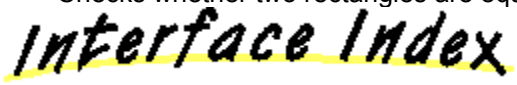
add(Rectangle)

Adds a rectangle to a rectangle.



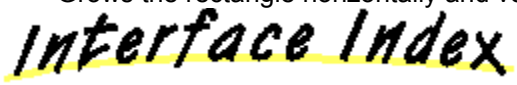
equals(Object)

Checks whether two rectangles are equal.



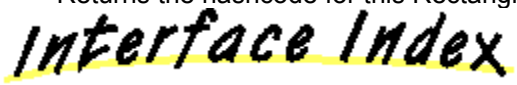
grow(int, int)

Grows the rectangle horizontally and vertically.



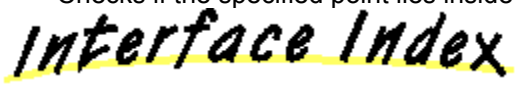
hashCode()

Returns the hashcode for this Rectangle.



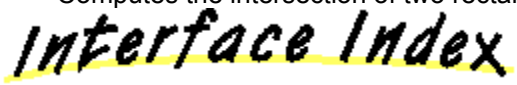
inside(int, int)

Checks if the specified point lies inside a rectangle.



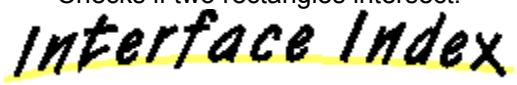
intersection(Rectangle)

Computes the intersection of two rectangles.



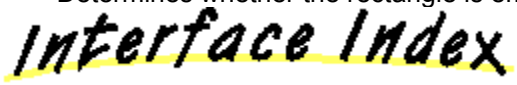
intersects(Rectangle)

Checks if two rectangles intersect.



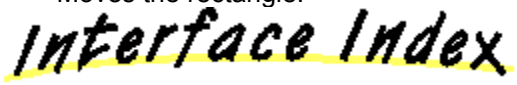
isEmpty()

Determines whether the rectangle is empty.



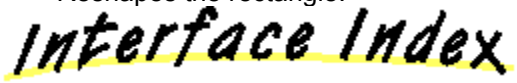
move(int, int)

Moves the rectangle.



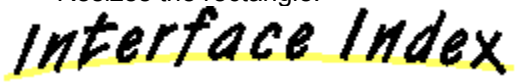
reshape(int, int, int, int)

Reshapes the rectangle.



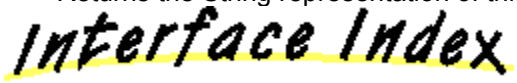
resize(int, int)

Resizes the rectangle.



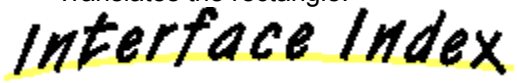
toString()

Returns the String representation of this Rectangle's values.



translate(int, int)

Translates the rectangle.



union(Rectangle)

Computes the union of two rectangles.

x

```
public int x
```

The x coordinate of the rectangle.

y

```
public int y
```

The y coordinate of the rectangle.

width

```
public int width
```

The width of the rectangle.

Interface Index **height**

```
public int height
```

The height of the rectangle.

Interface Index

Interface Index **Rectangle**

```
public Rectangle()
```

Constructs a new rectangle.

Interface Index **Rectangle**

```
public Rectangle(int x,
                 int y,
                 int width,
                 int height)
```

Constructs and initializes a rectangle with the specified parameters.

Parameters:

x - the x coordinate
y - the y coordinate
width - the width of the rectangle
height - the height of the rectangle

Interface Index **Rectangle**

```
public Rectangle(int width,
                 int height)
```

Constructs a rectangle and initializes it with the specified width and height parameters.

Parameters:

width - the width of the rectangle
height - the height of the rectangle

Interface Index **Rectangle**

```
public Rectangle(Point p,
                 Dimension d)
```

Constructs a rectangle and initializes it to a specified point and dimension.

Parameters:

p - the point
d - dimension

Interface Index **Rectangle**

```
public Rectangle(Point p)
```

Constructs a rectangle and initializes it to the specified point.

Parameters:

p - the value of the x and y coordinate

Interface Index **Rectangle**

```
public Rectangle(Dimension d)
```

Constructs a rectangle and initializes it to the specified width and height.

Parameters:

d - the value of the width and height

Interface Index

Interface Index **reshape**

```
public void reshape(int x,
                    int y,
                    int width,
                    int height)
```

Reshapes the rectangle.

move

```
public void move(int x,
                 int y)
```

Moves the rectangle.

translate

```
public void translate(int x,
                      int y)
```

Translates the rectangle.

resize

```
public void resize(int width,
                   int height)
```

Resizes the rectangle.

inside

```
public boolean inside(int x,
                      int y)
```

Checks if the specified point lies inside a rectangle.

Parameters:

x - the x coordinate

y - the y coordinate

intersects

```
public boolean intersects(Rectangle r)
```

Checks if two rectangles intersect.

intersection

```
public Rectangle intersection(Rectangle r)
```

Computes the intersection of two rectangles.

union

```
public Rectangle union(Rectangle r)
```

Computes the union of two rectangles.

add

```
public void add(int newx,
                int newy)
```

Adds a point to a rectangle. This results in the smallest rectangle that contains both the rectangle and the point.

add

```
public void add(Point pt)
```

Adds a point to a rectangle. This results in the smallest rectangle that contains both the rectangle and the point.

add

```
public void add(Rectangle r)
```

Adds a rectangle to a rectangle. This results in the union of the two rectangles.

grow

```
public void grow(int h,
                 int v)
```

Grows the rectangle horizontally and vertically.

isEmpty

```
public boolean isEmpty()
```

Determines whether the rectangle is empty.

hashCode

```
public int hashCode()
```

Returns the hashcode for this Rectangle.
Overrides:
hashCode in class Object

equals

```
public boolean equals(Object obj)
```

Checks whether two rectangles are equal.
Overrides:
equals in class Object

toString

```
public String toString()
```

Returns the String representation of this Rectangle's values.
Overrides:
toString in class Object

Class java.awt.Scrollbar

Class java.awt.Scrollbar

```
java.lang.Object
    |
    +----java.awt.Component
              |
              +----java.awt.Scrollbar
```

---

public class **Scrollbar**
extends Component

A Scrollbar component.

HORIZONTAL

The horizontal Scrollbar variable.

VERTICAL

The vertical Scrollbar variable.

Scrollbar()

Constructs a new vertical Scrollbar.

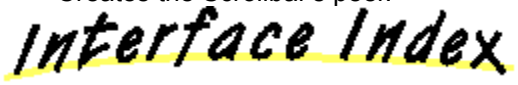
Scrollbar(int)

Constructs a new Scrollbar with the specified orientation.

Scrollbar(int, int, int, int, int)

Constructs a new Scrollbar with the specified orientation, value, page size, and minumum and maximum values.

addNotify()
Creates the Scrollbar's peer.



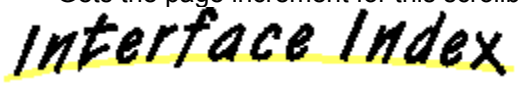
getLineIncrement()
Gets the line increment for this scrollbar.

getMaximum()
Returns the maximum value of this Scrollbar.

getMinimum()
Returns the minimum value of this Scrollbar.

getOrientation()
Returns the orientation for this Scrollbar.

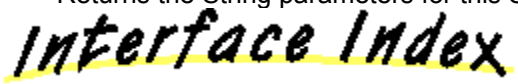
getPageIncrement()
Gets the page increment for this scrollbar.



getValue()
Returns the current value of this Scrollbar.

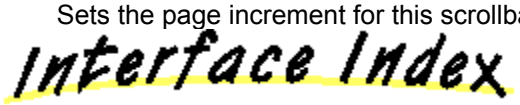
getVisible()
Returns the visible amount of the Scrollbar.

paramString()
Returns the String parameters for this Scrollbar.

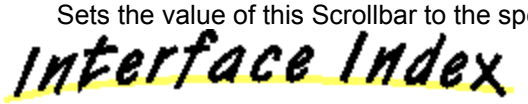


setLineIncrement(int)
Sets the line increment for this scrollbar.

setPageIncrement(int)
Sets the page increment for this scrollbar.



setValue(int)
Sets the value of this Scrollbar to the specified value.



setValues(int, int, int, int)
Sets the values for this Scrollbar.

Interface Index

Interface Index **HORIZONTAL**

```
public final static int HORIZONTAL
```

The horizontal Scrollbar variable.

Interface Index **VERTICAL**

```
public final static int VERTICAL
```

The vertical Scrollbar variable.

Interface Index

Interface Index **Scrollbar**

```
public Scrollbar()
```

Constructs a new vertical Scrollbar.

Interface Index **Scrollbar**

```
public Scrollbar(int orientation)
```

Constructs a new Scrollbar with the specified orientation.

Parameters:

orientation - either Scrollbar.HORIZONTAL or Scrollbar.VERTICAL

Throws: IllegalArgumentException

When an illegal scrollbar orientation is given.

Interface Index **Scrollbar**

```
public Scrollbar(int orientation,
                 int value,
                 int visible,
```

```
                int minimum,
                int maximum)
```

Constructs a new Scrollbar with the specified orientation, value, page size, and minumum and maximum values.

Parameters:

orientation - either Scrollbar.HORIZONTAL or Scrollbar.VERTICAL
value - the scrollbar's value
visible - the size of the visible portion of the scrollable area. The scrollbar will use this value when paging up or down by a page.
minimum - the minimum value of the scrollbar
maximum - the maximum value of the scrollbar

Interface Index

Interface Index **addNotify**

```
public synchronized void addNotify()
```

Creates the Scrollbar's peer. The peer allows you to modify the appearance of the Scrollbar without changing any of its functionality.

Overrides:

addNotify in class Component

Interface Index **getOrientation**

```
public int getOrientation()
```

Returns the orientation for this Scrollbar.

Interface Index **getValue**

```
public int getValue()
```

Returns the current value of this Scrollbar.

See Also:

getMinimum, getMaximum

 setValue

```
public void setValue(int value)
```

Sets the value of this Scrollbar to the specified value.

Parameters:

value - the new value of the Scrollbar. If this value is below the current minimum or above the current maximum, it becomes the new one of those values, respectively.

See Also:

getValue

 getMinimum

```
public int getMinimum()
```

Returns the minimum value of this Scrollbar.

See Also:

getMaximum, getValue

 getMaximum

```
public int getMaximum()
```

Returns the maximum value of this Scrollbar.

See Also:

getMinimum, getValue

 getVisible

```
public int getVisible()
```

Returns the visible amount of the Scrollbar.

 setLineIncrement

```
public void setLineIncrement(int l)
```

Sets the line increment for this scrollbar. This is the value that will be added (subtracted) when the user hits the line down (up) gadgets.

getLineIncrement

```
public int getLineIncrement()
```

Gets the line increment for this scrollbar.

setPageIncrement

```
public void setPageIncrement(int l)
```

Sets the page increment for this scrollbar. This is the value that will be added (subtracted) when the user hits the page down (up) gadgets.

getPageIncrement

```
public int getPageIncrement()
```

Gets the page increment for this scrollbar.

setValues

```
public void setValues(int value,
                      int visible,
                      int minimum,
                      int maximum)
```

Sets the values for this Scrollbar.
Parameters:
value - is the position in the current window.
visible - is the amount visible per page
minimum - is the minimum value of the scrollbar
maximum - is the maximum value of the scrollbar

paramString

```
protected String paramString()
```

Returns the String parameters for this Scrollbar.

Overrides:
paramString in class Component

Class java.awt.TextArea

Class java.awt.TextArea

```
java.lang.Object
    |
    +----java.awt.Component
            |
            +----java.awt.TextComponent
                    |
                    +----java.awt.TextArea
```

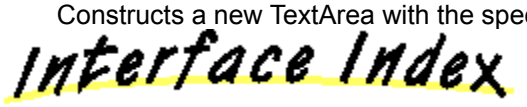
---

public class **TextArea**
extends TextComponent

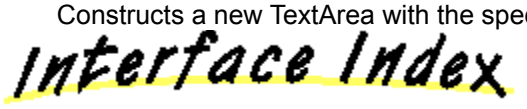
A TextArea object is a multi-line area that displays text. It can be set to allow editing or read-only modes.

TextArea()
Constructs a new TextArea.

TextArea(int, int)
Constructs a new TextArea with the specified number of rows and columns.



TextArea(String)
Constructs a new TextArea with the specified text displayed.

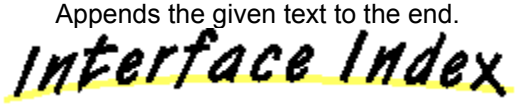


TextArea(String, int, int)
Constructs a new TextArea with the specified text and number of rows and columns.

addNotify()
Creates the TextArea's peer.

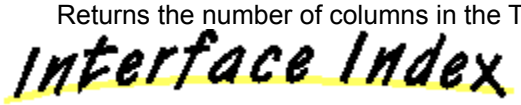
appendText(String)

Appends the given text to the end.



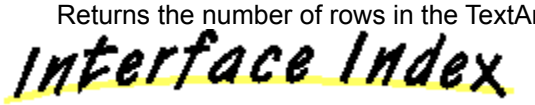
getColumns()

Returns the number of columns in the TextArea.



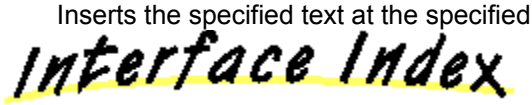
getRows()

Returns the number of rows in the TextArea.



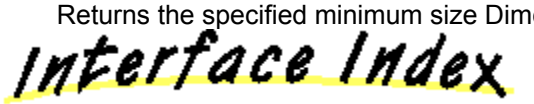
insertText(String, int)

Inserts the specified text at the specified position.



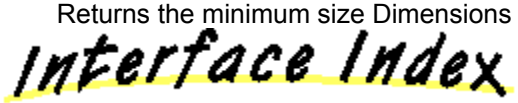
minimumSize(int, int)

Returns the specified minimum size Dimensions of the TextArea.



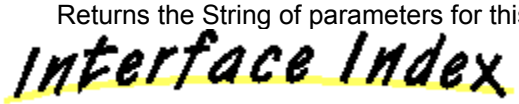
minimumSize()

Returns the minimum size Dimensions of the TextArea.



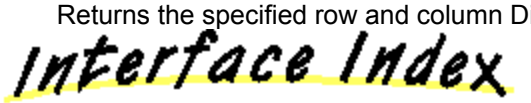
paramString()

Returns the String of parameters for this TextArea.



preferredSize(int, int)

Returns the specified row and column Dimensions of the TextArea.



preferredSize()

Returns the preferred size Dimensions of the TextArea.

replaceText(String, int, int)

Replaces text from the indicated start to end position with the new text specified.

TextArea

```
public TextArea()
```

Constructs a new TextArea.

TextArea

```
public TextArea(int rows,
```

```
                int cols)
```

Constructs a new TextArea with the specified number of rows and columns.
Parameters:
rows - the number of rows
cols - the number of columns

TextArea

```
public TextArea(String text)
```

Constructs a new TextArea with the specified text displayed.
Parameters:
text - the text to be displayed

TextArea

```
public TextArea(String text,
                int rows,
                int cols)
```

Constructs a new TextArea with the specified text and number of rows and columns.
Parameters:
text - the text to be displayed
rows - the number of rows
cols - the number of cols

addNotify

```
public synchronized void addNotify()
```

Creates the TextArea's peer. The peer allows us to modify the appearance of the TextArea without changing any of its functionality.
Overrides:
addNotify in class Component

Interface Index **insertText**

```
public void insertText(String str,
                       int pos)
```

Inserts the specified text at the specified position.
Parameters:
str - the text to insert.
pos - the position at which to insert.
See Also:
setText, replaceText

Interface Index **appendText**

```
public void appendText(String str)
```

Appends the given text to the end.
Parameters:
str - the text to insert
See Also:
insertText

Interface Index **replaceText**

```
public void replaceText(String str,
                        int start,
                        int end)
```

Replaces text from the indicated start to end position with the new text specified.
Parameters:
str - the text to use as the replacement.
start - the start position.
end - the end position.
See Also:
insertText, replaceText

Interface Index **getRows**

```
public int getRows()
```

Returns the number of rows in the TextArea.

getColumns

```
public int getColumns()
```

Returns the number of columns in the TextArea.

preferredSize

```
public Dimension preferredSize(int rows,
                               int cols)
```

Returns the specified row and column Dimensions of the TextArea.

Parameters:

rows - the preferred rows amount
cols - the preferred columns amount

preferredSize

```
public Dimension preferredSize()
```

Returns the preferred size Dimensions of the TextArea.

Overrides:

preferredSize in class Component

minimumSize

```
public Dimension minimumSize(int rows,
                             int cols)
```

Returns the specified minimum size Dimensions of the TextArea.

Parameters:

rows - the minimum row size
cols - the minimum column size

 minimumSize

```
public Dimension minimumSize()
```

Returns the minimum size Dimensions of the TextArea.
Overrides:
minimumSize in class Component

 paramString

```
protected String paramString()
```

Returns the String of parameters for this TextArea.
Overrides:
paramString in class TextComponent

Class java.awt.TextComponent

Class java.awt.TextComponent

```
java.lang.Object
    |
    +----java.awt.Component
            |
            +----java.awt.TextComponent
```

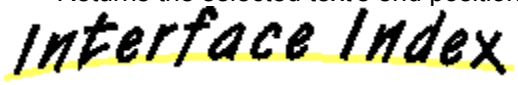
---

public class **TextComponent**
extends Component

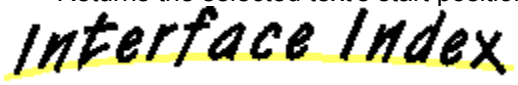
A TextComponent is a component that allows the editing of some text.

getSelectedText()
Returns the selected text contained in this TextComponent.

getSelectionEnd()
Returns the selected text's end position.



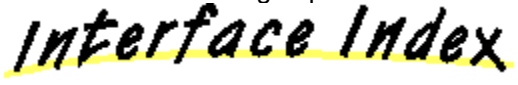
getSelectionStart()
Returns the selected text's start position.



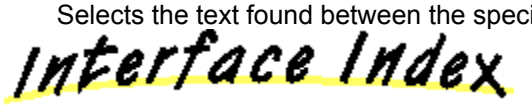
getText()
Returns the text contained in this TextComponent.

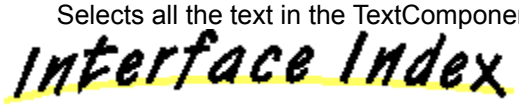
isEditable()
Returns the boolean indicating whether this TextComponent is editable or not.

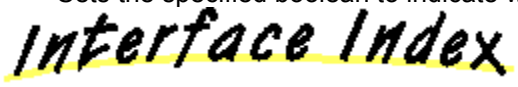
paramString()
Returns the String of parameters for this TextComponent.



removeNotify()
Removes the TextComponent's peer.

select(int, int)
Selects the text found between the specified start and end locations.


selectAll()
Selects all the text in the TextComponent.


setEditable(boolean)
Sets the specified boolean to indicate whether or not this TextComponent should be editable.


setText(String)
Sets the text of this TextComponent to the specified text.

removeNotify

```
public synchronized void removeNotify()
```

Removes the TextComponent's peer. The peer allows us to modify the appearance of the TextComponent without changing its functionality.
Overrides:
removeNotify in class Component

setText

```
public void setText(String t)
```

Sets the text of this TextComponent to the specified text.
Parameters:
t - the new text to be set
See Also:
getText

getText

```
public String getText()
```

Returns the text contained in this TextComponent.

See Also:
setText

getSelectedText

```
public String getSelectedText()
```

Returns the selected text contained in this TextComponent.
See Also:
setText

isEditable

```
public boolean isEditable()
```

Returns the boolean indicating whether this TextComponent is editable or not.
See Also:
setEditable

setEditable

```
public void setEditable(boolean t)
```

Sets the specified boolean to indicate whether or not this TextComponent should be editable.
Parameters:
t - the boolean to be set
See Also:
isEditable

getSelectionStart

```
public int getSelectionStart()
```

Returns the selected text's start position.

getSelectionEnd

```
public int getSelectionEnd()
```

Returns the selected text's end position.

select

```
public void select(int selStart,
                   int selEnd)
```

Selects the text found between the specified start and end locations.

Parameters:

selStart - the start position of the text
selEnd - the end position of the text

selectAll

```
public void selectAll()
```

Selects all the text in the TextComponent.

paramString

```
protected String paramString()
```

Returns the String of parameters for this TextComponent.

Overrides:

paramString in class Component

Class java.awt.TextField

Class java.awt.TextField

```
java.lang.Object
    |
    +----java.awt.Component
            |
            +----java.awt.TextComponent
                    |
                    +----java.awt.TextField
```

---

public class **TextField**
extends TextComponent

TextField is a component that allows the editing of a single line of text.

TextField()
Constructs a new TextField.

TextField(int)
Constructs a new TextField initialized with the specified columns.

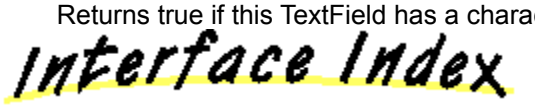
TextField(String)
Constructs a new TextField initialized with the specified text.

TextField(String, int)
Constructs a new TextField initialized with the specified text and columns.

addNotify()
Creates the TextField's peer.

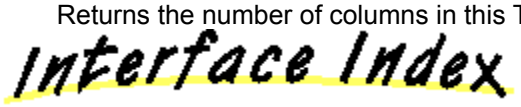
echoCharIsSet()

Returns true if this TextField has a character set for echoing.



getColumns()

Returns the number of columns in this TextField.



getEchoChar()

Returns the character to be used for echoing.

minimumSize(int)

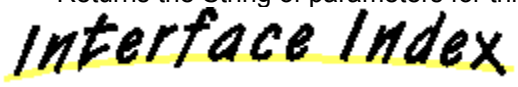
Returns the minimum size Dimensions needed for this TextField with the specified amount of columns.

minimumSize()

Returns the minimum size Dimensions needed for this TextField.

paramString()

Returns the String of parameters for this TExtField.

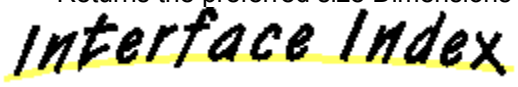


preferredSize(int)

Returns the preferred size Dimensions needed for this TextField with the specified amount of columns.

preferredSize()

Returns the preferred size Dimensions needed for this TextField.



setEchoCharacter(char)

Sets the echo character for this TextField.

TextField

```
public TextField()
```

Constructs a new TextField.

TextField

```
public TextField(int cols)
```

Constructs a new TextField initialized with the specified columns.
Parameters:
cols - the number of columns

TextField

```
public TextField(String text)
```

Constructs a new TextField initialized with the specified text.
Parameters:
text - the text to be displayed

TextField

```
public TextField(String text,
                 int cols)
```

Constructs a new TextField initialized with the specified text and columns.
Parameters:
text - the text to be displayed
cols - the number of columns

addNotify

```
public synchronized void addNotify()
```

Creates the TextField's peer. The peer allows us to modify the appearance of the TextField without changing its functionality.
Overrides:
addNotify in class Component

getEchoChar

```
public char getEchoChar()
```

Returns the character to be used for echoing.

See Also:

setEchoCharacter, echoCharIsSet

Interface Index **echoCharIsSet**

```
public boolean echoCharIsSet()
```

Returns true if this TextField has a character set for echoing.

See Also:

setEchoCharacter, getEchoChar

Interface Index **getColumns**

```
public int getColumns()
```

Returns the number of columns in this TextField.

Interface Index **setEchoCharacter**

```
public void setEchoCharacter(char c)
```

Sets the echo character for this TextField. This is useful for fields where the user input shouldn't be echoed to the screen, as in the case of a TextField that represents a password.

Parameters:

c - the echo character for this TextField

See Also:

echoCharIsSet, getEchoChar

Interface Index **preferredSize**

```
public Dimension preferredSize(int cols)
```

Returns the preferred size Dimensions needed for this TextField with the specified amount of columns.

Parameters:

cols - the number of columns in this TextField

preferredSize

```
public Dimension preferredSize()
```

Returns the preferred size Dimensions needed for this TextField.

Overrides:

preferredSize in class Component

minimumSize

```
public Dimension minimumSize(int cols)
```

Returns the minimum size Dimensions needed for this TextField with the specified amount of columns.

Parameters:

cols - the number of columns in this TextField

minimumSize

```
public Dimension minimumSize()
```

Returns the minimum size Dimensions needed for this TextField.

Overrides:

minimumSize in class Component

paramString

```
protected String paramString()
```

Returns the String of parameters for this TExtField.

Overrides:

paramString in class TextComponent

Class java.awt.Toolkit

Class java.awt.Toolkit

```
java.lang.Object
    |
    +----java.awt.Toolkit
```

---

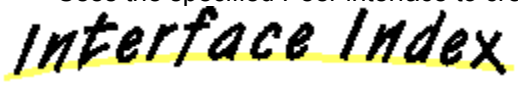
public class **Toolkit**
extends Object

An AWT toolkit. It is used to bind the abstract AWT classes to a particular native toolkit implementation.

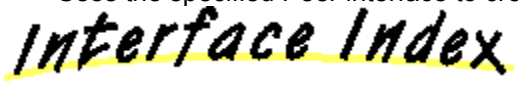
Toolkit()

checkImage(Image, int, int, ImageObserver)
Returns the status of the construction of the indicated method at the indicated width and height for the default screen.

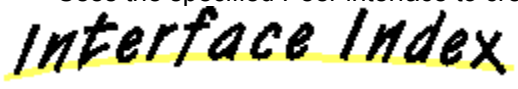
createButton(Button)
Uses the specified Peer interface to create a new Button.



createCanvas(Canvas)
Uses the specified Peer interface to create a new Canvas.

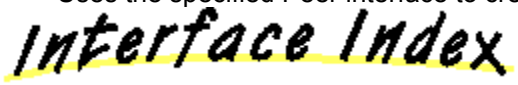


createCheckbox(Checkbox)
Uses the specified Peer interface to create a new Checkbox.



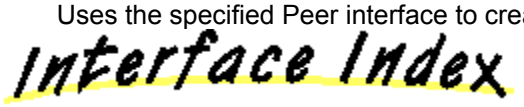
createCheckboxMenuItem(CheckboxMenuItem)

Uses the specified Peer interface to create a new CheckboxMenuItem.



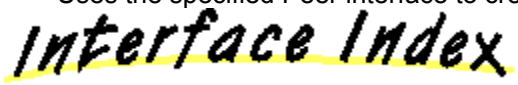
createChoice(Choice)

Uses the specified Peer interface to create a new Choice.



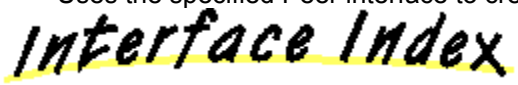
createDialog(Dialog)

Uses the specified Peer interface to create a new Dialog.



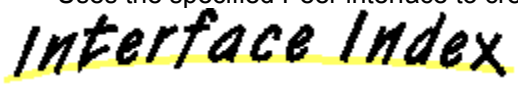
createFileDialog(FileDialog)

Uses the specified Peer interface to create a new FileDialog.



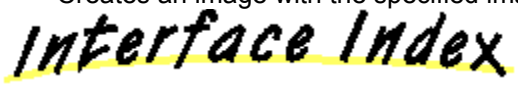
createFrame(Frame)

Uses the specified Peer interface to create a new Frame.



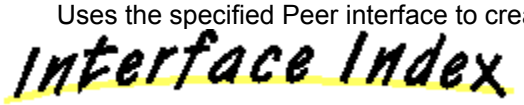
createImage(ImageProducer)

Creates an image with the specified image producer.



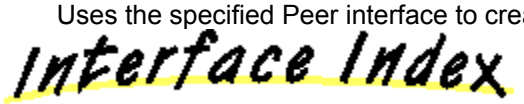
createLabel(Label)

Uses the specified Peer interface to create a new Label.



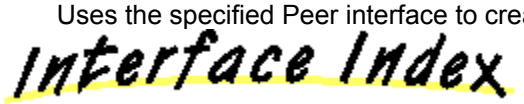
createList(List)

Uses the specified Peer interface to create a new List.



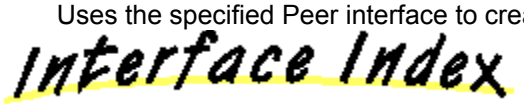
createMenu(Menu)

Uses the specified Peer interface to create a new Menu.



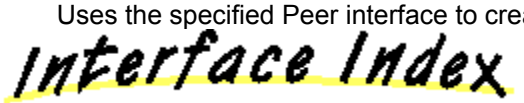
createMenuBar(MenuBar)

Uses the specified Peer interface to create a new MenuBar.



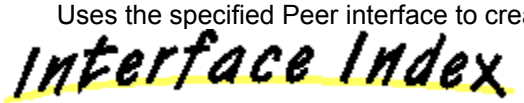
createMenuItem(MenuItem)

Uses the specified Peer interface to create a new MenuItem.



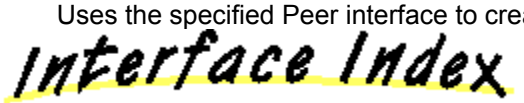
createPanel(Panel)

Uses the specified Peer interface to create a new Panel.



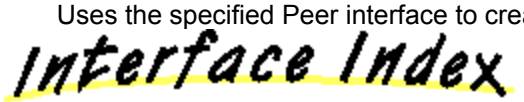
createScrollbar(Scrollbar)

Uses the specified Peer interface to create a new Scrollbar.



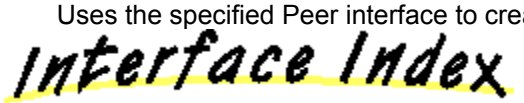
createTextArea(TextArea)

Uses the specified Peer interface to create a new TextArea.



createTextField(TextField)

Uses the specified Peer interface to create a new TextField.



createWindow(Window)

Uses the specified Peer interface to create a new Window.

getColorModel()

Returns the ColorModel of the screen.

getDefaultToolkit()

Returns the default toolkit.

getFontList()

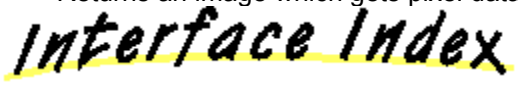
Returns the names of the available fonts.

getFontMetrics(Font)

Returns the screen metrics of the font.

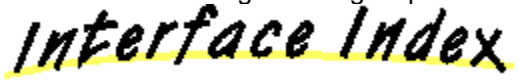
getImage(String)

Returns an image which gets pixel data from the specified file.



getImage(URL)

Returns an image which gets pixel data from the specified URL.



getScreenResolution()

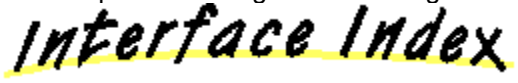
Returns the screen resolution in dots-per-inch.

getScreenSize()

Gets the size of the screen.

prepareImage(Image, int, int, ImageObserver)

Prepares an image for rendering on the default screen at the specified width and height.



sync()

Syncs the graphics state; useful when doing animation.

Toolkit

```
public Toolkit()
```

createButton

```
protected abstract ButtonPeer createButton(Button target)
```

Uses the specified Peer interface to create a new Button.
Parameters:
target - the Button to be created

createTextField

```
protected abstract TextFieldPeer createTextField(TextField target)
```

Uses the specified Peer interface to create a new TextField.
Parameters:
target - the TextField to be created

createLabel

```
protected abstract LabelPeer createLabel(Label target)
```

Uses the specified Peer interface to create a new Label.
Parameters:
target - the Label to be created

createList

```
protected abstract ListPeer createList(List target)
```

Uses the specified Peer interface to create a new List.
Parameters:
target - the List to be created

createCheckbox

```
protected abstract CheckboxPeer createCheckbox(Checkbox target)
```

Uses the specified Peer interface to create a new Checkbox.
Parameters:

target - the Checkbox to be created

createScrollbar

```
protected abstract ScrollbarPeer createScrollbar(Scrollbar target)
```

Uses the specified Peer interface to create a new Scrollbar.
Parameters:
target - the Scrollbar to be created

createTextArea

```
protected abstract TextAreaPeer createTextArea(TextArea target)
```

Uses the specified Peer interface to create a new TextArea.
Parameters:
target - the TextArea to be created

createChoice

```
protected abstract ChoicePeer createChoice(Choice target)
```

Uses the specified Peer interface to create a new Choice.
Parameters:
target - the Choice to be created

createFrame

```
protected abstract FramePeer createFrame(Frame target)
```

Uses the specified Peer interface to create a new Frame.
Parameters:
target - the Frame to be created

Interface Index **createCanvas**

```
protected abstract CanvasPeer createCanvas(Canvas target)
```

Uses the specified Peer interface to create a new Canvas.
Parameters:
target - the Canvas to be created

Interface Index **createPanel**

```
protected abstract PanelPeer createPanel(Panel target)
```

Uses the specified Peer interface to create a new Panel.
Parameters:
target - the Panel to be created

Interface Index **createWindow**

```
protected abstract WindowPeer createWindow(Window target)
```

Uses the specified Peer interface to create a new Window.
Parameters:
target - the Window to be created

Interface Index **createDialog**

```
protected abstract DialogPeer createDialog(Dialog target)
```

Uses the specified Peer interface to create a new Dialog.
Parameters:
target - the Dialog to be created

Interface Index **createMenuBar**

```
protected abstract MenuBarPeer createMenuBar(MenuBar target)
```

Uses the specified Peer interface to create a new MenuBar.
Parameters:
target - the MenuBar to be created

createMenu

```
protected abstract MenuPeer createMenu(Menu target)
```

Uses the specified Peer interface to create a new Menu.
Parameters:
target - the Menu to be created

createMenuItem

```
protected abstract MenuItemPeer createMenuItem(MenuItem target)
```

Uses the specified Peer interface to create a new MenuItem.
Parameters:
target - the MenuItem to be created

createFileDialog

```
protected abstract FileDialogPeer createFileDialog(FileDialog target)
```

Uses the specified Peer interface to create a new FileDialog.
Parameters:
target - the FileDialog to be created

createCheckboxMenuItem

```
protected abstract CheckboxMenuItemPeer
createCheckboxMenuItem(CheckboxMenuItem  target)
```

Uses the specified Peer interface to create a new CheckboxMenuItem.
Parameters:
target - the CheckboxMenuItem to be created

getScreenSize

```
public abstract Dimension getScreenSize()
```

Gets the size of the screen.

getScreenResolution

```
public abstract int getScreenResolution()
```

Returns the screen resolution in dots-per-inch.

getColorModel

```
public abstract ColorModel getColorModel()
```

Returns the ColorModel of the screen.

getFontList

```
public abstract String[] getFontList()
```

Returns the names of the available fonts.

getFontMetrics

```
public abstract FontMetrics getFontMetrics(Font font)
```

Returns the screen metrics of the font.

sync

```
public abstract void sync()
```

Syncs the graphics state; useful when doing animation.

 getDefaultToolkit

```
public static synchronized Toolkit getDefaultToolkit()
```

Returns the default toolkit. This is controlled by the "awt.toolkit" property.
Throws: AWTError
Toolkit not found or could not be instantiated.

 getImage

```
public abstract Image getImage(String filename)
```

Returns an image which gets pixel data from the specified file.
Parameters:
filename - the file containing the pixel data in one of the recognized file formats

 getImage

```
public abstract Image getImage(URL url)
```

Returns an image which gets pixel data from the specified URL.
Parameters:
url - the URL to use in fetching the pixel data

 prepareImage

```
public abstract boolean prepareImage(Image image,
                                     int width,
                                     int height,
                                     ImageObserver observer)
```

Prepares an image for rendering on the default screen at the specified width and height.

 checkImage

```
public abstract int checkImage(Image image,
                               int width,
                               int height,
```

ImageObserver observer)

Returns the status of the construction of the indicated method at the indicated width and height for the default screen.

createImage

```
public abstract Image createImage(ImageProducer producer)
```

Creates an image with the specified image producer.

Parameters:

producer - the image producer to be used

Class java.awt.Window

Class java.awt.Window

```
java.lang.Object
    |
    +----java.awt.Component
            |
            +----java.awt.Container
                    |
                    +----java.awt.Window
```

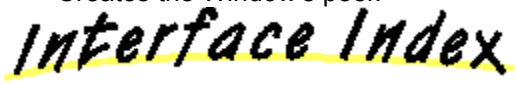
---

public class **Window**
extends Container

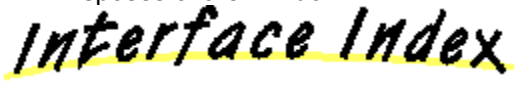
A Window is a top-level window with no borders and no menubar. It could be used to implement a pop-up menu. The default layout for a window is BorderLayout.

Window(Frame)
Constructs a new Window initialized to an invisible state.

addNotify()
Creates the Window's peer.



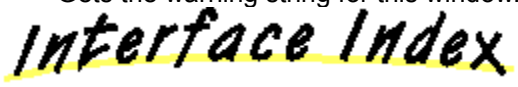
dispose()
Disposes of the Window.



getToolkit()
Returns the toolkit of this frame.

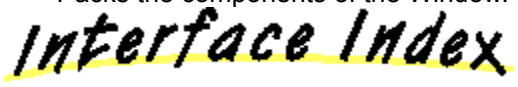
getWarningString()

Gets the warning string for this window.



pack()

Packs the components of the Window.



show()

Shows the Window.

toBack()

Sends the frame to the back of the Window.

toFront()

Brings the frame to the front of the Window.

Window

```
public Window(Frame parent)
```

Constructs a new Window initialized to an invisible state. It behaves as a modal dialog in that it will block input to other windows when shown.

Parameters:

parent - the owner of the dialog

See Also:

resize, show

addNotify

```
public synchronized void addNotify()
```

Creates the Window's peer. The peer allows us to modify the appearance of the Window without changing its functionality.

Overrides:

addNotify in class Container

 pack

```
public synchronized void pack()
```

Packs the components of the Window.

 show

```
public synchronized void show()
```

Shows the Window. This will bring the window to the front if the window is already visible.
Overrides:
show in class Component
See Also:
hide

 dispose

```
public synchronized void dispose()
```

Disposes of the Window. This method must be called to release the resources that are used for the window.

 toFront

```
public void toFront()
```

Brings the frame to the front of the Window.

 toBack

```
public void toBack()
```

Sends the frame to the back of the Window.

 getToolkit

```
public Toolkit getToolkit()
```

Returns the toolkit of this frame.

Overrides:

getToolkit in class Component

See Also:

Toolkit

getWarningString

```
public final String getWarningString()
```

Gets the warning string for this window. This is a string that will be displayed somewhere in the visible area of windows that are not secure.

Class java.io.BufferedInputStream

Class java.io.BufferedInputStream

```
java.lang.Object
    |
    +----java.io.InputStream
            |
            +----java.io.FilterInputStream
                    |
                    +----java.io.BufferedInputStream
```

---

public class **BufferedInputStream**
extends FilterInputStream

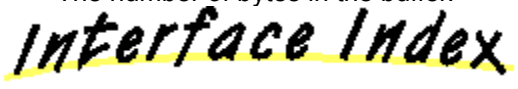
A buffered input stream. This stream lets you read in characters from a stream without causing a read every time. The data is read into a buffer, subsequent reads result in a fast buffer access.

buf

The buffer where data is stored.

count

The number of bytes in the buffer.



marklimit

The maximum readahead allowed after a mark() before subsequent calls to reset() fail.

markpos

The position in the buffer of the current mark.

pos

The current position in the buffer.

BufferedInputStream(InputStream)

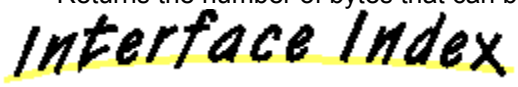
Creates a new buffered stream with a default buffer size.

BufferedInputStream(InputStream, int)

Creates a new buffered stream with the specified buffer size.

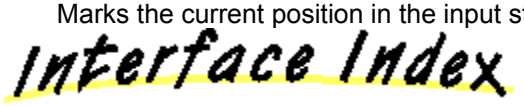
available()

Returns the number of bytes that can be read without blocking.



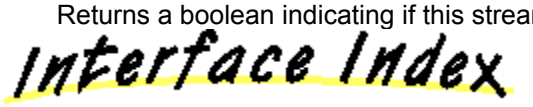
mark(int)

Marks the current position in the input stream.



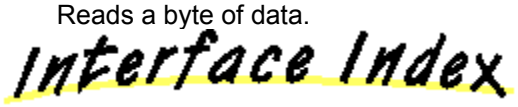
markSupported()

Returns a boolean indicating if this stream type supports mark/reset.



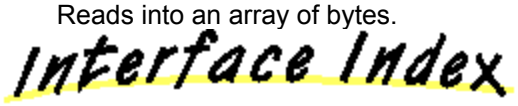
read()

Reads a byte of data.



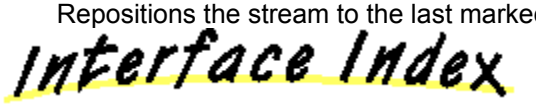
read(byte[], int, int)

Reads into an array of bytes.



reset()

Repositions the stream to the last marked position.



skip(long)

Skips n bytes of input.

buf

```
protected byte buf[]
```

The buffer where data is stored.

count

```
protected int count
```

The number of bytes in the buffer.

pos

```
protected int pos
```

The current position in the buffer.

markpos

```
protected int markpos
```

The position in the buffer of the current mark. This mark is set to -1 if there is no current mark.

marklimit

```
protected int marklimit
```

The maximum readahead allowed after a mark() before subsequent calls to reset() fail.

BufferedInputStream

```
public BufferedInputStream(InputStream in)
```

Creates a new buffered stream with a default buffer size.

Parameters:

in - the input stream

BufferedInputStream

```
public BufferedInputStream(InputStream in,
                           int size)
```

Creates a new buffered stream with the specified buffer size.

Parameters:

in - the input stream
size - the buffer size

Interface Index

Interface Index

read

```
public synchronized int read() throws IOException
```

Reads a byte of data. This method will block if no input is available.

Returns:

the byte read, or -1 if the end of the stream is reached.

Throws: IOException

If an I/O error has occurred.

Overrides:

read in class FilterInputStream

Interface Index

read

```
public synchronized int read(byte b[],
                             int off,
                             int len) throws IOException
```

Reads into an array of bytes. Blocks until some input is available.

Parameters:

b - the buffer into which the data is read
off - the start offset of the data
len - the maximum number of bytes read

Returns:

the actual number of bytes read, -1 is returned when the end of the stream is reached.

Throws: IOException

If an I/O error has occurred.

Overrides:

read in class FilterInputStream

Interface Index **skip**

```
public synchronized long skip(long n) throws IOException
```

Skips n bytes of input.
Parameters:
n - the number of bytes to be skipped
Returns:
the actual number of bytes skipped.
Throws: IOException
If an I/O error has occurred.
Overrides:
skip in class FilterInputStream

Interface Index **available**

```
public synchronized int available() throws IOException
```

Returns the number of bytes that can be read without blocking. This total is the number of bytes in the buffer and the number of bytes available from the input stream.
Returns:
the number of available bytes.
Overrides:
available in class FilterInputStream

Interface Index **mark**

```
public synchronized void mark(int readlimit)
```

Marks the current position in the input stream. A subsequent call to the reset() method will reposition the stream at the last marked position so that subsequent reads will re-read the same bytes. The stream promises to allow readlimit bytes to be read before the mark position gets invalidated.
Parameters:
readlimit - the maximum limit of bytes allowed to be read before the mark position becomes invalid.
Overrides:
mark in class FilterInputStream

Interface Index **reset**

```
public synchronized void reset() throws IOException
```

Repositions the stream to the last marked position. If the stream has not been marked, or if the mark has been invalidated, an IOException is thrown. Stream marks are intended to be used in situations where you need to read ahead a little to see what's in the stream. Often this is most easily done by invoking some general parser. If the stream is of the type handled by the parser, it just chugs along happily. If the stream is not of that type, the parser should toss an exception when it fails. If an exception gets tossed within readlimit bytes, the parser will allow the outer code to reset the stream and to try another parser.

Throws: IOException

If the stream has not been marked or if the mark has been invalidated.

Overrides:

reset in class FilterInputStream

markSupported

```
public boolean markSupported()
```

Returns a boolean indicating if this stream type supports mark/reset.

Overrides:

markSupported in class FilterInputStream

Class java.io.BufferedOutputStream

Class java.io.BufferedOutputStream

```
java.lang.Object
    |
    +----java.io.OutputStream
            |
            +----java.io.FilterOutputStream
                    |
                    +----java.io.BufferedOutputStream
```

---

public class **BufferedOutputStream**
extends FilterOutputStream

A buffered output stream. This stream lets you write characters to a stream without causing a write every time. The data is first written into a buffer. Data is written to the actual stream only when the buffer is full, or when the stream is flushed.

buf

The buffer where data is stored.

count

The number of bytes in the buffer.

BufferedOutputStream(OutputStream)

Creates a new buffered stream with a default buffer size.

BufferedOutputStream(OutputStream, int)

Creates a new buffered stream with the specified buffer size.

flush()

Flushes the stream.

write(int)

Writes a byte.

write(byte[], int, int)

Writes a subarray of bytes.

buf

```
protected byte buf[]
```

The buffer where data is stored.

count

```
protected int count
```

The number of bytes in the buffer.

BufferedOutputStream

```
public BufferedOutputStream(OutputStream out)
```

Creates a new buffered stream with a default buffer size.

Parameters:

out - the output stream

Interface Index **BufferedOutputStream**

```
public BufferedOutputStream(OutputStream out,
                            int size)
```

Creates a new buffered stream with the specified buffer size.

Parameters:

out - the output stream
size - the buffer size

Interface Index

Interface Index **write**

```
public synchronized void write(int b) throws IOException
```

Writes a byte. This method will block until the byte is actually written.

Parameters:

b - the byte to be written

Throws: IOException

If an I/O error has occurred.

Overrides:

write in class FilterOutputStream

Interface Index **write**

```
public synchronized void write(byte b[],
                               int off,
                               int len) throws IOException
```

Writes a subarray of bytes.

Parameters:

b - the data to be written
off - the start offset in the data
len - the number of bytes that are written

Throws: IOException

If an I/O error has occurred.

Overrides:

write in class FilterOutputStream

flush

```
public synchronized void flush() throws IOException
```

Flushes the stream. This will write any buffered output bytes.

Throws: IOException
If an I/O error has occurred.

Overrides:
flush in class FilterOutputStream

Class java.io.ByteArrayInputStream

Class java.io.ByteArrayInputStream

```
java.lang.Object
    |
    +----java.io.InputStream
             |
             +----java.io.ByteArrayInputStream
```

---

public class **ByteArrayInputStream**
extends InputStream

This class implements a buffer that can be used as an InputStream.

buf

The buffer where data is stored.

count

The number of characters to use in the buffer.

pos

The current position in the buffer.

ByteArrayInputStream(byte[])

Creates an ByteArrayInputStream from the specified array of bytes.

ByteArrayInputStream(byte[], int, int)

Creates an ByteArrayInputStream from the specified array of bytes.

Interface Index

Interface Index

available()

Returns the number of available bytes in the buffer.

Interface Index

read()

Reads a byte of data.

Interface Index

read(byte[], int, int)

Reads into an array of bytes.

Interface Index

reset()

Resets the buffer to the beginning.

Interface Index

skip(long)

Skips n bytes of input.

Interface Index

Interface Index

buf

```
protected byte buf[]
```

The buffer where data is stored.

Interface Index

pos

```
protected int pos
```

The current position in the buffer.

Interface Index

count

```
protected int count
```

The number of characters to use in the buffer.

Interface Index

Interface Index **ByteArrayInputStream**

```
public ByteArrayInputStream(byte buf[])
```

Creates an ByteArrayInputStream from the specified array of bytes.
Parameters:
buf - the input buffer (not copied)

Interface Index **ByteArrayInputStream**

```
public ByteArrayInputStream(byte buf[],
                            int offset,
                            int length)
```

Creates an ByteArrayInputStream from the specified array of bytes.
Parameters:
buf - the input buffer (not copied)
offset - the offset of the first byte to read
length - the number of bytes to read

Interface Index

Interface Index **read**

```
public synchronized int read()
```

Reads a byte of data.
Returns:
the byte read, or -1 if the end of the stream is reached.
Overrides:
read in class InputStream

Interface Index **read**

```
public synchronized int read(byte b[],
                             int off,
                             int len)
```

Reads into an array of bytes.
Parameters:
b - the buffer into which the data is read
off - the start offset of the data
len - the maximum number of bytes read
Returns:
the actual number of bytes read; -1 is returned when the end of the stream is reached.
Overrides:
read in class InputStream

Interface Index skip

```
public synchronized long skip(long n)
```

Skips n bytes of input.
Parameters:
n - the number of bytes to be skipped
Returns:
the actual number of bytes skipped.
Overrides:
skip in class InputStream

Interface Index available

```
public synchronized int available()
```

Returns the number of available bytes in the buffer.
Overrides:
available in class InputStream

Interface Index reset

```
public synchronized void reset()
```

Resets the buffer to the beginning.
Overrides:
reset in class InputStream

Class java.ByteArrayOutputStream

Class java.ByteArrayOutputStream

```
java.lang.Object
    |
    +----java.io.OutputStream
             |
             +----java.io.ByteArrayOutputStream
```

---

public class **ByteArrayOutputStream**
extends OutputStream

This class implements a buffer that can be used as an OutputStream. The buffer automatically grows when data is written to the stream. The data can be retrieved using toByteArray() and toString().

buf

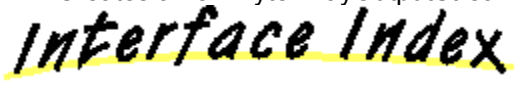
The buffer where data is stored.

count

The number of bytes in the buffer.

ByteArrayOutputStream()

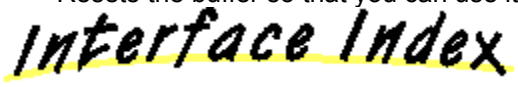
Creates a new ByteArrayOutputStream.



ByteArrayOutputStream(int)

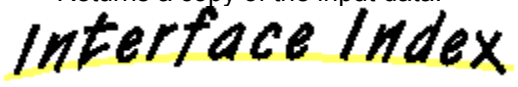
Creates a new ByteArrayOutputStream with the specified initial size.

reset()
Resets the buffer so that you can use it again without throwing away the already allocated buffer.

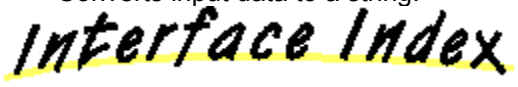


size()
Returns the current size of the buffer.

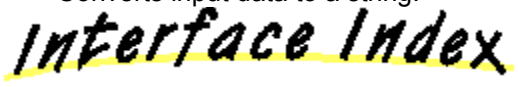
toByteArray()
Returns a copy of the input data.



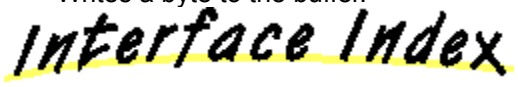
toString()
Converts input data to a string.



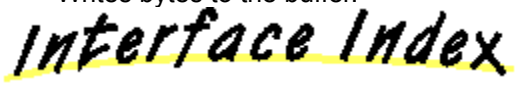
toString(int)
Converts input data to a string.



write(int)
Writes a byte to the buffer.



write(byte[], int, int)
Writes bytes to the buffer.



writeTo(OutputStream)
Writes the contents of the buffer to another stream.

buf

```
protected byte buf[]
```

The buffer where data is stored.

count

```
protected int count
```

The number of bytes in the buffer.

Interface Index

Interface Index**ByteArrayOutputStream**

```
public ByteArrayOutputStream()
```

Creates a new ByteArrayOutputStream.

Interface Index**ByteArrayOutputStream**

```
public ByteArrayOutputStream(int size)
```

Creates a new ByteArrayOutputStream with the specified initial size.

Parameters:

size - the initial size

Interface Index

Interface Index**write**

```
public synchronized void write(int b)
```

Writes a byte to the buffer.

Parameters:

b - the byte

Overrides:

write in class OutputStream

Interface Index**write**

```
public synchronized void write(byte b[],
                               int off,
                               int len)
```

Writes bytes to the buffer.

Parameters:

b - the data to be written

off - the start offset in the data
len - the number of bytes that are written

Overrides:

write in class OutputStream

writeTo

```
public synchronized void writeTo(OutputStream out) throws IOException
```

Writes the contents of the buffer to another stream.

Parameters:

out - the output stream to write to

reset

```
public synchronized void reset()
```

Resets the buffer so that you can use it again without throwing away the already allocated buffer.

toByteArray

```
public synchronized byte[] toByteArray()
```

Returns a copy of the input data.

size

```
public int size()
```

Returns the current size of the buffer.

toString

```
public String toString()
```

Converts input data to a string.

Returns:

the string.

Overrides:

toString in class Object

toString

```
public String toString(int hibyte)
```

Converts input data to a string. The top 8 bits of each 16 bit Unicode character are set to hibyte.

Parameters:

hibyte - the bits set

Interface java.io.DataInput

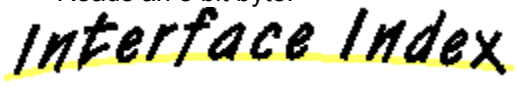
Interface java.io.DataInput

public interface **DataInput**
extends Object

DataInput is an interface describing streams that can read input in a machine-independent format.
See Also:
DataInputStream, DataOutput

readBoolean()
Reads in a boolean.

readByte()
Reads an 8 bit byte.



readChar()
Reads a 16 bit char.

readDouble()
Reads a 64 bit double.

readFloat()
Reads a 32 bit float.

readFully(byte[])
Reads bytes, blocking until all bytes are read.

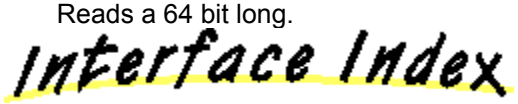
readFully(byte[], int, int)
Reads bytes, blocking until all bytes are read.

readInt()
Reads a 32 bit int.

readLine()

readLong()

Reads a 64 bit long.



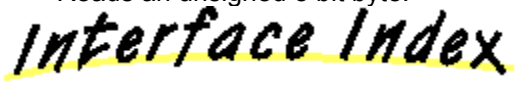
readShort()

Reads a 16 bit short.

readUTF()

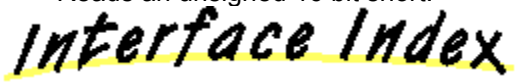
readUnsignedByte()

Reads an unsigned 8 bit byte.



readUnsignedShort()

Reads an unsigned 16 bit short.



skipBytes(int)

Skips bytes, block until all bytes are skipped.

readFully

```
public abstract void readFully(byte b[]) throws IOException
```

Reads bytes, blocking until all bytes are read.

Parameters:

b - the buffer into which the data is read

Throws: EOFException

If end of file is reached.

Throws: IOException

If other I/O error has occurred.

readFully

```
public abstract void readFully(byte b[],
                               int off,
                               int len) throws IOException
```

Reads bytes, blocking until all bytes are read.

Parameters:

b - the buffer into which the data is read

off - the start offset of the data

len - the maximum number of bytes to read

Throws: EOFException
If end of file is reached.

Throws: IOException
If other I/O error has occurred.

skipBytes

```
public abstract int skipBytes(int n) throws IOException
```

Skips bytes, block until all bytes are skipped.

Parameters:
n - the number of bytes to be skipped

Returns:
the actual number of bytes skipped.

Throws: EOFException
If end of file is reached.

Throws: IOException
If other I/O error has occurred.

readBoolean

```
public abstract boolean readBoolean() throws IOException
```

Reads in a boolean.

Returns:
the boolean read.

Throws: EOFException
If end of file is reached.

Throws: IOException
If other I/O error has occurred.

readByte

```
public abstract byte readByte() throws IOException
```

Reads an 8 bit byte.

Returns:
the 8 bit byte read.

Throws: EOFException
If end of file is reached.

Throws: IOException

If other I/O error has occurred.

readUnsignedByte

```
public abstract int readUnsignedByte() throws IOException
```

Reads an unsigned 8 bit byte.

Returns:

the 8 bit byte read.

Throws: EOFException

If end of file is reached.

Throws: IOException

If other I/O error has occurred.

readShort

```
public abstract short readShort() throws IOException
```

Reads a 16 bit short.

Returns:

the 16 bit short read.

Throws: EOFException

If end of file is reached.

Throws: IOException

If other I/O error has occurred.

readUnsignedShort

```
public abstract int readUnsignedShort() throws IOException
```

Reads an unsigned 16 bit short.

Returns:

the 16 bit short read.

Throws: EOFException

If end of file is reached.

Throws: IOException

If other I/O error has occurred.

 readChar

```
public abstract char readChar() throws IOException
```

Reads a 16 bit char.
Returns:
the 16 bit char read.
Throws: EOFException
If end of file is reached.
Throws: IOException
If other I/O error has occurred.

 readInt

```
public abstract int readInt() throws IOException
```

Reads a 32 bit int.
Returns:
the 32 bit integer read.
Throws: EOFException
If end of file is reached.
Throws: IOException
If other I/O error has occurred.

 readLong

```
public abstract long readLong() throws IOException
```

Reads a 64 bit long.
Returns:
the read 64 bit long.
Throws: EOFException
If end of file is reached.
Throws: IOException
If other I/O error has occurred.

 readFloat

```
public abstract float readFloat() throws IOException
```

Reads a 32 bit float.

Returns:
the 32 bit float read.

Throws: EOFException
If end of file is reached.

Throws: IOException
If other I/O error has occurred.

readDouble

```
public abstract double readDouble() throws IOException
```

Reads a 64 bit double.

Returns:
the 64 bit double read.

Throws: EOFException
If end of file is reached.

Throws: IOException
If other I/O error has occurred.

readLine

```
public abstract String readLine() throws IOException
```

readUTF

```
public abstract String readUTF() throws IOException
```

Class java.io.DataInputStream

Class java.io.DataInputStream

```
java.lang.Object
    |
    +----java.io.InputStream
            |
            +----java.io.FilterInputStream
                    |
                    +----java.io.DataInputStream
```

---

public class **DataInputStream**
extends FilterInputStream
implements DataInput

A data input stream that lets you read primitive Java data types from a stream in a portable way. Primitive data types are well understood types with associated operations. For example, Integers are considered primitive data types.

See Also:

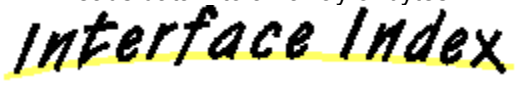
DataOutputStream

DataInputStream(InputStream)

Creates a new DataInputStream.

read(byte[])

Reads data into an array of bytes.

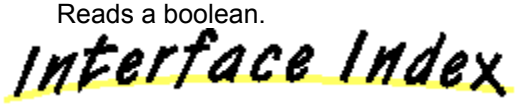


read(byte[], int, int)

Reads data into an array of bytes.

readBoolean()

Reads a boolean.

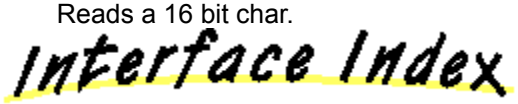


readByte()

Reads an 8 bit byte.

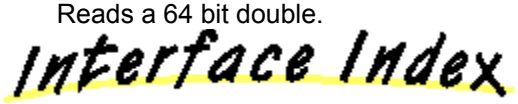
readChar()

Reads a 16 bit char.



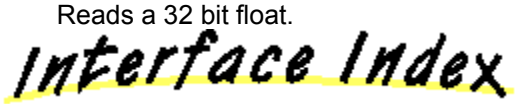
readDouble()

Reads a 64 bit double.



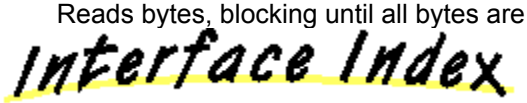
readFloat()

Reads a 32 bit float.



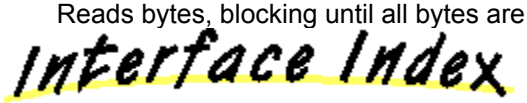
readFully(byte[])

Reads bytes, blocking until all bytes are read.



readFully(byte[], int, int)

Reads bytes, blocking until all bytes are read.

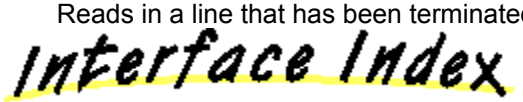


readInt()

Reads a 32 bit int.

readLine()

Reads in a line that has been terminated by a \n, \r, \r\n or EOF.



readLong()

Reads a 64 bit long.

readShort()

Reads a 16 bit short.

readUTF()

Reads a UTF format String.

readUTF(DataInput)

Reads a UTF format String from the given input stream.

readUnsignedByte()

Reads an unsigned 8 bit byte.

readUnsignedShort()

Reads 16 bit short.

skipBytes(int)
Skips bytes, blocks until all bytes are skipped.

DataInputStream

```
public DataInputStream(InputStream in)
```

Creates a new DataInputStream.
Parameters:
in - the input stream

read

```
public final int read(byte b[]) throws IOException
```

Reads data into an array of bytes. This method blocks until some input is available.
Parameters:
b - the buffer into which the data is read
Returns:
the actual number of bytes read, -1 is returned when the end of the stream is reached.
Throws: IOException
If an I/O error has occurred.
Overrides:
read in class FilterInputStream

read

```
public final int read(byte b[],
                      int off,
                      int len) throws IOException
```

Reads data into an array of bytes. This method blocks until some input is available.
Parameters:

b - the buffer into which the data is read
off - the start offset of the data
len - the maximum number of bytes read

Returns:

the actual number of bytes read, -1 is returned when the end of the stream is reached.

Throws: IOException

If an I/O error has occurred.

Overrides:

read in class FilterInputStream

readFully

```
public final void readFully(byte b[]) throws IOException
```

Reads bytes, blocking until all bytes are read.

Parameters:

b - the buffer into which the data is read

Throws: IOException

If an I/O error has occurred.

Throws: EOFException

If EOF reached before all bytes are read.

readFully

```
public final void readFully(byte b[],
                            int off,
                            int len) throws IOException
```

Reads bytes, blocking until all bytes are read.

Parameters:

b - the buffer into which the data is read
off - the start offset of the data
len - the maximum number of bytes read

Throws: IOException

If an I/O error has occurred.

Throws: EOFException

If EOF reached before all bytes are read.

skipBytes

```
public final int skipBytes(int n) throws IOException
```

Skips bytes, blocks until all bytes are skipped.
Parameters:
n - the number of bytes to be skipped
Returns:
the actual number of bytes skipped.
Throws: IOException
If an I/O error has occurred.

readBoolean

```
public final boolean readBoolean() throws IOException
```

Reads a boolean.
Returns:
the boolean read.

readByte

```
public final byte readByte() throws IOException
```

Reads an 8 bit byte.
Returns:
the 8 bit byte read.

readUnsignedByte

```
public final int readUnsignedByte() throws IOException
```

Reads an unsigned 8 bit byte.
Returns:
the 8 bit byte read.

readShort

```
public final short readShort() throws IOException
```

Reads a 16 bit short.

Returns:
the 16 bit short read.

readUnsignedShort

```
public final int readUnsignedShort() throws IOException
```

Reads 16 bit short.
Returns:
the 16 bit short read.

readChar

```
public final char readChar() throws IOException
```

Reads a 16 bit char.
Returns:
the read 16 bit char.

readInt

```
public final int readInt() throws IOException
```

Reads a 32 bit int.
Returns:
the 32 bit integer read.

readLong

```
public final long readLong() throws IOException
```

Reads a 64 bit long.
Returns:
the 64 bit long read.

 readFloat

```
public final float readFloat() throws IOException
```

Reads a 32 bit float.
Returns:
the read 32 bit float.

 readDouble

```
public final double readDouble() throws IOException
```

Reads a 64 bit double.
Returns:
the 64 bit double read.

 readLine

```
public final String readLine() throws IOException
```

Reads in a line that has been terminated by a \n, \r, \r\n or EOF.
Returns:
a String copy of the line.

 readUTF

```
public final String readUTF() throws IOException
```

Reads a UTF format String.
Returns:
the String.

 readUTF

```
public final static String readUTF(DataInput in) throws IOException
```

Reads a UTF format String from the given input stream.

Returns:

the String.

Interface java.DataOutput

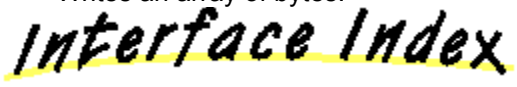
Interface java.DataOutput

public interface **DataOutput**
extends Object

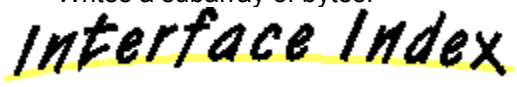
DataOutput is an interface describing streams that can write output in a machine-independent format.
See Also:
DataOutputStream, DataInput

write(int)
Writes a byte.

write(byte[])
Writes an array of bytes.

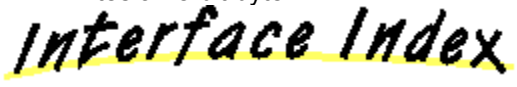


write(byte[], int, int)
Writes a subarray of bytes.

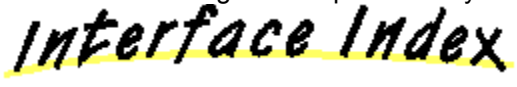


writeBoolean(boolean)
Writes a boolean.

writeByte(int)
Writes an 8 bit byte.



writeBytes(String)
Writes a String as a sequence of bytes.



writeChar(int)
Writes a 16 bit char.

writeChars(String)
Writes a String as a sequence of chars.

writeDouble(double)
Writes a 64 bit double.

writeFloat(float)

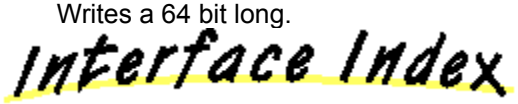
Writes a 32 bit float.

writeInt(int)

Writes a 32 bit int.

writeLong(long)

Writes a 64 bit long.



writeShort(int)

Writes a 16 bit short.

writeUTF(String)

Writes a String in UTF format.

write

```
public abstract void write(int b) throws IOException
```

Writes a byte. Will block until the byte is actually written.

Parameters:

b - the byte to be written

Throws: IOException

If an I/O error has occurred.

write

```
public abstract void write(byte b[]) throws IOException
```

Writes an array of bytes.

Parameters:

b - the data to be written

Throws: IOException

If an I/O error has occurred.

write

```
public abstract void write(byte b[],
                           int off,
                           int len) throws IOException
```

Writes a subarray of bytes.
Parameters:
b - the data to be written
off - the start offset in the data
len - the number of bytes that are written
Throws: IOException
If an I/O error has occurred.

writeBoolean

```
public abstract void writeBoolean(boolean v) throws IOException
```

Writes a boolean.
Parameters:
v - the boolean to be written

writeByte

```
public abstract void writeByte(int v) throws IOException
```

Writes an 8 bit byte.
Parameters:
v - the byte value to be written

writeShort

```
public abstract void writeShort(int v) throws IOException
```

Writes a 16 bit short.
Parameters:
v - the short value to be written

writeChar

```
public abstract void writeChar(int v) throws IOException
```

Writes a 16 bit char.

Parameters:

v - the char value to be written

writeInt

```
public abstract void writeInt(int v) throws IOException
```

Writes a 32 bit int.

Parameters:

v - the integer value to be written

writeLong

```
public abstract void writeLong(long v) throws IOException
```

Writes a 64 bit long.

Parameters:

v - the long value to be written

writeFloat

```
public abstract void writeFloat(float v) throws IOException
```

Writes a 32 bit float.

Parameters:

v - the float value to be written

writeDouble

```
public abstract void writeDouble(double v) throws IOException
```

Writes a 64 bit double.
Parameters:
v - the double value to be written

writeBytes

```
public abstract void writeBytes(String s) throws IOException
```

Writes a String as a sequence of bytes.
Parameters:
s - the String of bytes to be written

writeChars

```
public abstract void writeChars(String s) throws IOException
```

Writes a String as a sequence of chars.
Parameters:
s - the String of chars to be written

writeUTF

```
public abstract void writeUTF(String str) throws IOException
```

Writes a String in UTF format.
Parameters:
str - the String in UTF format

Class java.io.DataOutputStream

Class java.io.DataOutputStream

```
java.lang.Object
    |
    +----java.io.OutputStream
            |
            +----java.io.FilterOutputStream
                    |
                    +----java.io.DataOutputStream
```

---

public class **DataOutputStream**
extends FilterOutputStream
implements DataOutput

This class lets you write primitive Java data types to a stream in a portable way. Primitive data types are well understood types with associated operations. For example, an Integer is considered to be a good primitive data type. The data can be converted back using a DataInputStream.

written

The number of bytes written so far.

DataOutputStream(OutputStream)

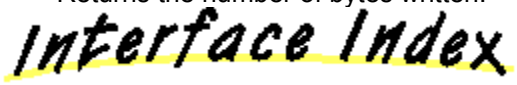
Creates a new DataOutputStream.

flush()

Flushes the stream.

size()

Returns the number of bytes written.

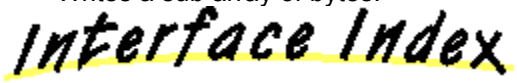


write(int)

Writes a byte.

write(byte[], int, int)

Writes a sub array of bytes.

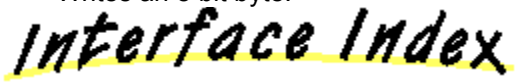


writeBoolean(boolean)

Writes a boolean.

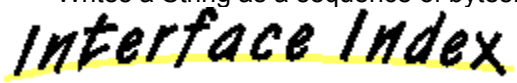
writeByte(int)

Writes an 8 bit byte.



writeBytes(String)

Writes a String as a sequence of bytes.

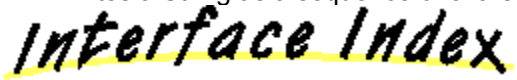


writeChar(int)

Writes a 16 bit char.

writeChars(String)

Writes a String as a sequence of chars.



writeDouble(double)

Writes a 64 bit double.

writeFloat(float)

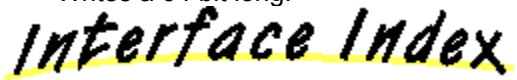
Writes a 32 bit float.

writeInt(int)

Writes a 32 bit int.

writeLong(long)

Writes a 64 bit long.



writeShort(int)

Writes a 16 bit short.

writeUTF(String)

Writes a String in UTF format.

Interface Index

written

```
protected int written
```

The number of bytes written so far.

Interface Index

Interface Index

DataOutputStream

```
public DataOutputStream(OutputStream out)
```

Creates a new DataOutputStream.
Parameters:
out - the output stream

Interface Index

Interface Index

write

```
public synchronized void write(int b) throws IOException
```

Writes a byte. Will block until the byte is actually written.
Parameters:
b - the byte to be written
Throws: IOException
If an I/O error has occurred.
Overrides:
write in class FilterOutputStream

Interface Index

write

```
public synchronized void write(byte b[],
```

```
                                int off,
                                int len) throws IOException
```

Writes a sub array of bytes.
Parameters:
b - the data to be written
off - the start offset in the data
len - the number of bytes that are written
Throws: IOException
If an I/O error has occurred.
Overrides:
write in class FilterOutputStream

flush

```
public void flush() throws IOException
```

Flushes the stream. This will write any buffered output bytes.
Throws: IOException
If an I/O error has occurred.
Overrides:
flush in class FilterOutputStream

writeBoolean

```
public final void writeBoolean(boolean v) throws IOException
```

Writes a boolean.
Parameters:
v - the boolean to be written

writeByte

```
public final void writeByte(int v) throws IOException
```

Writes an 8 bit byte.
Parameters:
v - the byte value to be written

Interface Index **writeShort**

```
public final void writeShort(int v) throws IOException
```

Writes a 16 bit short.
Parameters:
v - the short value to be written

Interface Index **writeChar**

```
public final void writeChar(int v) throws IOException
```

Writes a 16 bit char.
Parameters:
v - the char value to be written

Interface Index **writeInt**

```
public final void writeInt(int v) throws IOException
```

Writes a 32 bit int.
Parameters:
v - the integer value to be written

Interface Index **writeLong**

```
public final void writeLong(long v) throws IOException
```

Writes a 64 bit long.
Parameters:
v - the long value to be written

Interface Index **writeFloat**

```
public final void writeFloat(float v) throws IOException
```

Writes a 32 bit float.
Parameters:
v - the float value to be written

writeDouble

```
public final void writeDouble(double v) throws IOException
```

Writes a 64 bit double.
Parameters:
v - the double value to be written

writeBytes

```
public final void writeBytes(String s) throws IOException
```

Writes a String as a sequence of bytes.
Parameters:
s - the String of bytes to be written

writeChars

```
public final void writeChars(String s) throws IOException
```

Writes a String as a sequence of chars.
Parameters:
s - the String of chars to be written

writeUTF

```
public final void writeUTF(String str) throws IOException
```

Writes a String in UTF format.
Parameters:
str - the String in UTF format

size

```
public final int size()
```

Returns the number of bytes written.

Returns:

the number of bytes written thus far.

Class java.io.EOFException

Class java.io.EOFException

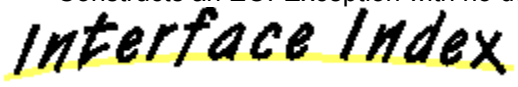
```
java.lang.Object
    |
    +----java.lang.Throwable
            |
            +----java.lang.Exception
                    |
                    +----java.io.IOException
                            |
                            +----java.io.EOFException
```

---

public class **EOFException**
extends IOException

Signals that and EOF has been reached unexpectedly during input.
See Also:
IOException, DataInputStream

EOFException()
Constructs an EOFException with no detail message.



EOFException(String)
Constructs an EOFException with the specified detail message.

EOFException

```
public EOFException()
```

Constructs an EOFException with no detail message. A detail message is a String that describes this particular exception.

EOFException

```
public EOFException(String s)
```

Constructs an EOFException with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class java.io.File

Class java.io.File

```
java.lang.Object
   |
   +----java.io.File
```

---

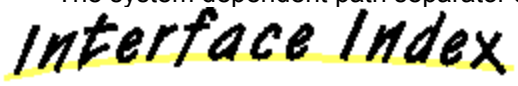
public class **File**
extends Object

This class represents a file name of the host file system. The file name can be relative or absolute. It must use the file name conventions of the host platform.

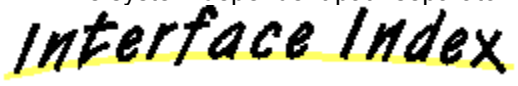
The intention is to provide an abstraction that deals with most of the system-dependent file name features such as the separator character, root, device name, etc. Not all features are currently fully implemented.

Note that whenever a file name or path is used it is assumed that the host's file name conventions are used.

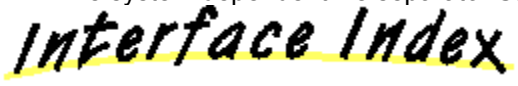
pathSeparator
The system dependent path separator string.



pathSeparatorChar
The system dependent path separator character.



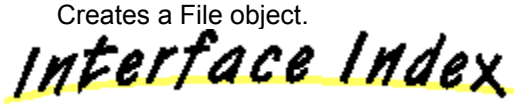
separator
The system dependent file separator String.



separatorChar
The system dependent file separator character.

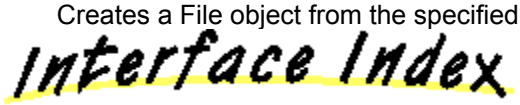
File(String)

Creates a File object.



File(String, String)

Creates a File object from the specified directory.

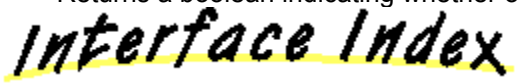


File(File, String)

Creates a File object (given a directory File object).

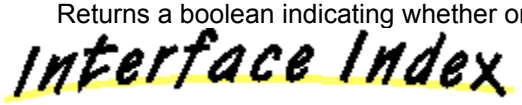
canRead()

Returns a boolean indicating whether or not a readable file exists.



canWrite()

Returns a boolean indicating whether or not a writable file exists.

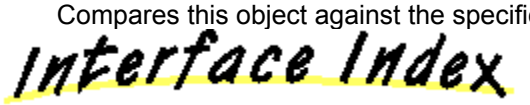


delete()

Deletes the specified file.

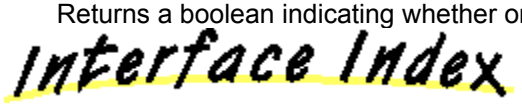
equals(Object)

Compares this object against the specified object.



exists()

Returns a boolean indicating whether or not a file exists.

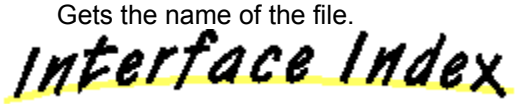


getAbsolutePath()

Gets the absolute path of the file.

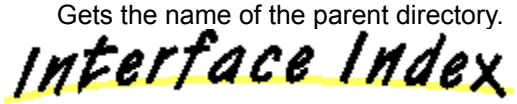
getName()

Gets the name of the file.



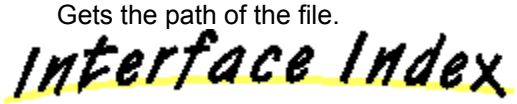
getParent()

Gets the name of the parent directory.



getPath()

Gets the path of the file.

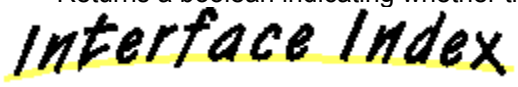


hashCode()

Computes a hashcode for the file.

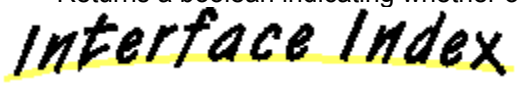
isAbsolute()

Returns a boolean indicating whether the file name is absolute.



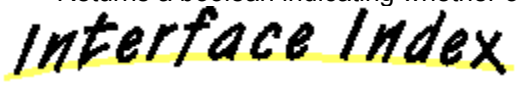
isDirectory()

Returns a boolean indicating whether or not a directory file exists.



isFile()

Returns a boolean indicating whether or not a normal file exists.

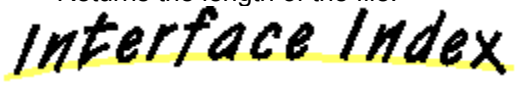


lastModified()

Returns the last modification time.

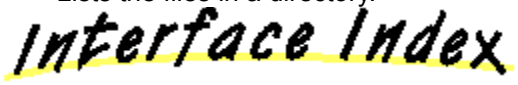
length()

Returns the length of the file.



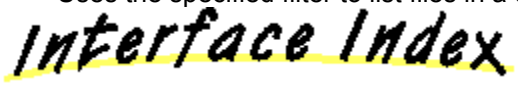
list()

Lists the files in a directory.



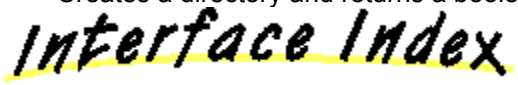
list(FilenameFilter)

Uses the specified filter to list files in a directory.



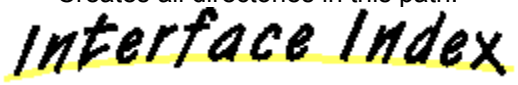
mkdir()

Creates a directory and returns a boolean indicating the success of the creation.



mkdirs()

Creates all directories in this path.



renameTo(File)

Renames a file and returns a boolean indicating whether or not this method was successful.

toString()

Returns a String object representing this file's path.

separator

```
public final static String separator
```

The system dependent file separator String.

separatorChar

```
public final static char separatorChar
```

The system dependent file separator character.

Interface Index **pathSeparator**

```
public final static String pathSeparator
```

The system dependent path separator string.

Interface Index **pathSeparatorChar**

```
public final static char pathSeparatorChar
```

The system dependent path separator character.

Interface Index

Interface Index **File**

```
public File(String path)
```

Creates a File object.
Parameters:
path - the file path
Throws: NullPointerException
If the file path is equal to null.

Interface Index **File**

```
public File(String path,
            String name)
```

Creates a File object from the specified directory.
Parameters:
path - the directory path
name - the file name

File

```
public File(File dir,
            String name)
```

Creates a File object (given a directory File object).
Parameters:
dir - the directory
name - the file name

getName

```
public String getName()
```

Gets the name of the file. This method does not include the directory.
Returns:
the file name.

getPath

```
public String getPath()
```

Gets the path of the file.
Returns:
the file path.

getAbsolutePath

```
public String getAbsolutePath()
```

Gets the absolute path of the file.
Returns:
the absolute file path.

getParent

```
public String getParent()
```

Gets the name of the parent directory.

Returns:

the parent directory, or null if one is not found.

exists

```
public boolean exists()
```

Returns a boolean indicating whether or not a file exists.

canWrite

```
public boolean canWrite()
```

Returns a boolean indicating whether or not a writable file exists.

canRead

```
public boolean canRead()
```

Returns a boolean indicating whether or not a readable file exists.

isFile

```
public boolean isFile()
```

Returns a boolean indicating whether or not a normal file exists.

isDirectory

```
public boolean isDirectory()
```

Returns a boolean indicating whether or not a directory file exists.

isAbsolute

```
public boolean isAbsolute()
```

Returns a boolean indicating whether the file name is absolute.

lastModified

```
public long lastModified()
```

Returns the last modification time. The return value should only be used to compare modification dates. It is meaningless as an absolute time.

length

```
public long length()
```

Returns the length of the file.

mkdir

```
public boolean mkdir()
```

Creates a directory and returns a boolean indicating the success of the creation.

renameTo

```
public boolean renameTo(File dest)
```

Renames a file and returns a boolean indicating whether or not this method was successful.
Parameters:
dest - the new file name

mkdirs

```
public boolean mkdirs()
```

Creates all directories in this path. This method returns true if all directories in this path are created.

list

```
public String[] list()
```

Lists the files in a directory. Works only on directories.

Returns:

an array of file names. This list will include all files in the directory except the equivalent of "." and ".." .

list

```
public String[] list(FilenameFilter filter)
```

Uses the specified filter to list files in a directory.

Parameters:

filter - the filter used to select file names

Returns:

the filter selected files in this directory.

See Also:

FilenameFilter

delete

```
public boolean delete()
```

Deletes the specified file. Returns true if the file could be deleted.

hashCode

```
public int hashCode()
```

Computes a hashcode for the file.

Overrides:

hashCode in class Object

equals

```
public boolean equals(Object obj)
```

Compares this object against the specified object.

Parameters:

obj - the object to compare with

Returns:

true if the objects are the same; false otherwise.

Overrides:

equals in class Object

toString

```
public String toString()
```

Returns a String object representing this file's path.

Overrides:

toString in class Object

Class java.io.FileDescriptor

Class java.io.FileDescriptor

```
java.lang.Object
   |
   +----java.io.FileDescriptor
```

---

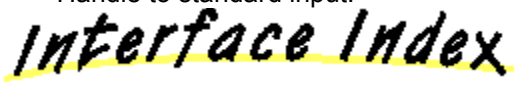
public final class **FileDescriptor**
extends Object

err

Handle to standard error.

in

Handle to standard input.



out

Handle to standard output.

FileDescriptor()

valid()

Determines whether the file descriptor object is valid.

in

```
public final static FileDescriptor in
```

Handle to standard input.

out

```
public final static FileDescriptor out
```

Handle to standard output.

err

```
public final static FileDescriptor err
```

Handle to standard error.

FileDescriptor

```
public FileDescriptor()
```

valid

```
public boolean valid()
```

Determines whether the file descriptor object is valid.

Class java.io.FileInputStream

Class java.io.FileInputStream

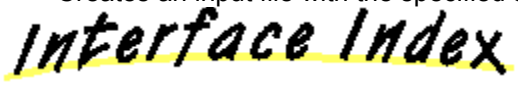
```
java.lang.Object
    |
    +----java.io.InputStream
             |
             +----java.io.FileInputStream
```

---

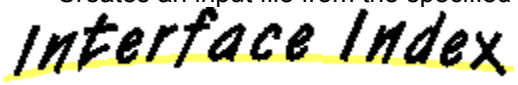
public class **FileInputStream**
extends InputStream

File input stream, can be constructed from a file descriptor or a file name.
See Also:
FileOutputStream, File

FileInputStream(String)
Creates an input file with the specified system dependent file name.

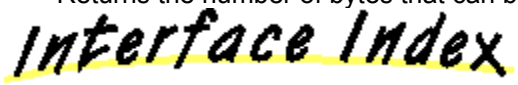


FileInputStream(File)
Creates an input file from the specified File object.



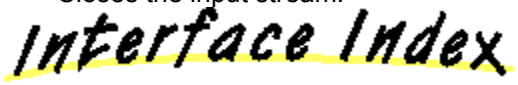
FileInputStream(FileDescriptor)

available()
Returns the number of bytes that can be read without blocking.



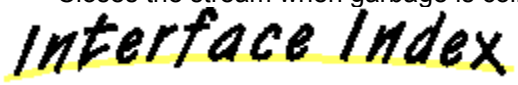
close()

Closes the input stream.



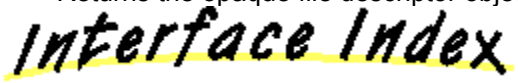
finalize()

Closes the stream when garbage is collected.



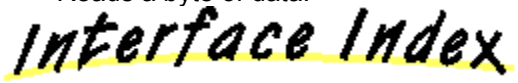
getFD()

Returns the opaque file descriptor object associated with this stream.



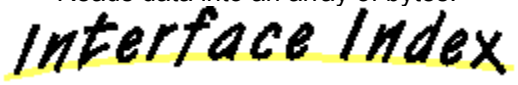
read()

Reads a byte of data.



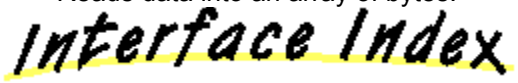
read(byte[])

Reads data into an array of bytes.



read(byte[], int, int)

Reads data into an array of bytes.



skip(long)

Skips n bytes of input.

FileInputStream

```
public FileInputStream(String name) throws FileNotFoundException
```

Creates an input file with the specified system dependent file name.

Parameters:

name - the system dependent file name

Throws: IOException

If the file is not found.

FileInputStream

```
public FileInputStream(File file) throws FileNotFoundException
```

Creates an input file from the specified File object.

Parameters:

file - the file to be opened for reading

Throws: IOException

If the file is not found.

FileInputStream

```
public FileInputStream(FileDescriptor fdObj)
```

read

```
public int read() throws IOException
```

Reads a byte of data. This method will block if no input is available.

Returns:

the byte read, or -1 if the end of the stream is reached.

Throws: IOException

If an I/O error has occurred.

Overrides:

read in class InputStream

read

```
public int read(byte b[]) throws IOException
```

Reads data into an array of bytes. This method blocks until some input is available.

Parameters:

b - the buffer into which the data is read

Returns:

the actual number of bytes read. -1 is returned if the end of stream is reached.

Throws: IOException

If an I/O error has occurred.

Overrides:

read in class InputStream

read

```
public int read(byte b[],
                int off,
                int len) throws IOException
```

Reads data into an array of bytes. This method blocks until some input is available.

Parameters:

b - the buffer into which the data is read

off - the start offset of the data

len - the maximum number of bytes read

Returns:

the actual number of bytes read. -1 is returned when the end of the stream is reached.

Throws: IOException

If an I/O error has occurred.

Overrides:

read in class InputStream

skip

```
public long skip(long n) throws IOException
```

Skips n bytes of input.

Parameters:

n - the number of bytes to be skipped

Returns:

the actual number of bytes skipped.

Throws: IOException

If an I/O error has occurred.

Overrides:

skip in class InputStream

available

```
public int available() throws IOException
```

Returns the number of bytes that can be read without blocking.

Returns:

the number of available bytes, which is initially equal to the file size.

Overrides:

available in class InputStream

close

```
public void close() throws IOException
```

Closes the input stream. This method must be called to release any resources associated with the stream.

Throws: IOException
If an I/O error has occurred.
Overrides:
close in class InputStream

getFD

```
public final FileDescriptor getFD() throws IOException
```

Returns the opaque file descriptor object associated with this stream.
Returns:
the file descriptor.

finalize

```
protected void finalize() throws IOException
```

Closes the stream when garbage is collected.
Overrides:
finalize in class Object

Interface java.io.FilenameFilter

Interface java.io.FilenameFilter

public interface **FilenameFilter**
extends Object

A filter interface for file names.
See Also:
File

accept(File, String)
Determines whether a name should be included in a file list.

accept

```
public abstract boolean accept(File dir,
                               String name)
```

Determines whether a name should be included in a file list.
Parameters:
dir - the directory in which the file was found
name - the name of the file
Returns:
true if name should be included in file list; false otherwise.

Class java.io.FileNotFoundException

Class java.io.FileNotFoundException

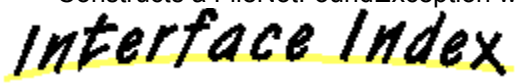
```
java.lang.Object
    |
    +----java.lang.Throwable
            |
            +----java.lang.Exception
                    |
                    +----java.io.IOException
                            |
                            +----java.io.FileNotFoundException
```

---

public class **FileNotFoundException**
extends IOException

Signals that a file was not found.

FileNotFoundException()
Constructs a FileNotFoundException with no detail message.



FileNotFoundException(String)
Constructs a FileNotFoundException with the specified detail message.

FileNotFoundException

```
public FileNotFoundException()
```

Constructs a FileNotFoundException with no detail message. A detail message is a String that describes this particular exception.

FileNotFoundException

```
public FileNotFoundException(String s)
```

Constructs a FileNotFoundException with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class java.io.FileOutputStream

Class java.io.FileOutputStream

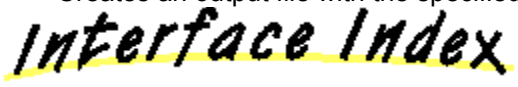
```
java.lang.Object
    |
    +----java.io.OutputStream
             |
             +----java.io.FileOutputStream
```

---

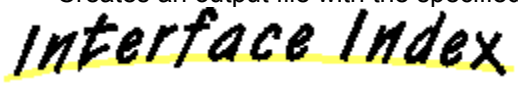
public class **FileOutputStream**
extends OutputStream

File output stream, can be constructed from a file descriptor or a file name.
See Also:
FileInputStream, File

FileOutputStream(String)
Creates an output file with the specified system dependent file name.



FileOutputStream(File)
Creates an output file with the specified File object.

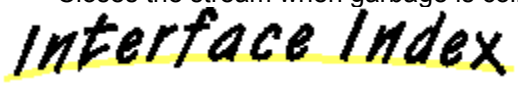


FileOutputStream(FileDescriptor)

close()
Closes the stream.

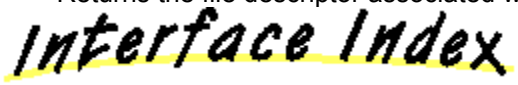
finalize()

Closes the stream when garbage is collected.



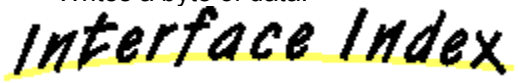
getFD()

Returns the file descriptor associated with this stream.



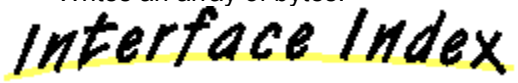
write(int)

Writes a byte of data.



write(byte[])

Writes an array of bytes.



write(byte[], int, int)

Writes a sub array of bytes.

FileOutputStream

```
public FileOutputStream(String name) throws IOException
```

Creates an output file with the specified system dependent file name.

Parameters:

name - the system dependent file name

Throws: IOException

If the file is not found.

FileOutputStream

```
public FileOutputStream(File file) throws IOException
```

Creates an output file with the specified File object.

Parameters:

file - the file to be opened for reading

Throws: IOException

If the file is not found.

FileOutputStream

```
public FileOutputStream(FileDescriptor fdObj)
```

Interface Index

write

```
public void write(int b) throws IOException
```

Writes a byte of data. This method will block until the byte is actually written.

Parameters:

b - the byte to be written

Throws: IOException

If an I/O error has occurred.

Overrides:

write in class OutputStream

Interface Index

write

```
public void write(byte b[]) throws IOException
```

Writes an array of bytes. Will block until the bytes are actually written.

Parameters:

b - the data to be written

Throws: IOException

If an I/O error has occurred.

Overrides:

write in class OutputStream

Interface Index

write

```
public void write(byte b[],
                  int off,
                  int len) throws IOException
```

Writes a sub array of bytes.

Parameters:

b - the data to be written

off - the start offset in the data

len - the number of bytes that are written

Throws: IOException

If an I/O error has occurred.

Overrides:

write in class OutputStream

close

```
public void close() throws IOException
```

Closes the stream. This method must be called to release any resources associated with the stream.

Throws: IOException
If an I/O error has occurred.

Overrides:
close in class OutputStream

getFD

```
public final FileDescriptor getFD() throws IOException
```

Returns the file descriptor associated with this stream.

Returns:
the file descriptor.

finalize

```
protected void finalize() throws IOException
```

Closes the stream when garbage is collected.

Overrides:
finalize in class Object

Class java.io.FilterInputStream

Class java.io.FilterInputStream

```
java.lang.Object
    |
    +----java.io.InputStream
             |
             +----java.io.FilterInputStream
```

---

public class **FilterInputStream**
extends InputStream

Abstract class representing a filtered input stream of bytes. This class is the basis for enhancing input stream functionality. It allows multiple input stream filters to be chained together, each providing additional functionality.

in

The actual input stream.

FilterInputStream(InputStream)

Creates an input stream filter.

available()

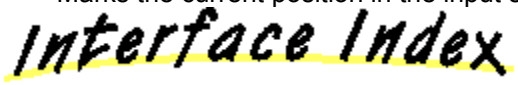
Returns the number of bytes that can be read without blocking.

close()

Closes the input stream.

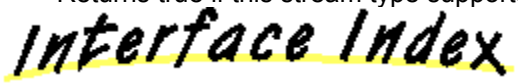
mark(int)

Marks the current position in the input stream.



markSupported()

Returns true if this stream type supports mark/reset.

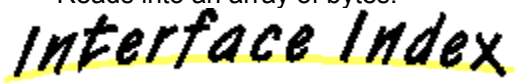


read()

Reads a byte.

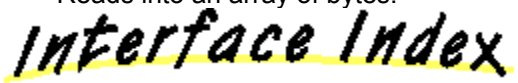
read(byte[])

Reads into an array of bytes.



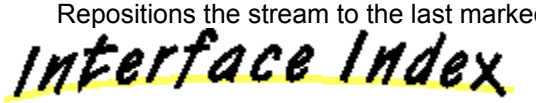
read(byte[], int, int)

Reads into an array of bytes.



reset()

Repositions the stream to the last marked position.



skip(long)

Skips bytes of input.

in

```
protected InputStream in
```

The actual input stream.

FilterInputStream

```
protected FilterInputStream(InputStream in)
```

Creates an input stream filter.

Parameters:
in - the input stream

Interface Index

Interface Index **read**

```
public int read() throws IOException
```

Reads a byte. Will block if no input is available.
Returns:
the byte read, or -1 if the end of the stream is reached.
Throws: IOException
If an I/O error has occurred.
Overrides:
read in class InputStream

Interface Index **read**

```
public int read(byte b[]) throws IOException
```

Reads into an array of bytes. Blocks until some input is available.
Parameters:
b - the buffer into which the data is read
Returns:
the actual number of bytes read. Returns -1 when the end of the stream is reached.
Throws: IOException
If an I/O error has occurred.
Overrides:
read in class InputStream

Interface Index **read**

```
public int read(byte b[],
                int off,
                int len) throws IOException
```

Reads into an array of bytes. Blocks until some input is available. This method should be overridden in a subclass for efficiency (the default implementation reads 1 byte at a time).

Parameters:
b - the buffer into which the data is read
off - the start offset of the data
len - the maximum number of bytes read
Returns:
the actual number of bytes read. Returns -1 when the end of the stream is reached.
Throws: IOException
If an I/O error has occurred.
Overrides:
read in class InputStream

skip

```
public long skip(long n) throws IOException
```

Skips bytes of input.
Parameters:
n - bytes to be skipped
Returns:
actual number of bytes skipped
Throws: IOException
If an I/O error has occurred.
Overrides:
skip in class InputStream

available

```
public int available() throws IOException
```

Returns the number of bytes that can be read without blocking.
Returns:
the number of available bytes
Overrides:
available in class InputStream

close

```
public void close() throws IOException
```

Closes the input stream. Must be called to release any resources associated with the stream.
Throws: IOException
If an I/O error has occurred.

Overrides:
close in class InputStream

mark

```
public synchronized void mark(int readlimit)
```

Marks the current position in the input stream. A subsequent call to reset() will reposition the stream at the last marked position so that subsequent reads will re-read the same bytes. The stream promises to allow readlimit bytes to be read before the mark position gets invalidated.

Parameters:
readlimit - the maximum limit of bytes allowed tobe read before the mark position becomes invalid.

Overrides:
mark in class InputStream

reset

```
public synchronized void reset() throws IOException
```

Repositions the stream to the last marked position. If the stream has not been marked, or if the mark has been invalidated, an IOException is thrown. Stream marks are intended to be used in situations where you need to read ahead a little to see what's in the stream. Often this is most easily done by invoking some general parser. If the stream is of the type handled by the parse, it just chugs along happily. If the stream is not of that type, the parser should toss an exception when it fails. If this happens within readlimit bytes, it allows the outer code to reset the stream and try another parser.

Overrides:
reset in class InputStream

markSupported

```
public boolean markSupported()
```

Returns true if this stream type supports mark/reset.

Overrides:
markSupported in class InputStream

Class java.io.FilterOutputStream

Class java.io.FilterOutputStream

```
java.lang.Object
    |
    +----java.io.OutputStream
             |
             +----java.io.FilterOutputStream
```

---

public class **FilterOutputStream**
extends OutputStream

Abstract class representing a filtered output stream of bytes. This class is the basis for enhancing output stream functionality. It allows multiple output stream filters to be chained together, each providing additional functionality.

out

The actual output stream.

FilterOutputStream(OutputStream)

Creates an output stream filter.

close()

Closes the stream.

flush()

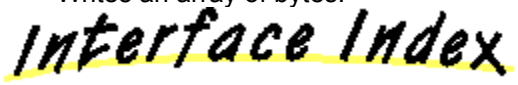
Flushes the stream.

write(int)

Writes a byte.

write(byte[])

Writes an array of bytes.



write(byte[], int, int)

Writes a subarray of bytes.

out

```
protected OutputStream out
```

The actual output stream.

FilterOutputStream

```
public FilterOutputStream(OutputStream out)
```

Creates an output stream filter.

Parameters:

out - the output stream

write

```
public void write(int b) throws IOException
```

Writes a byte. Will block until the byte is actually written.

Parameters:

b - the byte

Throws: IOException

If an I/O error has occurred.

Overrides:

write in class OutputStream

write

```
public void write(byte b[]) throws IOException
```

Writes an array of bytes. Will block until the bytes are actually written.

Parameters:

b - the data to be written

Throws: IOException

If an I/O error has occurred.

Overrides:

write in class OutputStream

write

```
public void write(byte b[],
                  int off,
                  int len) throws IOException
```

Writes a subarray of bytes. To be efficient it should be overridden in a subclass.

Parameters:

b - the data to be written

off - the start offset in the data

len - the number of bytes that are written

Throws: IOException

If an I/O error has occurred.

Overrides:

write in class OutputStream

flush

```
public void flush() throws IOException
```

Flushes the stream. This will write any buffered output bytes.
Throws: IOException
If an I/O error has occurred.
Overrides:
flush in class OutputStream

close

```
public void close() throws IOException
```

Closes the stream. This method must be called to release any resources associated with the stream.
Throws: IOException
If an I/O error has occurred.
Overrides:
close in class OutputStream

Class java.io.InputStream

Class java.io.InputStream

```
java.lang.Object
    |
    +----java.io.InputStream
```

---

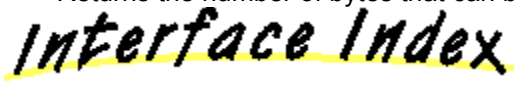
public class **InputStream**
extends Object

An abstract class representing an input stream of bytes. All InputStreams are based on this class.
See Also:
OutputStream, FilterInputStream, BufferedInputStream, DataInputStream , ByteArrayInputStream, PushbackInputStream

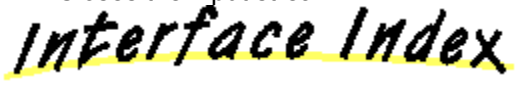
---

InputStream()

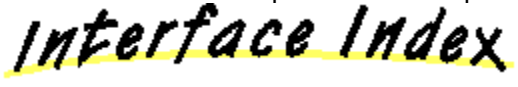
available()
Returns the number of bytes that can be read without blocking.



close()
Closes the input stream.



mark(int)
Marks the current position in the input stream.



markSupported()
Returns a boolean indicating whether or not this stream type supports mark/reset.

Interface Index

read()

Reads a byte of data.

Interface Index

read(byte[])

Reads into an array of bytes.

Interface Index

read(byte[], int, int)

Reads into an array of bytes.

Interface Index

reset()

Repositions the stream to the last marked position.

Interface Index

skip(long)

Skips n bytes of input.

Interface Index

Interface Index

InputStream

```
public InputStream()
```

Interface Index

Interface Index

read

```
public abstract int read() throws IOException
```

Reads a byte of data. This method will block if no input is available.

Returns:

the byte read, or -1 if the end of the stream is reached.

Throws: IOException

If an I/O error has occurred.

Interface Index

read

```
public int read(byte b[]) throws IOException
```

Reads into an array of bytes. This method will block until some input is available.

Parameters:

b - the buffer into which the data is read

Returns:

the actual number of bytes read, -1 is returned when the end of the stream is reached.

Throws: IOException

If an I/O error has occurred.

 read

```
public int read(byte b[],
                int off,
                int len) throws IOException
```

Reads into an array of bytes. This method will block until some input is available.

Parameters:

b - the buffer into which the data is read

off - the start offset of the data

len - the maximum number of bytes read

Returns:

the actual number of bytes read, -1 is returned when the end of the stream is reached.

Throws: IOException

If an I/O error has occurred.

 skip

```
public long skip(long n) throws IOException
```

Skips n bytes of input.

Parameters:

n - the number of bytes to be skipped

Returns:

the actual number of bytes skipped.

Throws: IOException

If an I/O error has occurred.

 available

```
public int available() throws IOException
```

Returns the number of bytes that can be read without blocking.

Returns:

the number of available bytes.

close

```
public void close() throws IOException
```

Closes the input stream. Must be called to release any resources associated with the stream.
Throws: IOException
If an I/O error has occurred.

mark

```
public synchronized void mark(int readlimit)
```

Marks the current position in the input stream. A subsequent call to reset() will reposition the stream at the last marked position so that subsequent reads will re-read the same bytes. The stream promises to allow readlimit bytes to be read before the mark position gets invalidated.
Parameters:
readlimit - the maximum limit of bytes allowed to be read before the mark position becomes invalid.

reset

```
public synchronized void reset() throws IOException
```

Repositions the stream to the last marked position. If the stream has not been marked, or if the mark has been invalidated, an IOException is thrown. Stream marks are intended to be used in situations where you need to read ahead a little to see what's in the stream. Often this is most easily done by invoking some general parser. If the stream is of the type handled by the parser, it just chugs along happily. If the stream is not of that type, the parser should toss an exception when it fails, which, if it happens within readlimit bytes, allows the outer code to reset the stream and try another parser.
Throws: IOException
If the stream has not been marked or if the mark has been invalidated.

markSupported

```
public boolean markSupported()
```

Returns a boolean indicating whether or not this stream type supports mark/reset.

Returns:

true if this stream type supports mark/reset; false otherwise.

Class java.io.InterruptedIOException

Class java.io.InterruptedIOException

```
java.lang.Object
    |
    +----java.lang.Throwable
             |
             +----java.lang.Exception
                      |
                      +----java.io.IOException
                               |
                               +----java.io.InterruptedIOException
```

---

public class **InterruptedIOException**
extends IOException

Signals that an I/O operation has been interrupted.
See Also:
InputStream, OutputStream

bytesTransferred
Reports how many bytes had been transferred as part of the IO operation before it was interrupted.

InterruptedIOException()
Constructs an IOException with no detail message.

InterruptedIOException(String)
Constructs an IOException with the specified detail message.

bytesTransferred

```
public int bytesTransferred
```

Reports how many bytes had been transferred as part of the IO operation before it was interrupted.

InterruptedIOException

```
public InterruptedIOException()
```

Constructs an IOException with no detail message. A detail message is a String that describes this particular exception.

InterruptedIOException

```
public InterruptedIOException(String s)
```

Constructs an IOException with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class java.io.IOException

Class java.io.IOException

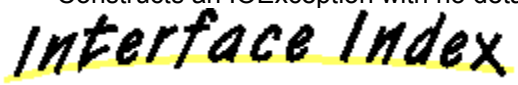
```
java.lang.Object
    |
    +----java.lang.Throwable
            |
            +----java.lang.Exception
                    |
                    +----java.io.IOException
```

---

public class **IOException**
extends Exception

Signals that an I/O exception has occurred.
See Also:
InputStream, OutputStream

IOException()
Constructs an IOException with no detail message.



IOException(String)
Constructs an IOException with the specified detail message.

IOException

```
public IOException()
```

Constructs an IOException with no detail message. A detail message is a String that describes this

particular exception.

IOException

```
public IOException(String s)
```

Constructs an IOException with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class java.io.LineNumberInputStream

Class java.io.LineNumberInputStream

```
java.lang.Object
    |
    +----java.io.InputStream
            |
            +----java.io.FilterInputStream
                    |
                    +----java.io.LineNumberInputStream
```

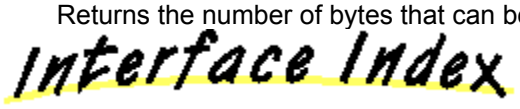
---

public class **LineNumberInputStream**
extends FilterInputStream

An input stream that keeps track of line numbers.

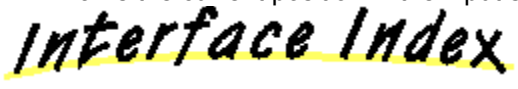
LineNumberInputStream(InputStream)
Constructs a new LineNumberInputStream initialized with the specified input stream.

available()
Returns the number of bytes that can be read without blocking.



getLineNumber()
Returns the current line number.

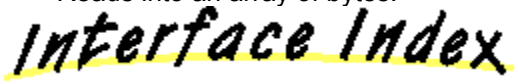
mark(int)
Marks the current position in the input stream.



read()
Reads a byte of data.

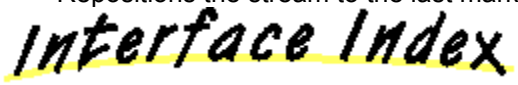
read(byte[], int, int)

Reads into an array of bytes.



reset()

Repositions the stream to the last marked position.



setLineNumber(int)

Sets the current line number.

skip(long)

Skips n bytes of input.

LineNumberInputStream

```
public LineNumberInputStream(InputStream in)
```

Constructs a new LineNumberInputStream initialized with the specified input stream.

Parameters:

in - the input stream

read

```
public int read() throws IOException
```

Reads a byte of data. The method will block if no input is available.

Returns:

the byte read, or -1 if the end of the stream is reached.

Throws: IOException

If an I/O error has occurred.

Overrides:

read in class FilterInputStream

read

```
public int read(byte b[],
                int off,
                int len) throws IOException
```

Reads into an array of bytes. This method will blocks until some input is available.

Parameters:

b - the buffer into which the data is read
off - the start offset of the data
len - the maximum number of bytes read

Returns:

the actual number of bytes read, -1 is returned when the end of the stream is reached.

Throws: IOException

If an I/O error has occurred.

Overrides:

read in class FilterInputStream

setLineNumber

```
public void setLineNumber(int lineNumber)
```

Sets the current line number.

Parameters:

lineNumber - the line number to be set

getLineNumber

```
public int getLineNumber()
```

Returns the current line number.

skip

```
public long skip(long n) throws IOException
```

Skips n bytes of input.

Parameters:

n - the number of bytes to be skipped

Returns:

the actual number of bytes skipped.

Throws: IOException
If an I/O error has occurred.
Overrides:
skip in class FilterInputStream

available

```
public int available() throws IOException
```

Returns the number of bytes that can be read without blocking.
Returns:
the number of available bytes
Overrides:
available in class FilterInputStream

mark

```
public void mark(int readlimit)
```

Marks the current position in the input stream. A subsequent call to reset() will reposition the stream at the last marked position so that subsequent reads will re-read the same bytes. The stream promises to allow readlimit bytes to be read before the mark position gets invalidated.
Parameters:
readlimit - the maximum limit of bytes allowed to be read before the mark position becomes invalid.
Overrides:
mark in class FilterInputStream

reset

```
public void reset() throws IOException
```

Repositions the stream to the last marked position. If the stream has not been marked, or if the mark has been invalidated, an IOException is thrown. Stream marks are intended to be used in situations where you need to read ahead a little to see what's in the stream. Often this is most easily done by invoking some general parser. If the stream is of the type handled by the parser, it just chugs along happily. If the stream is not of that type, the parser should toss an exception when it fails, which, if it happens within readlimit bytes, allows the outer code to reset the stream and try another parser.
Overrides:
reset in class FilterInputStream

Class java.io.OutputStream

Class java.io.OutputStream

```
java.lang.Object
    |
    +----java.io.OutputStream
```

---

public class **OutputStream**
extends Object

Abstract class representing an output stream of bytes. All OutputStreams are based on this class.

See Also:

InputStream, FilterOutputStream, BufferedOutputStream, DataOutputStream , ByteArrayOutputStream

---

OutputStream()

close()

Closes the stream.

flush()

Flushes the stream.

write(int)

Writes a byte.

write(byte[])

Writes an array of bytes.

Interface Index

write(byte[], int, int)

Writes a sub array of bytes.

Interface Index

Interface Index

OutputStream

```
public OutputStream()
```

Interface Index

Interface Index

write

```
public abstract void write(int b) throws IOException
```

Writes a byte. This method will block until the byte is actually written.

Parameters:

b - the byte

Throws: IOException

If an I/O error has occurred.

Interface Index

write

```
public void write(byte b[]) throws IOException
```

Writes an array of bytes. This method will block until the bytes are actually written.

Parameters:

b - the data to be written

Throws: IOException

If an I/O error has occurred.

Interface Index

write

```
public void write(byte b[],
                  int off,
                  int len) throws IOException
```

Writes a sub array of bytes.
Parameters:
b - the data to be written
off - the start offset in the data
len - the number of bytes that are written
Throws: IOException
If an I/O error has occurred.

flush

```
public void flush() throws IOException
```

Flushes the stream. This will write any buffered output bytes.
Throws: IOException
If an I/O error has occurred.

close

```
public void close() throws IOException
```

Closes the stream. This method must be called to release any resources associated with the stream.
Throws: IOException
If an I/O error has occurred.

Class java.io.PipedInputStream

Class java.io.PipedInputStream

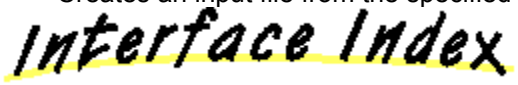
```
java.lang.Object
    |
    +----java.io.InputStream
            |
            +----java.io.PipedInputStream
```

---

public class **PipedInputStream**
extends InputStream

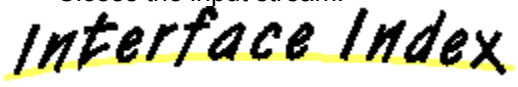
PipedInputStream must be connected to a PipedOutputStream to be useful. A thread reading from a PipedInputStream recieves data from a thread writing to the PipedOutputStream it is connected to.
See Also:
PipedOutputStream

PipedInputStream(PipedOutputStream)
Creates an input file from the specified PiledOutputStream.



PipedInputStream()
Creates an input file that isn't connected to anything (yet).

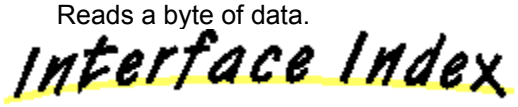
close()
Closes the input stream.



connect(PipedOutputStream)
Connects this input stream to a sender.

read()

Reads a byte of data.



read(byte[], int, int)

Reads into an array of bytes.

PipedInputStream

```
public PipedInputStream(PipedOutputStream src) throws IOException
```

Creates an input file from the specified PiledOutputStream.

Parameters:

src - the stream to connect to.

PipedInputStream

```
public PipedInputStream()
```

Creates an input file that isn't connected to anything (yet). It must be connected to a PipedOutputStream before being used.

connect

```
public void connect(PipedOutputStream src) throws IOException
```

Connects this input stream to a sender.

Parameters:

src - The OutputStream to connect to.

read

```
public synchronized int read() throws IOException
```

Reads a byte of data. This method will block if no input is available.

Returns:

the byte read, or -1 if the end of the stream is reached.

Throws: IOException

If the pipe is broken.

Overrides:

read in class InputStream

read

```
public synchronized int read(byte b[],
                             int off,
                             int len) throws IOException
```

Reads into an array of bytes. Blocks until some input is available.

Parameters:

b - the buffer into which the data is read

off - the start offset of the data

len - the maximum number of bytes read

Returns:

the actual number of bytes read, -1 is returned when the end of the stream is reached.

Throws: IOException

If an I/O error has occurred.

Overrides:

read in class InputStream

close

```
public void close() throws IOException
```

Closes the input stream. Must be called to release any resources associated with the stream.

Throws: IOException

If an I/O error has occurred.

Overrides:

close in class InputStream

Class java.io.PipedOutputStream

Class java.io.PipedOutputStream

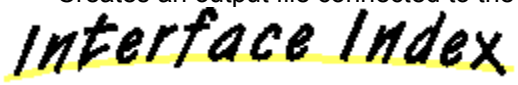
```
java.lang.Object
    |
    +----java.io.OutputStream
             |
             +----java.io.PipedOutputStream
```

---

public class **PipedOutputStream**
extends OutputStream

Piped output stream, must be connected to a PipedInputStream. A thread reading from a PipedInputStream receives data from a thread writing to the PipedOutputStream it is connected to.
See Also:
PipedInputStream

PipedOutputStream(PipedInputStream)
Creates an output file connected to the specified PipedInputStream.



PipedOutputStream()
Creates an output file that isn't connected to anything (yet).

close()
Closes the stream.

connect(PipedInputStream)
Connect this output stream to a receiver.

Interface Index

write(int)

Write a byte.

Interface Index

write(byte[], int, int)

Writes a sub array of bytes.

Interface Index

Interface Index

PipedOutputStream

```
public PipedOutputStream(PipedInputStream snk) throws IOException
```

Creates an output file connected to the specified PipedInputStream.

Parameters:

snk - The InputStream to connect to.

Interface Index

PipedOutputStream

```
public PipedOutputStream()
```

Creates an output file that isn't connected to anything (yet). It must be connected before being used.

Interface Index

Interface Index

connect

```
public void connect(PipedInputStream snk) throws IOException
```

Connect this output stream to a receiver.

Parameters:

snk - The InputStream to connect to.

Interface Index

write

```
public void write(int b) throws IOException
```

Write a byte. This method will block until the byte is actually written.

Parameters:

b - the byte to be written

Throws: IOException

If an I/O error has occurred.

Overrides:

write in class OutputStream

write

```
public void write(byte b[],
                  int off,
                  int len) throws IOException
```

Writes a sub array of bytes.

Parameters:

b - the data to be written

off - the start offset in the data

len - the number of bytes that are written

Throws: IOException

If an I/O error has occurred.

Overrides:

write in class OutputStream

close

```
public void close() throws IOException
```

Closes the stream. This method must be called to release any resources associated with the stream.

Throws: IOException

If an I/O error has occurred.

Overrides:

close in class OutputStream

Class java.io.PrintStream

Class java.io.PrintStream

```
java.lang.Object
    |
    +----java.io.OutputStream
            |
            +----java.io.FilterOutputStream
                    |
                    +----java.io.PrintStream
```

---

public class **PrintStream**
extends FilterOutputStream

This class implements an output stream that has additional methods for printing. You can specify that the stream should be flushed every time a newline character is written.

The top byte of 16 bit characters is discarded.

Example:

```
System.out.println("Hello world!");
System.out.print("x = ");
System.out.println(x);
System.out.println("y = " + y);
```

PrintStream(OutputStream)

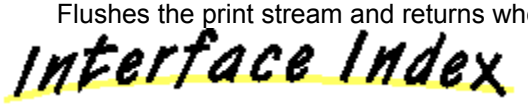
Creates a new PrintStream.

PrintStream(OutputStream, boolean)

Creates a new PrintStream, with auto flushing.

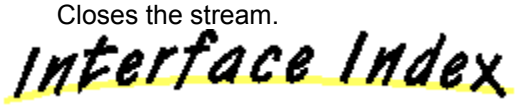
checkError()

Flushes the print stream and returns whether or not there was an error on the output stream.



close()

Closes the stream.



flush()

Flushes the stream.

print(Object)

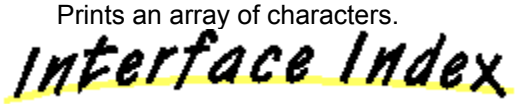
Prints an object.

print(String)

Prints a String.

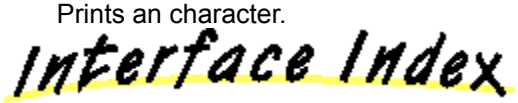
print(char[])

Prints an array of characters.



print(char)

Prints an character.



print(int)

Prints an integer.

print(long)

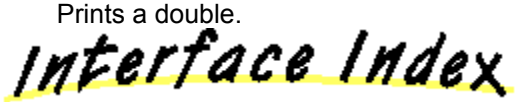
Prints a long.

print(float)

Prints a float.

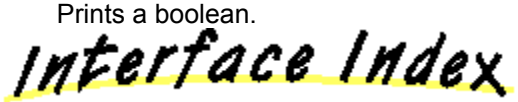
print(double)

Prints a double.



print(boolean)

Prints a boolean.



println()

Prints a newline.

println(Object)

Prints an object followed by a newline.

println(String)

Prints a string followed by a newline.

Interface Index **println**(char[])

Prints an array of characters followed by a newline.

Interface Index **println**(char)

Prints a character followed by a newline.

Interface Index **println**(int)

Prints an integer followed by a newline.

Interface Index **println**(long)

Prints a long followed by a newline.

Interface Index **println**(float)

Prints a float followed by a newline.

Interface Index **println**(double)

Prints a double followed by a newline.

Interface Index **println**(boolean)

Prints a boolean followed by a newline.

Interface Index **write**(int)

Writes a byte.

Interface Index **write**(byte[], int, int)

Writes a sub array of bytes.

Interface Index

Interface Index **PrintStream**

```
public PrintStream(OutputStream out)
```

Creates a new PrintStream.

Parameters:

out - the output stream

Interface Index **PrintStream**

```
public PrintStream(OutputStream out,
```

```
                    boolean autoflush)
```

Creates a new PrintStream, with auto flushing.

Parameters:

out - the output stream

autoflush - if true the stream automatically flushes its output when a newline character is printed

Interface Index

Interface Index

write

```
public void write(int b)
```

Writes a byte. This method will block until the byte is actually written.

Parameters:

b - the byte

Throws: IOException

If an I/O error has occurred.

Overrides:

write in class FilterOutputStream

Interface Index

write

```
public void write(byte b[],
                  int off,
                  int len)
```

Writes a sub array of bytes.

Parameters:

b - the data to be written

off - the start offset in the data

len - the number of bytes that are written

Throws: IOException

If an I/O error has occurred.

Overrides:

write in class FilterOutputStream

Interface Index

flush

```
public void flush()
```

Flushes the stream. This will write any buffered output bytes.
Overrides:
flush in class FilterOutputStream

close

```
public void close()
```

Closes the stream.
Overrides:
close in class FilterOutputStream

checkError

```
public boolean checkError()
```

Flushes the print stream and returns whether or not there was an error on the output stream. Errors are cumulative; once the print stream encounters an error this routine will continue to return true on all successive calls.
Returns:
true if the print stream has ever encountered an error on the output stream.

print

```
public void print(Object obj)
```

Prints an object.
Parameters:
obj - the object to be printed

print

```
public synchronized void print(String s)
```

Prints a String.
Parameters:
s - the String to be printed

print

```
public synchronized void print(char s[])
```

Prints an array of characters.
Parameters:
s - the array of chars to be printed

print

```
public void print(char c)
```

Prints an character.
Parameters:
c - the character to be printed

print

```
public void print(int i)
```

Prints an integer.
Parameters:
i - the integer to be printed

print

```
public void print(long l)
```

Prints a long.
Parameters:
l - the long to be printed.

print

```
public void print(float f)
```

Prints a float.
Parameters:
f - the float to be printed

print

```
public void print(double d)
```

Prints a double.
Parameters:
d - the double to be printed

print

```
public void print(boolean b)
```

Prints a boolean.
Parameters:
b - the boolean to be printed

println

```
public void println()
```

Prints a newline.

println

```
public synchronized void println(Object obj)
```

Prints an object followed by a newline.

Parameters:
obj - the object to be printed

println

```
public synchronized void println(String s)
```

Prints a string followed by a newline.
Parameters:
s - the String to be printed

println

```
public synchronized void println(char s[])
```

Prints an array of characters followed by a newline.
Parameters:
s - the array of characters to be printed

println

```
public synchronized void println(char c)
```

Prints a character followed by a newline.
Parameters:
c - the character to be printed

println

```
public synchronized void println(int i)
```

Prints an integer followed by a newline.
Parameters:
i - the integer to be printed

```
public synchronized void println(long l)
```

Prints a long followed by a newline.
Parameters:
l - the long to be printed


```
public synchronized void println(float f)
```

Prints a float followed by a newline.
Parameters:
f - the float to be printed


```
public synchronized void println(double d)
```

Prints a double followed by a newline.
Parameters:
d - the double to be printed


```
public synchronized void println(boolean b)
```

Prints a boolean followed by a newline.
Parameters:
b - the boolean to be printed

Class java.io.PushbackInputStream

Class java.io.PushbackInputStream

```
java.lang.Object
    |
    +----java.io.InputStream
            |
            +----java.io.FilterInputStream
                    |
                    +----java.io.PushbackInputStream
```

---

public class **PushbackInputStream**
extends FilterInputStream

An input stream that has a 1 byte push back buffer.

pushBack

Push back character.

PushbackInputStream(InputStream)

Creates a PushbackInputStream.

available()

Returns the number of bytes that can be read.

Interface Index **markSupported**()
Returns true if this stream type supports mark/reset.
Interface Index **read**()
Reads a byte of data.
Interface Index **read**(byte[], int, int)
Reads into an array of bytes.
Interface Index **unread**(int)
Pushes back a character.

Interface Index
Interface Index **pushBack**

```
protected int pushBack
```

Push back character.

Interface Index
Interface Index **PushbackInputStream**

```
public PushbackInputStream(InputStream in)
```

Creates a PushbackInputStream.
Parameters:
in - the input stream

Interface Index
Interface Index **read**

```
public int read() throws IOException
```

Reads a byte of data. This method will block if no input is available.

Returns:

the byte read, or -1 if the end of the stream is reached.

Throws: IOException

If an I/O error has occurred.

Overrides:

read in class FilterInputStream

read

```
public int read(byte bytes[],
                int offset,
                int length) throws IOException
```

Reads into an array of bytes. This method blocks until some input is available.

Parameters:

b - the buffer into which the data is read

off - the start offset of the data

len - the maximum number of bytes read

Returns:

the actual number of bytes read, -1 is returned when the end of the stream is reached.

Throws: IOException

If an I/O error has occurred.

Overrides:

read in class FilterInputStream

unread

```
public void unread(int ch) throws IOException
```

Pushes back a character.

Parameters:

ch - the character to push back.

Throws: IOException

If an attempt to push back more than one character is made.

available

```
public int available() throws IOException
```

Returns the number of bytes that can be read. without blocking.
Overrides:
available in class FilterInputStream

markSupported

```
public boolean markSupported()
```

Returns true if this stream type supports mark/reset.
Overrides:
markSupported in class FilterInputStream

Class java.io.RandomAccessFile

Class java.io.RandomAccessFile

```
java.lang.Object
    |
    +----java.io.RandomAccessFile
```

---

public class **RandomAccessFile**
extends Object
implements DataOutput, DataInput

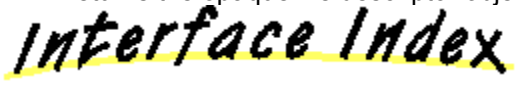
Random access files can be constructed from file descriptors, file names, or file objects. This class provides a sense of security by offering methods that allow specified mode accesses of read-only or read-write to files.

RandomAccessFile(String, String)
Creates a RandomAccessFile with the specified system dependent file name and the specified mode.

RandomAccessFile(File, String)
Creates a RandomAccessFile from a specified File object and mode ("r" or "rw").

close()
Closes the file.

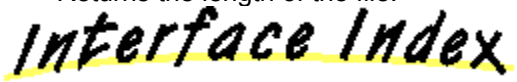
getFD()
Returns the opaque file descriptor object.



getFilePointer()
Returns the current location of the file pointer.

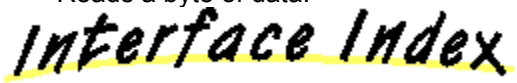
length()

Returns the length of the file.



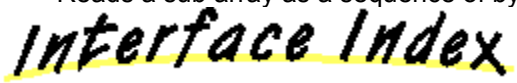
read()

Reads a byte of data.



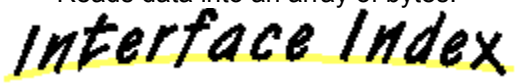
read(byte[], int, int)

Reads a sub array as a sequence of bytes.



read(byte[])

Reads data into an array of bytes.



readBoolean()

Reads a boolean.

readByte()

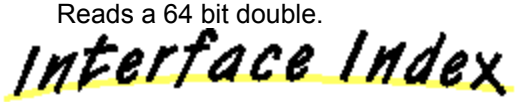
Reads a byte.

readChar()

Reads a 16 bit char.

readDouble()

Reads a 64 bit double.

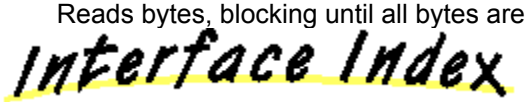


readFloat()

Reads a 32 bit float.

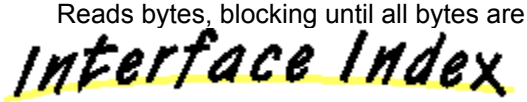
readFully(byte[])

Reads bytes, blocking until all bytes are read.



readFully(byte[], int, int)

Reads bytes, blocking until all bytes are read.

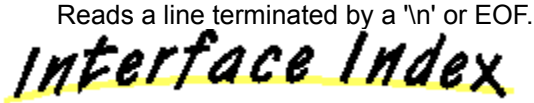


readInt()

Reads a 32 bit int.

readLine()

Reads a line terminated by a '\n' or EOF.



readLong()

Reads a 64 bit long.

readShort()

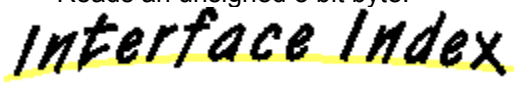
Reads 16 bit short.

readUTF()

Reads a UTF formatted String.

readUnsignedByte()

Reads an unsigned 8 bit byte.

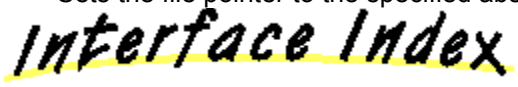


readUnsignedShort()

Reads 16 bit short.

seek(long)

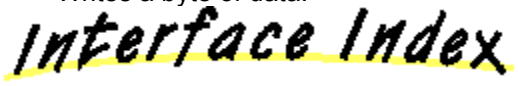
Sets the file pointer to the specified absolute position.



skipBytes(int)

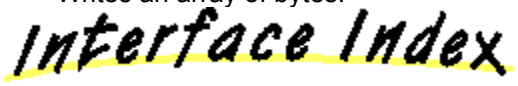
write(int)

Writes a byte of data.



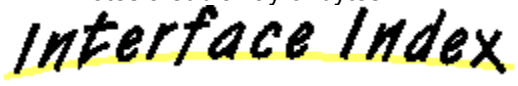
write(byte[])

Writes an array of bytes.



write(byte[], int, int)

Wrotes a sub array of bytes.



writeBoolean(boolean)

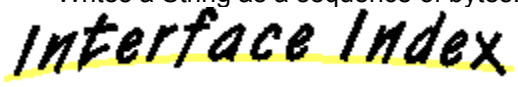
Writes a boolean.

writeByte(int)

Writes a byte.

writeBytes(String)

Writes a String as a sequence of bytes.

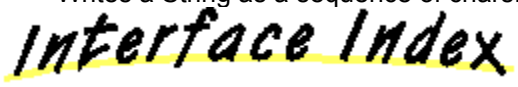


writeChar(int)

Writes a character.

writeChars(String)

Writes a String as a sequence of chars.



writeDouble(double)

writeFloat(float)

writeInt(int)

Writes an integer.

writeLong(long)

Writes a long.

writeShort(int)

Writes a short.

writeUTF(String)

Writes a String in UTF format.

RandomAccessFile

```
public RandomAccessFile(String name,
                        String mode) throws IOException
```

Creates a RandomAccessFile with the specified system dependent file name and the specified mode. Mode "r" is for read-only and mode "rw" is for read+write.

Parameters:

name - the system dependent file name

mode - the access mode

Throws: IOException

If an I/O error has occurred.

RandomAccessFile

```
public RandomAccessFile(File file,
                        String mode) throws IOException
```

Creates a RandomAccessFile from a specified File object and mode ("r" or "rw").

Parameters:

file - the file object

mode - the access mode

getFD

```
public final FileDescriptor getFD() throws IOException
```

Returns the opaque file descriptor object.

Returns:

the file descriptor.

read

```
public int read() throws IOException
```

Reads a byte of data. This method will block if no input is available.

Returns:

the byte read, or -1 if the end of the stream is reached.

Throws: IOException

If an I/O error has occurred.

read

```
public int read(byte b[],
                int off,
                int len) throws IOException
```

Reads a sub array as a sequence of bytes.

Parameters:

b - the data to be written

off - the start offset in the data

len - the number of bytes that are written

Throws: IOException

If an I/O error has occurred.

read

```
public int read(byte b[]) throws IOException
```

Reads data into an array of bytes. This method blocks until some input is available.

Returns:

the actual number of bytes read, -1 is returned when the end of the stream is reached.

Throws: IOException

If an I/O error has occurred.

 readFully

```
public final void readFully(byte b[]) throws IOException
```

Reads bytes, blocking until all bytes are read.

Parameters:

b - the buffer into which the data is read

Returns:

the actual number of bytes read, -1 is returned when the end of the stream is reached.

Throws: IOException

If an I/O error has occurred.

 readFully

```
public final void readFully(byte b[],
                            int off,
                            int len) throws IOException
```

Reads bytes, blocking until all bytes are read.

Parameters:

b - the buffer into which the data is read

off - the start offset of the data

len - the maximum number of bytes read

Returns:

the actual number of bytes read, -1 is returned when the end of the stream is reached.

Throws: IOException

If an I/O error has occurred.

 skipBytes

```
public int skipBytes(int n) throws IOException
```

 write

```
public void write(int b) throws IOException
```

Writes a byte of data. This method will block until the byte is actually written.

Parameters:

b - the byte to be written

Throws: IOException
If an I/O error has occurred.

write

```
public void write(byte b[]) throws IOException
```

Writes an array of bytes. Will block until the bytes are actually written.
Parameters:
b - the data to be written
Throws: IOException
If an I/O error has occurred.

write

```
public void write(byte b[],
                  int off,
                  int len) throws IOException
```

Wrotes a sub array of bytes.
Parameters:
b - the data to be written
off - the start offset in the data
len - the number of bytes that are written
Throws: IOException
If an I/O error has occurred.

getFilePointer

```
public long getFilePointer() throws IOException
```

Returns the current location of the file pointer.

seek

```
public void seek(long pos) throws IOException
```

Sets the file pointer to the specified absolute position.
Parameters:

pos - the absolute position

length

```
public long length() throws IOException
```

Returns the length of the file.

close

```
public void close() throws IOException
```

Closes the file.
Throws: IOException
If an I/O error has occurred.

readBoolean

```
public final boolean readBoolean() throws IOException
```

Reads a boolean.

readByte

```
public final byte readByte() throws IOException
```

Reads a byte.

readUnsignedByte

```
public final int readUnsignedByte() throws IOException
```

Reads an unsigned 8 bit byte.
Returns:
the 8 bit byte read.

readShort

```
public final short readShort() throws IOException
```

Reads 16 bit short.

Returns:

the read 16 bit short.

readUnsignedShort

```
public final int readUnsignedShort() throws IOException
```

Reads 16 bit short.

Returns:

the read 16 bit short.

readChar

```
public final char readChar() throws IOException
```

Reads a 16 bit char.

Returns:

the read 16 bit char.

readInt

```
public final int readInt() throws IOException
```

Reads a 32 bit int.

Returns:

the read 32 bit integer.

readLong

```
public final long readLong() throws IOException
```

Reads a 64 bit long.
Returns:
the read 64 bit long.

readFloat

```
public final float readFloat() throws IOException
```

Reads a 32 bit float.
Returns:
the read 32 bit float.

readDouble

```
public final double readDouble() throws IOException
```

Reads a 64 bit double.
Returns:
the read 64 bit double.

readLine

```
public final String readLine() throws IOException
```

Reads a line terminated by a '\n' or EOF.

readUTF

```
public final String readUTF() throws IOException
```

Reads a UTF formatted String.

writeBoolean

```
public final void writeBoolean(boolean v) throws IOException
```

Writes a boolean.
Parameters:
v - the boolean value

writeByte

```
public final void writeByte(int v) throws IOException
```

Writes a byte.
Parameters:
v - the byte

writeShort

```
public final void writeShort(int v) throws IOException
```

Writes a short.
Parameters:
v - the short

writeChar

```
public final void writeChar(int v) throws IOException
```

Writes a character.
Parameters:
v - the char

writeInt

```
public final void writeInt(int v) throws IOException
```

Writes an integer.
Parameters:
v - the integer

Interface Index **writeLong**

```
public final void writeLong(long v) throws IOException
```

Writes a long.
Parameters:
v - the long

Interface Index **writeFloat**

```
public final void writeFloat(float v) throws IOException
```

Interface Index **writeDouble**

```
public final void writeDouble(double v) throws IOException
```

Interface Index **writeBytes**

```
public final void writeBytes(String s) throws IOException
```

Writes a String as a sequence of bytes.
Parameters:
s - the String

Interface Index **writeChars**

```
public final void writeChars(String s) throws IOException
```

Writes a String as a sequence of chars.
Parameters:
s - the String

Interface Index **writeUTF**

```
public final void writeUTF(String str) throws IOException
```

Writes a String in UTF format.

Parameters:

str - the String

Class java.io.SequenceInputStream

Class java.io.SequenceInputStream

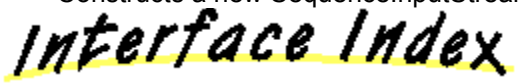
```
java.lang.Object
    |
    +----java.io.InputStream
             |
             +----java.io.SequenceInputStream
```

---

public class **SequenceInputStream**
extends InputStream

Converts a sequence of input streams into an InputStream.

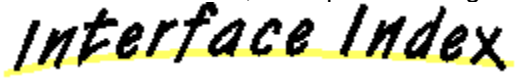
SequenceInputStream(Enumeration)
Constructs a new SequenceInputStream initialized to the specified list.



SequenceInputStream(InputStream, InputStream)
Constructs a new SequenceInputStream initialized to the two specified input streams.

close()
Closes the input stream; flipping to the next stream, if an EOF is reached.

read()
Reads a stream, and upon reaching an EOF, flips to the next stream.



read(byte[], int, int)
Reads data into an array of bytes, and upon reaching an EOF, flips to the next stream.

Interface Index

Interface Index **SequenceInputStream**

```
public SequenceInputStream(Enumeration e)
```

Constructs a new SequenceInputStream initialized to the specified list.
Parameters:
e - the list

Interface Index **SequenceInputStream**

```
public SequenceInputStream(InputStream s1,
                           InputStream s2)
```

Constructs a new SequenceInputStream initialized to the two specified input streams.
Parameters:
s1 - the first input stream
s2 - the second input stream

Interface Index

Interface Index **read**

```
public int read() throws IOException
```

Reads a stream, and upon reaching an EOF, flips to the next stream.
Overrides:
read in class InputStream

Interface Index **read**

```
public int read(byte buf[],
                int pos,
                int len) throws IOException
```

Reads data into an array of bytes, and upon reaching an EOF, flips to the next stream.

Parameters:

buf - the buffer into which the data is read
pos - the start position of the data
len - the maximum number of bytes read

Throws: IOException

If an I/O error has occurred.

Overrides:

read in class InputStream

close

```
public void close() throws IOException
```

Closes the input stream; flipping to the next stream, if an EOF is reached. This method must be called to release any resources associated with the stream.

Throws: IOException

If an I/O error has occurred.

Overrides:

close in class InputStream

Class java.io.StreamTokenizer

Class java.io.StreamTokenizer

```
java.lang.Object
    |
    +----java.io.StreamTokenizer
```

---

public class **StreamTokenizer**
extends Object

A class to turn an input stream into a stream of tokens. There are a number of methods that define the lexical syntax of tokens.


The End-of-file token.

TT_EOF

The End-of-line token.

TT_EOL


The number token.

TT_NUMBER


The word token.

TT_WORD


The number value.

nval


The Stream value.

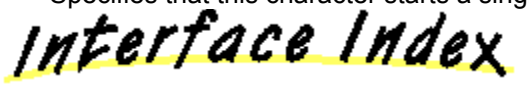
sval


The type of the last token returned.

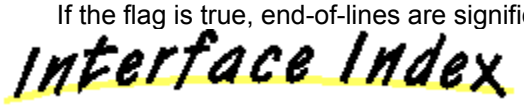
ttype

StreamTokenizer(InputStream)
Creates a stream tokenizer that parses the specified input stream.

commentChar(int)
Specifies that this character starts a single line comment.



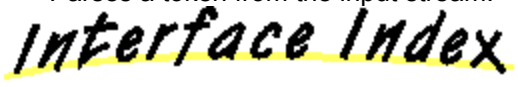
eolIsSignificant(boolean)
If the flag is true, end-of-lines are significant (TT_EOL will be returned by nexttoken).



lineno()
Return the current line number.

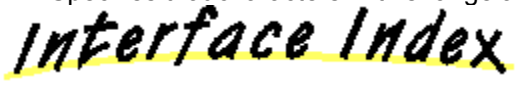
lowerCaseMode(boolean)
Examines a boolean to decide whether TT_WORD tokens are forced to be lower case.

nextToken()
Parses a token from the input stream.

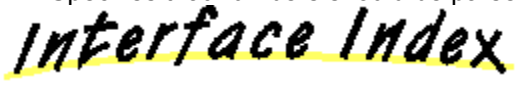


ordinaryChar(int)
Specifies that this character is 'ordinary': it removes any significance as a word, comment, string, whitespace or number character.

ordinaryChars(int, int)
Specifies that characters in this range are 'ordinary'.



parseNumbers()
Specifies that numbers should be parsed.

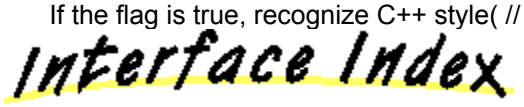


pushBack()
Pushes back a stream token.

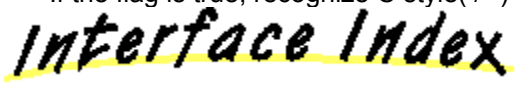
quoteChar(int)
Specifies that matching pairs of this character delimit String constants.

resetSyntax()
Resets the syntax table so that all characters are special.

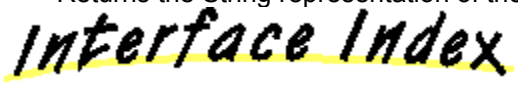
slashSlashComments(boolean)
If the flag is true, recognize C++ style(//) comments.



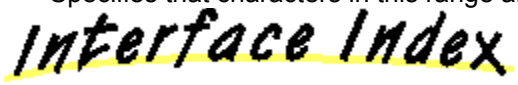
slashStarComments(boolean)
If the flag is true, recognize C style(/*) comments.



toString()
Returns the String representation of the stream token.



whitespaceChars(int, int)
Specifies that characters in this range are whitespace characters.



wordChars(int, int)
Specifies that characters in this range are word characters.

ttype

```
public int ttype
```

The type of the last token returned. It's value will either be one of the following TT_* constants, or a single character. For example, if '+' is encountered and is not a valid word character, ttype will be '+'

TT_EOF

```
public final static int TT_EOF
```

The End-of-file token.

TT_EOL

```
public final static int TT_EOL
```

The End-of-line token.

TT_NUMBER

```
public final static int TT_NUMBER
```

The number token. This value is in nval.

Interface Index TT_WORD

```
public final static int TT_WORD
```

The word token. This value is in sval.

Interface Index sval

```
public String sval
```

The Stream value.

Interface Index nval

```
public double nval
```

The number value.

Interface Index

Interface Index StreamTokenizer

```
public StreamTokenizer(InputStream I)
```

Creates a stream tokenizer that parses the specified input stream. By default, it recognizes numbers, Strings quoted with single and double quotes, and all the alphabetics.
Parameters:
I - the input stream

Interface Index

Interface Index resetSyntax

```
public void resetSyntax()
```

Resets the syntax table so that all characters are special.

wordChars

```
public void wordChars(int low,
                      int hi)
```

Specifies that characters in this range are word characters.
Parameters:
low - the low end of the range
hi - the high end of the range

whitespaceChars

```
public void whitespaceChars(int low,
                            int hi)
```

Specifies that characters in this range are whitespace characters.
Parameters:
low - the low end of the range
hi - the high end of the range

ordinaryChars

```
public void ordinaryChars(int low,
                          int hi)
```

Specifies that characters in this range are 'ordinary'. Ordinary characters mean that any significance as words, comments, strings, whitespaces or number characters are removed. When these characters are encountered by the parser, they return a ttype equal to the character.
Parameters:
low - the low end of the range
hi - the high end of the range

ordinaryChar

```
public void ordinaryChar(int ch)
```

Specifies that this character is 'ordinary': it removes any significance as a word, comment, string, whitespace or number character. When encountered by the parser, it returns a ttype equal to the character.

Parameters:

ch - the character

commentChar

```
public void commentChar(int ch)
```

Specifies that this character starts a single line comment.

Parameters:

ch - the character

quoteChar

```
public void quoteChar(int ch)
```

Specifies that matching pairs of this character delimit String constants. When a String constant is recognized, ttype will be the character that delimits the String, and sval will have the body of the String.

Parameters:

ch - the character

parseNumbers

```
public void parseNumbers()
```

Specifies that numbers should be parsed. This method accepts double precision floating point numbers and returns a ttype of TT_NUMBER with the value in nval.

eolIsSignificant

```
public void eolIsSignificant(boolean flag)
```

If the flag is true, end-of-lines are significant (TT_EOL will be returned by nexttoken). If false, they will

be treated as whitespace.

slashStarComments

```
public void slashStarComments(boolean flag)
```

If the flag is true, recognize C style(/*) comments.

slashSlashComments

```
public void slashSlashComments(boolean flag)
```

If the flag is true, recognize C++ style(//) comments.

lowerCaseMode

```
public void lowerCaseMode(boolean fl)
```

Examines a boolean to decide whether TT_WORD tokens are forced to be lower case.
Parameters:
fl - the boolean flag

nextToken

```
public int nextToken() throws IOException
```

Parses a token from the input stream. The return value is the same as the value of ttype. Typical clients of this class first set up the syntax tables and then sit in a loop calling nextToken to parse successive tokens until TT_EOF is returned.

pushBack

```
public void pushBack()
```

Pushes back a stream token.

lineno

```
public int lineno()
```

Return the current line number.

toString

```
public String toString()
```

Returns the String representation of the stream token.

Overrides:

toString in class Object

Class java.io.StringBufferInputStream

Class java.io.StringBufferInputStream

```
java.lang.Object
    |
    +----java.io.InputStream
             |
             +----java.io.StringBufferInputStream
```

---

public class **StringBufferInputStream**
extends InputStream

This class implements a String buffer that can be used as an InputStream.

buffer

The buffer where data is stored.

count

The number of characters to use in the buffer.

pos

The position in the buffer.

StringBufferInputStream(String)

Creates an StringBufferInputStream from the specified array of bytes.

Interface Index **available**()

Returns the number of available bytes in the buffer.

Interface Index **read**()

Reads a byte of data.

Interface Index **read**(byte[], int, int)

Reads into an array of bytes.

Interface Index **reset**()

Resets the buffer to the beginning.

Interface Index **skip**(long)

Skips n bytes of input.

Interface Index

Interface Index **buffer**

```
protected String buffer
```

The buffer where data is stored.

Interface Index **pos**

```
protected int pos
```

The position in the buffer.

Interface Index **count**

```
protected int count
```

The number of characters to use in the buffer.

Interface Index

StringBufferInputStream

```
public StringBufferInputStream(String s)
```

Creates an StringBufferInputStream from the specified array of bytes.
Parameters:
s - the input buffer (not copied)

read

```
public synchronized int read()
```

Reads a byte of data.
Returns:
the byte read, or -1 if the end of the stream is reached.
Overrides:
read in class InputStream

read

```
public synchronized int read(byte b[],
                             int off,
                             int len)
```

Reads into an array of bytes.
Parameters:
b - the buffer into which the data is read
off - the start offset of the data
len - the maximum number of bytes read
Returns:
the actual number of bytes read; -1 is returned when the end of the stream is reached.
Overrides:
read in class InputStream

skip

```
public synchronized long skip(long n)
```

Skips n bytes of input.
Parameters:
n - the number of bytes to be skipped
Returns:
the actual number of bytes skipped.
Overrides:
skip in class InputStream

 available

```
public synchronized int available()
```

Returns the number of available bytes in the buffer.
Overrides:
available in class InputStream

 reset

```
public synchronized void reset()
```

Resets the buffer to the beginning.
Overrides:
reset in class InputStream

Class java.io.UTFDataFormatException

Class java.io.UTFDataFormatException

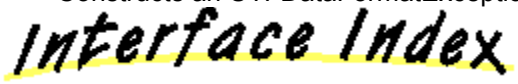
```
java.lang.Object
    |
    +----java.lang.Throwable
            |
            +----java.lang.Exception
                    |
                    +----java.io.IOException
                            |
                            +----java.io.UTFDataFormatException
```

---

public class **UTFDataFormatException**
extends IOException

Signals that a malformed UTF-8 string has been read in a DataInput stream.
See Also:
IOException, DataInput

UTFDataFormatException()
Constructs an UTFDataFormatException with no detail message.



UTFDataFormatException(String)
Constructs an UTFDataFormatException with the specified detail message.

UTFDataFormatException

```
public UTFDataFormatException()
```

Constructs an UTFDataFormatException with no detail message. A detail message is a String that describes this particular exception.

UTFDataFormatException

```
public UTFDataFormatException(String s)
```

Constructs an UTFDataFormatException with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class java.lang.AbstractMethodError

Class java.lang.AbstractMethodError

```
java.lang.Object
    |
    +----java.lang.Throwable
            |
            +----java.lang.Error
                    |
                    +----java.lang.LinkageError
                            |
                            +----java.lang.IncompatibleClassChangeError
                                    |
                                    +----java.lang.AbstractMethodError
```

---

public class **AbstractMethodError**
extends IncompatibleClassChangeError

Signals an attempt to call an abstract method.

AbstractMethodError()
Constructs an AbstractMethodError with no detail message.

AbstractMethodError(String)
Constructs an AbstractMethodError with the specified detail message.

AbstractMethodError

```
public AbstractMethodError()
```

Constructs an AbstractMethodError with no detail message. A detail message is a String that describes this particular exception.


```
public AbstractMethodError(String s)
```

Constructs an AbstractMethodError with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the String that contains the detail message

Class java.lang.ArithmeticException

Class java.lang.ArithmeticException

```
java.lang.Object
    |
    +----java.lang.Throwable
            |
            +----java.lang.Exception
                    |
                    +----java.lang.RuntimeException
                            |
                            +----java.lang.ArithmeticException
```

---

public class **ArithmeticException**
extends RuntimeException

Signals that an exceptional arithmetic condition has occurred. For example, dividing by zero would invoke this class.

Interface Index

Interface Index **ArithmeticException**

```
public ArithmeticException()
```

Constructs an ArithmeticException with no detail message. A detail message is a String that

describes this particular exception.

ArithmeticException

```
public ArithmeticException(String s)
```

Constructs an ArithmeticException with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the String that contains a detailed message

Class java.lang.ArrayIndexOutOfBoundsException

Class java.lang.ArrayIndexOutOfBoundsException

```
java.lang.Object
    |
    +----java.lang.Throwable
             |
             +----java.lang.Exception
                      |
                      +----java.lang.RuntimeException
                               |
                               +----java.lang.IndexOutOfBoundsException
                                        |
                                        +----
java.lang.ArrayIndexOutOfBoundsException
```

---

public class **ArrayIndexOutOfBoundsException**
extends IndexOutOfBoundsException

Signals that an invalid array index has been used.

ArrayIndexOutOfBoundsException()
Constructs an ArrayIndexOutOfBoundsException with no detail message.

ArrayIndexOutOfBoundsException(int)
Constructs a new ArrayIndexOutOfBoundsException class initialized to the specific index.

ArrayIndexOutOfBoundsException(String)
Constructs an ArrayIndexOutOfBoundsException class with the specified detail message.

ArrayIndexOutOfBoundsException

```
public ArrayIndexOutOfBoundsException()
```

Constructs an ArrayIndexOutOfBoundsException with no detail message. A detail message is a String that describes this particular exception.

ArrayIndexOutOfBoundsException

```
public ArrayIndexOutOfBoundsException(int index)
```

Constructs a new ArrayIndexOutOfBoundsException class initialized to the specific index.

Parameters:

index - the index where the error occurred

ArrayIndexOutOfBoundsException

```
public ArrayIndexOutOfBoundsException(String s)
```

Constructs an ArrayIndexOutOfBoundsException class with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the String containing a detail message

Class java.lang.ArrayStoreException

Class java.lang.ArrayStoreException

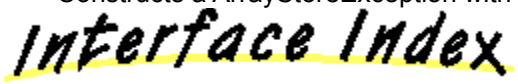
```
java.lang.Object
    |
    +----java.lang.Throwable
            |
            +----java.lang.Exception
                    |
                    +----java.lang.RuntimeException
                            |
                            +----java.lang.ArrayStoreException
```

---

public class **ArrayStoreException**
extends RuntimeException

An attempt has been made to store the wrong type of Object to an array.

ArrayStoreException()
Constructs a ArrayStoreException with no detail message.



ArrayStoreException(String)
Constructs a ArrayStoreException with the specified detail message.

ArrayStoreException

```
public ArrayStoreException()
```

Constructs a ArrayStoreException with no detail message. A detail message is a String that describes this particular exception.

ArrayStoreException

```
public ArrayStoreException(String s)
```

Constructs a ArrayStoreException with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the String containing a detail message

Class java.lang.Boolean

Class java.lang.Boolean

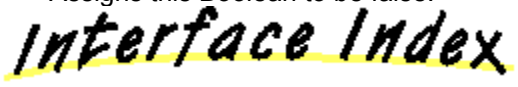
```
java.lang.Object
    |
    +----java.lang.Boolean
```

---

public final class **Boolean**
extends Object

The Boolean class provides an object wrapper for Boolean data values, and serves as a place for boolean-oriented operations. A wrapper is useful because most of Java's utility classes require the use of objects. Since booleans are not objects in Java, they need to be "wrapped" in a Boolean instance.

FALSE
Assigns this Boolean to be false.



MAX_VALUE
The maximum value a Character can have.

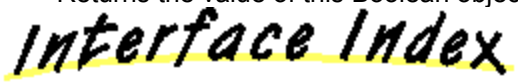
MIN_VALUE
The minimum value a Charater can have.

TRUE
Assigns this Boolean to be true.

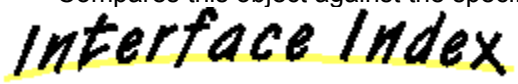
Boolean(boolean)
Constructs a Boolean object initialized to the specified boolean value.

Boolean(String)
Constructs a Boolean object initialized to the value specified by the String parameter.

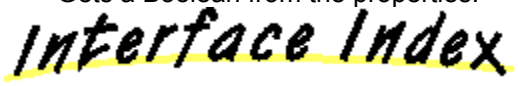
booleanValue()
Returns the value of this Boolean object as a boolean.



equals(Object)
Compares this object against the specified object.

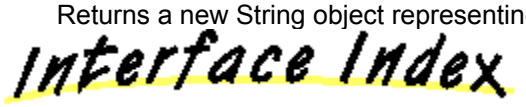


getBoolean(String)
Gets a Boolean from the properties.



hashCode()
Returns a hashcode for this Boolean.

toString()
Returns a new String object representing this Boolean's value.



valueOf(String)
Returns the boolean value represented by the specified String.

TRUE

```
public final static Boolean TRUE
```

Assigns this Boolean to be true.

FALSE

```
public final static Boolean FALSE
```

Assigns this Boolean to be false.

Interface Index **MIN_VALUE**

```
public final static char MIN_VALUE
```

The minimum value a Character can have. The lowest minimum value an Integer can have is .

Interface Index **MAX_VALUE**

```
public final static char MAX_VALUE
```

The maximum value a Character can have. The greatest maximum value an Integer can have is .

Interface Index

Interface Index **Boolean**

```
public Boolean(boolean value)
```

Constructs a Boolean object initialized to the specified boolean value.
Parameters:
value - the value of the boolean

Interface Index **Boolean**

```
public Boolean(String s)
```

Constructs a Boolean object initialized to the value specified by the String parameter.
Parameters:
s - the String to be converted to a Boolean

Interface Index

Interface Index **booleanValue**

```
public boolean booleanValue()
```

Returns the value of this Boolean object as a boolean.

valueOf

```
public static Boolean valueOf(String s)
```

Returns the boolean value represented by the specified String.

Parameters:

s - the String to be parsed

toString

```
public String toString()
```

Returns a new String object representing this Boolean's value.

Overrides:

toString in class Object

hashCode

```
public int hashCode()
```

Returns a hashcode for this Boolean.

Overrides:

hashCode in class Object

equals

```
public boolean equals(Object obj)
```

Compares this object against the specified object.

Parameters:

obj - the object to compare with

Returns:

true if the objects are the same; false otherwise.

Overrides:

equals in class Object

getBoolean

```
public static boolean getBoolean(String name)
```

Gets a Boolean from the properties.

Parameters:

name - the property name.

Class java.lang.Character

Class java.lang.Character

```
java.lang.Object
    |
    +----java.lang.Character
```

---

public final class **Character**
extends Object

The Character class provides an object wrapper for Character data values and serves as a place for character-oriented operations. A wrapper is useful because most of Java's utility classes require the use of objects. Since characters are not objects in Java, they need to be "wrapped" in a Character instance.

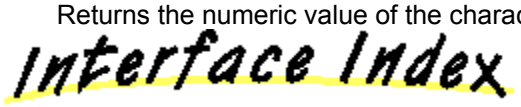
MAX_RADIX
The maximum radix available for conversion to and from Strings.

MIN_RADIX
The minimum radix available for conversion to and from Strings.

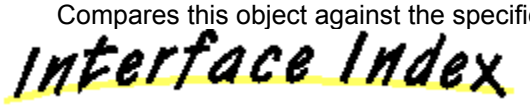
Character(char)
Constructs a Character object with the specified value.

charValue()
Returns the value of this Character object.

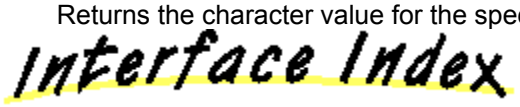
digit(char, int)
Returns the numeric value of the character digit using the specified radix.



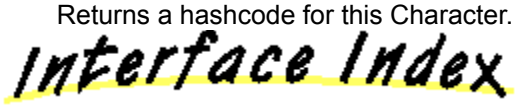
equals(Object)
Compares this object against the specified object.



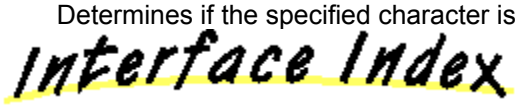
forDigit(int, int)
Returns the character value for the specified digit in the specified radix.



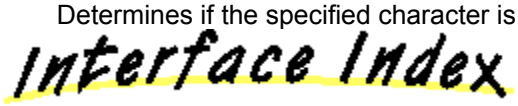
hashCode()
Returns a hashcode for this Character.



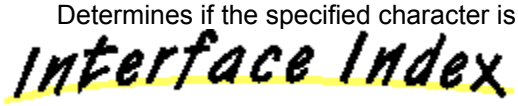
isDigit(char)
Determines if the specified character is a ISO-LATIN-1 digit.



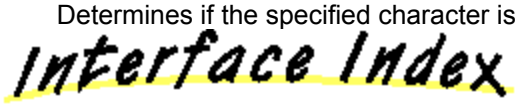
isLowerCase(char)
Determines if the specified character is ISO-LATIN-1 lower case.



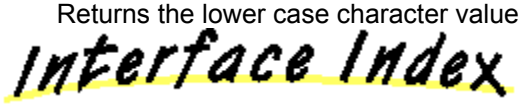
isSpace(char)
Determines if the specified character is ISO-LATIN-1 white space according to Java.



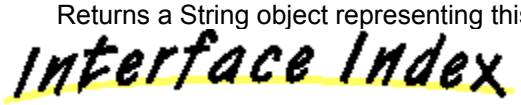
isUpperCase(char)
Determines if the specified character is ISO-LATIN-1 upper case.



toLowerCase(char)
Returns the lower case character value of the specified ISO-LATIN-1 character.



toString()
Returns a String object representing this character's value.



toUpperCase(char)
Returns the upper case character value of the specified ISO-LATIN-1 character.

MIN_RADIX

```
public final static int MIN_RADIX
```

The minimum radix available for conversion to and from Strings. The lowest minimum value that a radix can be is 2.
See Also:

toString

Interface Index **MAX_RADIX**

```
public final static int MAX_RADIX
```

The maximum radix available for conversion to and from Strings. The largest maximum value that a radix can have is 36.

See Also:

toString

Interface Index

Interface Index **Character**

```
public Character(char value)
```

Constructs a Character object with the specified value.

Parameters:

value - value of this Character object

Interface Index

Interface Index **isLowerCase**

```
public static boolean isLowerCase(char ch)
```

Determines if the specified character is ISO-LATIN-1 lower case.

Parameters:

ch - the character to be tested

Returns:

true if the character is lower case; false otherwise.

Interface Index **isUpperCase**

```
public static boolean isUpperCase(char ch)
```

Determines if the specified character is ISO-LATIN-1 upper case.

Parameters:

ch - the character to be tested

Returns:

true if the character is upper case; false otherwise.

Interface Index **isDigit**

```
public static boolean isDigit(char ch)
```

Determines if the specified character is a ISO-LATIN-1 digit.

Parameters:

ch - the character to be tested

Returns:

true if this character is a digit; false otherwise.

Interface Index **isSpace**

```
public static boolean isSpace(char ch)
```

Determines if the specified character is ISO-LATIN-1 white space according to Java.

Parameters:

ch - the character to be tested

Returns:

true if the character is white space; false otherwise.

Interface Index **toLowerCase**

```
public static char toLowerCase(char ch)
```

Returns the lower case character value of the specified ISO-LATIN-1 character. Characters that are not upper case letters are returned unmodified.

Parameters:

ch - the character to be converted

Interface Index **toUpperCase**

```
public static char toUpperCase(char ch)
```

Returns the upper case character value of the specified ISO-LATIN-1 character. Characters that are not lower case letters are returned unmodified. Note that German ess-zed and latin small letter y diaeresis have no corresponding upper case letters, even though they are lower case. There is a capital y diaeresis, but not in ISO-LATIN-1...

Parameters:

ch - the character to be converted

Interface Index **digit**

```
public static int digit(char ch,
                        int radix)
```

Returns the numeric value of the character digit using the specified radix. If the character is not a valid digit, it returns -1.

Parameters:

ch - the character to be converted

radix - the radix

Interface Index **forDigit**

```
public static char forDigit(int digit,
                            int radix)
```

Returns the character value for the specified digit in the specified radix. If the digit is not valid in the radix, the 0 character is returned.

Parameters:

digit - the digit chosen by the character value

radix - the radix containing the digit

Interface Index **charValue**

```
public char charValue()
```

Returns the value of this Character object.

hashCode

```
public int hashCode()
```

Returns a hashcode for this Character.

Overrides:

hashCode in class Object

equals

```
public boolean equals(Object obj)
```

Compares this object against the specified object.

Parameters:

obj - the object to compare with

Returns:

true if the objects are the same; false otherwise.

Overrides:

equals in class Object

toString

```
public String toString()
```

Returns a String object representing this character's value.

Overrides:

toString in class Object

Class java.lang.Class

Class java.lang.Class

```
java.lang.Object
    |
    +----java.lang.Class
```

---

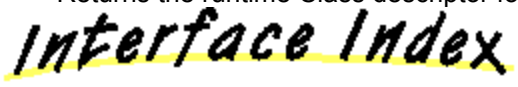
public final class **Class**
extends Object

Class objects contain runtime representations of classes. Every object in the system is an instance of some Class, and for each Class there is one of these descriptor objects. A Class descriptor is not modifiable at runtime.

The following example uses a Class object to print the Class name of an object:

```
void printClassName(Object obj) {
    System.out.println("The class of " + obj +
                       " is " + obj.getClass().getName());
}
```

forName(String)
Returns the runtime Class descriptor for the specified Class.



getClassLoader()
Returns the Class loader of this Class.

getInterfaces()
Returns the interfaces of this Class.

getName()
Returns the name of this Class.

getSuperclass()
Returns the superclass of this Class.

Interface Index **isInterface**()
Returns a boolean indicating whether or not this Class is an interface.
Interface Index **newInstance**()
Creates a new instance of this Class.
Interface Index **toString**()
Returns the name of this class or interface.

Interface Index

Interface Index **forName**

```
public static Class forName(String className) throws ClassNotFoundException
```

Returns the runtime Class descriptor for the specified Class. For example, the following code fragment returns the runtime Class descriptor for the Class named java.lang.Thread:

```
Class t = Class.forName("java.lang.Thread")
```

Parameters:
className - the fully qualified name of the desired Class
Throws: ClassNotFoundException
If the Class could not be found.

Interface Index **newInstance**

```
public Object newInstance() throws InstantiationException,
IllegalAccessException
```

Creates a new instance of this Class.
Returns:
the new instance of this Class.
Throws: InstantiationException
If you try to instantiate an abstract class or an interface, or if the instantiation fails for some other reason.
Throws: IllegalAccessException
If the class or initializer is not accessible.

getName

```
public String getName()
```

Returns the name of this Class.

getSuperclass

```
public Class getSuperclass()
```

Returns the superclass of this Class.

getInterfaces

```
public Class[] getInterfaces()
```

Returns the interfaces of this Class. An array of length 0 is returned if this Class implements no interfaces.

getClassLoader

```
public ClassLoader getClassLoader()
```

Returns the Class loader of this Class. Returns null if this Class does not have a Class loader.

See Also:

ClassLoader

isInterface

```
public boolean isInterface()
```

Returns a boolean indicating whether or not this Class is an interface.

toString

```
public String toString()
```

Returns the name of this class or interface. The word "class" is prepended if it is a Class; the word

"interface" is prepended if it is an interface.

Overrides:

toString in class Object

Class java.lang.ClassCastException

Class java.lang.ClassCastException

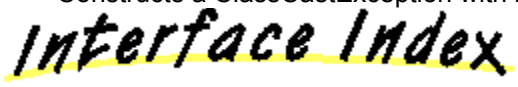
```
java.lang.Object
    |
    +----java.lang.Throwable
            |
            +----java.lang.Exception
                    |
                    +----java.lang.RuntimeException
                            |
                            +----java.lang.ClassCastException
```

---

public class **ClassCastException**
extends RuntimeException

Signals that an invalid cast has occurred.

ClassCastException()
Constructs a ClassCastException with no detail message.



ClassCastException(String)
Constructs a ClassCastException with the specified detail message.

ClassCastException

```
public ClassCastException()
```

Constructs a ClassCastException with no detail message. A detail message is a String that describes this particular exception.

ClassCastException

```
public ClassCastException(String s)
```

Constructs a ClassCastException with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the String containing a detail message

Class java.lang.ClassCircularityError

Class java.lang.ClassCircularityError

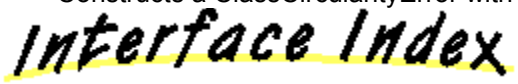
```
java.lang.Object
    |
    +----java.lang.Throwable
            |
            +----java.lang.Error
                    |
                    +----java.lang.LinkageError
                            |
                            +----java.lang.ClassCircularityError
```

---

public class **ClassCircularityError**
extends LinkageError

Signals that a circularity has been detected when initializing a class.

ClassCircularityError()
Constructs a ClassCircularityError with no detail message.



ClassCircularityError(String)
Constructs a ClassCircularityError with the specified detail message.

ClassCircularityError

```
public ClassCircularityError()
```

Constructs a ClassCircularityError with no detail message. A detail message is a String that describes this particular exception.

```
public ClassCircularityError(String s)
```

Constructs a ClassCircularityError with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class java.lang.ClassFormatError

Class java.lang.ClassFormatError

```
java.lang.Object
    |
    +----java.lang.Throwable
             |
             +----java.lang.Error
                      |
                      +----java.lang.LinkageError
                               |
                               +----java.lang.ClassFormatError
```

---

public class **ClassFormatError**
extends LinkageError

Signals an invalid file format has occurred.

```
public ClassFormatError()
```

Constructs a ClassFormatError with no detail message. A detail message is a String that describes this particular exception.

ClassFormatError

```
public ClassFormatError(String s)
```

Constructs a ClassFormatError with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the String containing the detail message

Class java.lang.ClassLoader

Class java.lang.ClassLoader

```
java.lang.Object
   |
   +----java.lang.ClassLoader
```

---

public class **ClassLoader**
extends Object

ClassLoader is an abstract Class that can be used to define a policy for loading Java classes into the runtime environment. By default, the runtime system loads classes that originate as files by reading them from the directory defined by the CLASSPATH environment variable (this is platform dependent). The default mechanism does not involve a Class loader.

However, some classes may not originate from a file; they could be loaded from some other source, e.g., the network. Classes loaded from the network are an array of bytes. A ClassLoader can be used to tell the runtime system to convert an array of bytes into an instance of class Class. This conversion information is passed to the runtime using the defineClass() method.

Classes that are created through the defineClass() mechanism can reference other classes by name. To resolve those names, the runtime system calls the ClassLoader that originally created the Class. The runtime system calls the abstract method loadClass() to load the referenced classes.

```
ClassLoader loader = new NetworkClassLoader(host, port);
Object main = loader.loadClass("Main").newInstance();
....
```

The NetworkClassLoader subclass must define the method loadClass() to load a Class from the network. Once it has downloaded the bytes that make up the Class it should use the method defineClass() to create a Class instance. A sample implementation could be:

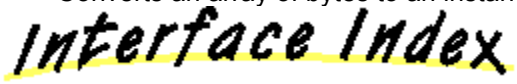
```
class NetworkClassLoader {
    String host;
    int port;
    Hashtable cache = new Hashtable();
    private byte loadClassData(String name)[] {
        // load the class data from the connection
        ...
    }
    public synchronized Class loadClass(String name) {
        Class c = cache.get(name);
        if (c == null) {
```

```
                byte data[] = loadClassData(name);
                cache.put(name, defineClass(data, 0, data.length));
            }
            return c;
        }
    }
```

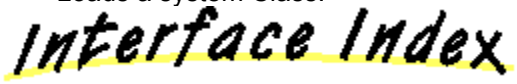
See Also:
Class

ClassLoader()
Constructs a new Class loader and initializes it.

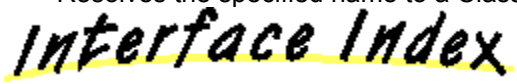
defineClass(byte[], int, int)
Converts an array of bytes to an instance of class Class.



findSystemClass(String)
Loads a system Class.



loadClass(String, boolean)
Resolves the specified name to a Class.



resolveClass(Class)
Resolves classes referenced by this Class.

ClassLoader

```
protected ClassLoader()
```

Constructs a new Class loader and initializes it.

Interface Index

Interface Index **loadClass**

```
  protected abstract Class loadClass(String name,
                                     boolean resolve) throws
ClassNotFoundException
```

Resolves the specified name to a Class. The method loadClass() is called by the virtual machine. As an abstract method, loadClass() must be defined in a subclass of ClassLoader. By using a Hashtable, you can avoid loading the same Class more than once.

Parameters:

name - the name of the desired Class

resolve - true if the Class needs to be resolved

Returns:

the resulting Class, or null if it was not found.

Throws: ClassNotFoundException

Cannot find a definition for the class

See Also:

Hashtable

Interface Index **defineClass**

```
  protected final Class defineClass(byte data[],
                                    int offset,
                                    int length)
```

Converts an array of bytes to an instance of class Class. Before the Class can be used it must be resolved.

Parameters:

data - the bytes that make up the Class

offset - the start offset of the Class data

length - the length of the Class data

Returns:

the Class object which was created from the data.

Throws: ClassFormatError

If the data does not contain a valid Class.

See Also:

loadClass, resolveClass

Interface Index **resolveClass**

```
  protected final void resolveClass(Class c)
```

Resolves classes referenced by this Class. This must be done before the Class can be used. Class names referenced by the resulting Class are resolved by calling loadClass().

Parameters:

c - the Class to be resolved

See Also:

defineClass

findSystemClass

```
protected final Class findSystemClass(String name) throws
ClassNotFoundException
```

Loads a system Class. A system Class is a class with the primordial Class loader (which is null).

Parameters:

name - the name of the system Class

Throws: NoClassDefFoundError

If the Class is not found.

Throws: ClassNotFoundException

Cannot find a definition for the class

Class java.lang.ClassNotFoundException

Class java.lang.ClassNotFoundException

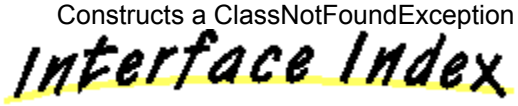
```
java.lang.Object
    |
    +----java.lang.Throwable
             |
             +----java.lang.Exception
                      |
                      +----java.lang.ClassNotFoundException
```

---

public class **ClassNotFoundException**
extends Exception

Signals that a class could not be found.

ClassNotFoundException()
Constructs a ClassNotFoundException with no detail message.



ClassNotFoundException(String)
Constructs a ClassNotFoundException with the specified detail message.

ClassNotFoundException

```
public ClassNotFoundException()
```

Constructs a ClassNotFoundException with no detail message. A detail message is a String that describes this particular exception.

ClassNotFoundException

```
public ClassNotFoundException(String s)
```

Constructs a ClassNotFoundException with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Interface java.lang.Cloneable

Interface java.lang.Cloneable

public interface **Cloneable**
extends Object

An interface indicating that this object may be copied or cloned.

Class java.lang.CloneNotSupportedExceptio n

Class java.lang.CloneNotSupportedException

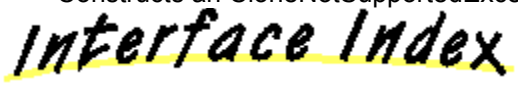
```
java.lang.Object
   |
   +----java.lang.Throwable
           |
           +----java.lang.Exception
                   |
                   +----java.lang.CloneNotSupportedException
```

---

public class **CloneNotSupportedException**
extends Exception

Signals that an attempt has been made to clone an object that does not want to be cloned.

CloneNotSupportedException()
Constructs an CloneNotSupportedException with no detail message.



CloneNotSupportedException(String)
Constructs an CloneNotSupportedException with the specified detail message.

CloneNotSupportedException

```
public CloneNotSupportedException()
```

Constructs an CloneNotSupportedException with no detail message. A detail message is a String that describes this particular exception.


```
public CloneNotSupportedException(String s)
```

Constructs an CloneNotSupportedException with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class java.lang.Compiler

Class java.lang.Compiler

```
java.lang.Object
   |
   +----java.lang.Compiler
```

---

public final class **Compiler**
extends Object

compileClass

```
public static boolean compileClass(Class clazz)
```

compileClasses

public static boolean compileClasses(String string)

command

public static Object command(Object any)

enable

public static void enable()

disable

public static void disable()

Class java.lang.Double

Class java.lang.Double

```
java.lang.Object
    |
    +----java.lang.Number
            |
            +----java.lang.Double
```

---

public final class **Double**
extends Number

The Double class provides an object wrapper for Double data values and serves as a place for double-oriented operations. A wrapper is useful because most of Java's utility classes require the use of objects. Since doubles are not objects in Java, they need to be "wrapped" in a Double instance.

MAX_VALUE

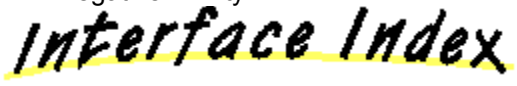
The maximum value a double can have.

MIN_VALUE

The minimum value a double can have.

NEGATIVE_INFINITY

Negative infinity.



NaN

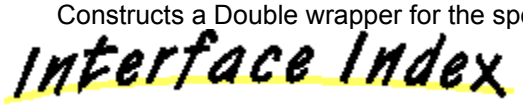
Not-a-Number.

POSITIVE_INFINITY

Positive infinity.

Double(double)

Constructs a Double wrapper for the specified double value.

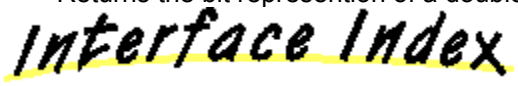


Double(String)

Constructs a Double object initialized to the value specified by the String parameter.

doubleToLongBits(double)

Returns the bit represention of a double-float value

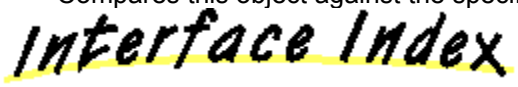


doubleValue()

Returns the double value of this Double.

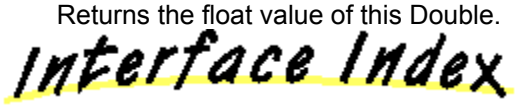
equals(Object)

Compares this object against the specified object.



floatValue()

Returns the float value of this Double.

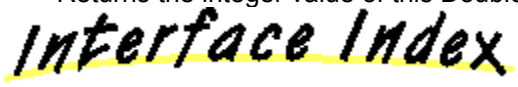


hashCode()

Returns a hashcode for this Double.

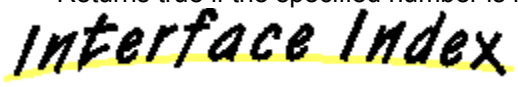
intValue()

Returns the integer value of this Double (by casting to an int).



isInfinite(double)

Returns true if the specified number is infinitely large in magnitude.

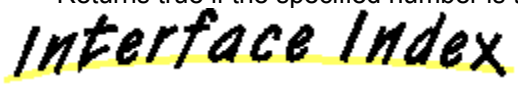


isInfinite()

Returns true if this Double value is infinitely large in magnitude.

isNaN(double)

Returns true if the specified number is the special Not-a-Number (NaN) value.

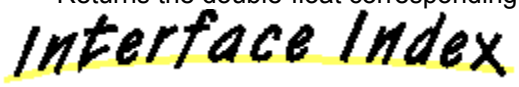


isNaN()

Returns true if this Double value is the special Not-a-Number (NaN) value.

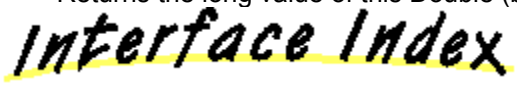
longBitsToDouble(long)

Returns the double-float corresponding to a given bit represention.



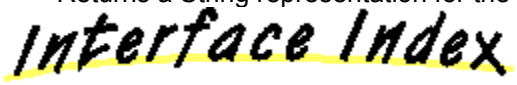
longValue()

Returns the long value of this Double (by casting to a long).



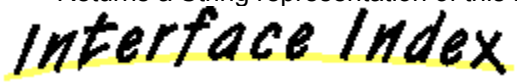
toString(double)

Returns a String representation for the specified double value.



toString()

Returns a String representation of this Double object.



valueOf(String)

Returns a new Double value initialized to the value represented by the specified String.

POSITIVE_INFINITY

```
public final static double POSITIVE_INFINITY
```

Positive infinity.

NEGATIVE_INFINITY

```
public final static double NEGATIVE_INFINITY
```

Negative infinity.

NaN

```
public final static double NaN
```

Not-a-Number. *Note: is not equal to anything, including itself*

MAX_VALUE

```
public final static double MAX_VALUE
```

The maximum value a double can have. The greatest maximum value that a double can have is 1.79769313486231570e+308d.

MIN_VALUE

```
public final static double MIN_VALUE
```

The minimum value a double can have. The lowest minimum value that a double can have is 4.94065645841246544e-324d.

Interface Index

Interface Index **Double**

```
public Double(double value)
```

Constructs a Double wrapper for the specified double value.
Parameters:
value - the initial value of the double

Interface Index **Double**

```
public Double(String s) throws NumberFormatException
```

Constructs a Double object initialized to the value specified by the String parameter.
Parameters:
s - the String to be converted to a Double
Throws: NumberFormatException
If the String does not contain a parsable number.

Interface Index

Interface Index **toString**

```
public static String toString(double d)
```

Returns a String representation for the specified double value.
Parameters:
d - the double to be converted

valueOf

```
public static Double valueOf(String s) throws NumberFormatException
```

Returns a new Double value initialized to the value represented by the specified String.

Parameters:

s - the String to be parsed

Throws: NumberFormatException

If the String cannot be parsed.

isNaN

```
public static boolean isNaN(double v)
```

Returns true if the specified number is the special Not-a-Number (NaN) value.

Parameters:

v - the value to be tested

isInfinite

```
public static boolean isInfinite(double v)
```

Returns true if the specified number is infinitely large in magnitude.

Parameters:

v - the value to be tested

isNaN

```
public boolean isNaN()
```

Returns true if this Double value is the special Not-a-Number (NaN) value.

isInfinite

```
public boolean isInfinite()
```

Returns true if this Double value is infinitely large in magnitude.

Interface Index **toString**

```
public String toString()
```

Returns a String representation of this Double object.
Overrides:
toString in class Object

Interface Index **intValue**

```
public int intValue()
```

Returns the integer value of this Double (by casting to an int).
Overrides:
intValue in class Number

Interface Index **longValue**

```
public long longValue()
```

Returns the long value of this Double (by casting to a long).
Overrides:
longValue in class Number

Interface Index **floatValue**

```
public float floatValue()
```

Returns the float value of this Double.
Overrides:
floatValue in class Number

Interface Index **doubleValue**

```
public double doubleValue()
```

Returns the double value of this Double.
Overrides:
doubleValue in class Number

hashCode

```
public int hashCode()
```

Returns a hashcode for this Double.
Overrides:
hashCode in class Object

equals

```
public boolean equals(Object obj)
```

Compares this object against the specified object.

Note: To be useful in hashtables this method considers two NaN double values to be equal. This is not according to IEEE specification
Parameters:
obj - the object to compare with
Returns:
true if the objects are the same; false otherwise.
Overrides:
equals in class Object

doubleToLongBits

```
public static long doubleToLongBits(double value)
```

Returns the bit represention of a double-float value

longBitsToDouble

```
public static double longBitsToDouble(long bits)
```

Returns the double-float corresponding to a given bit represention.

Class java.lang.Error

Class java.lang.Error

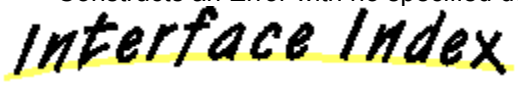
```
java.lang.Object
    |
    +----java.lang.Throwable
             |
             +----java.lang.Error
```

---

public class **Error**
extends Throwable

Error is a subtype of Throwable for abnormal events that should not occur. Do not try to catch Error's unless you really know what you're doing.

Error()
Constructs an Error with no specified detail message.



Error(String)
Constructs an Error with the specified detail message.

Error

```
public Error()
```

Constructs an Error with no specified detail message. A detail message is a String that describes this particular error.

Error

```
public Error(String s)
```

Constructs an Error with the specified detail message. A detail message is a String that describes this particular error

Parameters:

s - the detail message

Class java.lang.Exception

Class java.lang.Exception

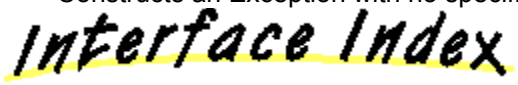
```
java.lang.Object
    |
    +----java.lang.Throwable
             |
             +----java.lang.Exception
```

---

public class **Exception**
extends Throwable

Exception are a form of Throwable that normal programs may wish to try and catch.

Exception()
Constructs an Exception with no specified detail message.



Exception(String)
Constructs a Exception with the specified detail message.

Exception

```
public Exception()
```

Constructs an Exception with no specified detail message. A detail message is a String that describes this particular exception.

Exception

```
public Exception(String s)
```

Constructs a Exception with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class java.lang.Float

Class java.lang.Float

```
java.lang.Object
    |
    +----java.lang.Number
            |
            +----java.lang.Float
```

---

public final class **Float**
extends Number

The Float class provides an object wrapper for Float data values, and serves as a place for float-oriented operations. A wrapper is useful because most of Java's utility classes require the use of objects. Since floats are not objects in Java, they need to be "wrapped" in a Float instance.

MAX_VALUE

The maximum value a float can have.

MIN_VALUE

The minimum value a float can have.

NEGATIVE_INFINITY

Negative infinity.

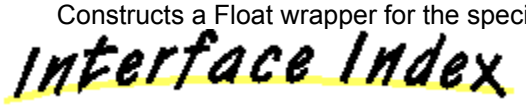
NaN

Not-a-Number.

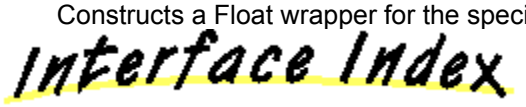
POSITIVE_INFINITY

Positive infinity.

Float(float)
Constructs a Float wrapper for the specified float value.



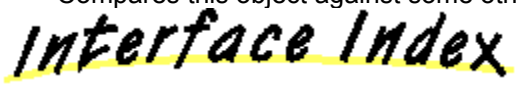
Float(double)
Constructs a Float wrapper for the specified double value.



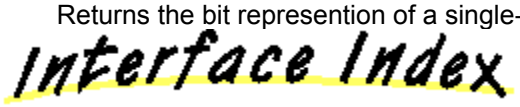
Float(String)
Constructs a Float object initialized to the value specified by the String parameter.

doubleValue()
Returns the double value of this Float.

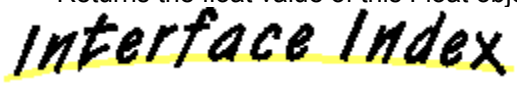
equals(Object)
Compares this object against some other object.



floatToIntBits(float)
Returns the bit represention of a single-float value

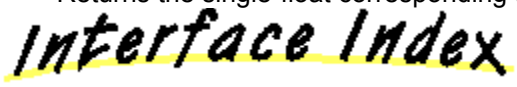


floatValue()
Returns the float value of this Float object.

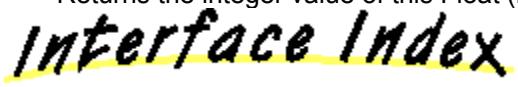


hashCode()
Returns a hashcode for this Float.

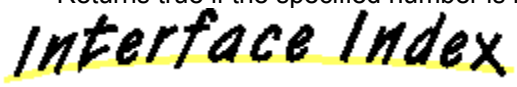
intBitsToFloat(int)
Returns the single-float corresponding to a given bit represention.



intValue()
Returns the integer value of this Float (by casting to an int).

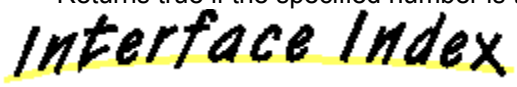


isInfinite(float)
Returns true if the specified number is infinitely large in magnitude.



isInfinite()
Returns true if this Float value is infinitely large in magnitude.

isNaN(float)
Returns true if the specified number is the special Not-a-Number (NaN) value.

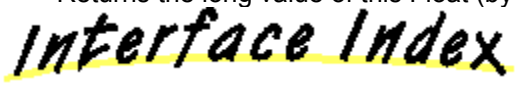


isNaN()

Returns true if this Float value is Not-a-Number (NaN).

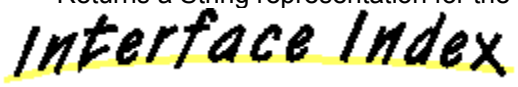
longValue()

Returns the long value of this Float (by casting to a long).



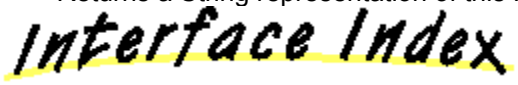
toString(float)

Returns a String representation for the specified float value.



toString()

Returns a String representation of this Float object.



valueOf(String)

Returns the floating point value represented by the specified String.

POSITIVE_INFINITY

```
public final static float POSITIVE_INFINITY
```

Positive infinity.

NEGATIVE_INFINITY

```
public final static float NEGATIVE_INFINITY
```

Negative infinity.

NaN

```
public final static float NaN
```

Not-a-Number. *Note: is not equal to anything, including itself*

MAX_VALUE

```
public final static float MAX_VALUE
```

The maximum value a float can have. The largest maximum value possible is 3.40282346638528860e+38.

MIN_VALUE

```
public final static float MIN_VALUE
```

The minimum value a float can have. The lowest minimum value possible is 1.40129846432481707e-45.

Float

```
public Float(float value)
```

Constructs a Float wrapper for the specified float value.
Parameters:
value - the value of the Float

Float

```
public Float(double value)
```

Constructs a Float wrapper for the specified double value.
Parameters:
value - the value of the Float

Float

```
public Float(String s) throws NumberFormatException
```

Constructs a Float object initialized to the value specified by the String parameter.
Parameters:
s - the String to be converted to a Float
Throws: NumberFormatException
If the String does not contain a parsable number.

Interface Index

Interface Index **toString**

```
public static String toString(float f)
```

Returns a String representation for the specified float value.
Parameters:
f - the float to be converted

Interface Index **valueOf**

```
public static Float valueOf(String s) throws NumberFormatException
```

Returns the floating point value represented by the specified String.
Parameters:
s - the String to be parsed
Throws: NumberFormatException
If the String does not contain a parsable Float.

Interface Index **isNaN**

```
public static boolean isNaN(float v)
```

Returns true if the specified number is the special Not-a-Number (NaN) value.
Parameters:
v - the value to be tested

Interface Index **isInfinite**

```
public static boolean isInfinite(float v)
```

Returns true if the specified number is infinitely large in magnitude.
Parameters:
v - the value to be tested

Interface Index **isNaN**

```
public boolean isNaN()
```

Returns true if this Float value is Not-a-Number (NaN).

Interface Index **isInfinite**

```
public boolean isInfinite()
```

Returns true if this Float value is infinitely large in magnitude.

Interface Index **toString**

```
public String toString()
```

Returns a String representation of this Float object.
Overrides:
toString in class Object

Interface Index **intValue**

```
public int intValue()
```

Returns the integer value of this Float (by casting to an int).
Overrides:
intValue in class Number

Interface Index **longValue**

```
public long longValue()
```

Returns the long value of this Float (by casting to a long).
Overrides:
longValue in class Number

floatValue

```
public float floatValue()
```

Returns the float value of this Float object.

Overrides:

floatValue in class Number

doubleValue

```
public double doubleValue()
```

Returns the double value of this Float.

Overrides:

doubleValue in class Number

hashCode

```
public int hashCode()
```

Returns a hashcode for this Float.

Overrides:

hashCode in class Object

equals

```
public boolean equals(Object obj)
```

Compares this object against some other object.

Note: To be useful in hashtables this method considers two Nan floating point values to be equal. This is not according to IEEE specification

Parameters:

obj - the object to compare with

Returns:

true if the objects are the same; false otherwise.

Overrides:

equals in class Object

```
public static int floatToIntBits(float value)
```

Returns the bit represention of a single-float value


```
public static float intBitsToFloat(int bits)
```

Returns the single-float corresponding to a given bit represention.

Class java.lang.IllegalAccessError

Class java.lang.IllegalAccessError

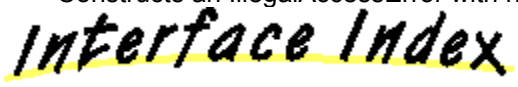
```
java.lang.Object
    |
    +----java.lang.Throwable
            |
            +----java.lang.Error
                    |
                    +----java.lang.LinkageError
                            |
                            +----java.lang.IncompatibleClassChangeError
                                    |
                                    +----java.lang.IllegalAccessError
```

---

public class **IllegalAccessError**
extends IncompatibleClassChangeError

Signals that an illegal access exception has occurred.

IllegalAccessError()
Constructs an IllegalAccessError with no detail message.



IllegalAccessError(String)
Constructs an IllegalAccessError with the specified detail message.

IllegalAccessError

```
public IllegalAccessError()
```

Constructs an IllegalAccessError with no detail message. A detail message is a String that describes this particular exception.


```
public IllegalAccessError(String s)
```

Constructs an IllegalAccessError with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class java.lang.IllegalAccessException

Class java.lang.IllegalAccessException

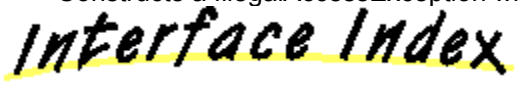
```
java.lang.Object
    |
    +----java.lang.Throwable
            |
            +----java.lang.Exception
                    |
                    +----java.lang.IllegalAccessException
```

---

public class **IllegalAccessException**
extends Exception

Signals that a particular method could not be found.

IllegalAccessException()
Constructs a IllegalAccessException without a detail message.



IllegalAccessException(String)
Constructs a IllegalAccessException with a detail message.

IllegalAccessException

```
public IllegalAccessException()
```

Constructs a IllegalAccessException without a detail message. A detail message is a String that describes this particular exception.

```
public IllegalAccessException(String s)
```

Constructs a IllegalAccessException with a detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class java.lang.IllegalArgumentException

Class java.lang.IllegalArgumentException

```
java.lang.Object
    |
    +----java.lang.Throwable
            |
            +----java.lang.Exception
                    |
                    +----java.lang.RuntimeException
                            |
                            +----java.lang.IllegalArgumentException
```

---

public class **IllegalArgumentException**
extends RuntimeException

Signals that an illegal argument exception has occurred.
See Also:
setPriority

Interface Index

Interface Index

IllegalArgumentException()
Constructs an IllegalArgumentException with no detail message.

Interface Index

IllegalArgumentException(String)
Constructs an IllegalArgumentException with the specified detail message.

Interface Index

Interface Index

IllegalArgumentException

```
public IllegalArgumentException()
```

Constructs an IllegalArgumentException with no detail message. A detail message is a String that describes this particular exception.


```
public IllegalArgumentException(String s)
```

Constructs an IllegalArgumentException with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class java.lang.IllegalMonitorStateException

Class java.lang.IllegalMonitorStateException

```
java.lang.Object
    |
    +----java.lang.Throwable
            |
            +----java.lang.Exception
                    |
                    +----java.lang.RuntimeException
                            |
                            +----java.lang.IllegalMonitorStateException
```

---

public class **IllegalMonitorStateException**
extends RuntimeException

Signals that a monitor operation has been attempted when the monitor is in an invalid state. For example, trying to notify a monitor that you do not own would invoke this class.

Interface Index

Interface Index

IllegalMonitorStateException

```
public IllegalMonitorStateException()
```

Constructs an IllegalMonitorStateException with no detail message. A detail message is a String that describes this particular exception.

IllegalMonitorStateException

```
public IllegalMonitorStateException(String s)
```

Constructs an IllegalMonitorStateException with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the String that contains a detailed message

Class java.lang.IllegalThreadStateException

Class java.lang.IllegalThreadStateException

```
java.lang.Object
    |
    +----java.lang.Throwable
            |
            +----java.lang.Exception
                    |
                    +----java.lang.RuntimeException
                            |
                            +----java.lang.IllegalArgumentException
                                    |
                                    +----java.lang.IllegalThreadStateException
```

---

public class **IllegalThreadStateException**
extends IllegalArgumentException

Exception indicating that a thread is not in the proper state for the requested operation.
See Also:
suspend, resume

Interface Index

Interface Index

IllegalThreadStateException()
Constructs an IllegalThreadStateException with no detail message.

Interface Index

IllegalThreadStateException(String)
Constructs an IllegalThreadStateException with the specified detail message.

Interface Index

```
public IllegalThreadStateException()
```

Constructs an IllegalThreadStateException with no detail message. A detail message is a String that describes this particular exception.

Interface Index IllegalThreadStateException

```
public IllegalThreadStateException(String s)
```

Constructs an IllegalThreadStateException with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class java.lang.IncompatibleClassChangeError

Class java.lang.IncompatibleClassChangeError

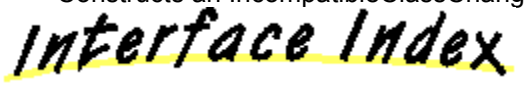
```
java.lang.Object
    |
    +----java.lang.Throwable
            |
            +----java.lang.Error
                    |
                    +----java.lang.LinkageError
                            |
                            +----java.lang.IncompatibleClassChangeError
```

---

public class **IncompatibleClassChangeError**
extends LinkageError

Signals that an incompatible class change has occurred.

IncompatibleClassChangeError()
Constructs an IncompatibleClassChangeError with no detail message.



IncompatibleClassChangeError(String)
Constructs an IncompatibleClassChangeError with the specified detail message.

IncompatibleClassChangeError

```
public IncompatibleClassChangeError()
```

Constructs an IncompatibleClassChangeError with no detail message. A detail message is a String that describes this particular exception.


```
public IncompatibleClassChangeError(String s)
```

Constructs an IncompatibleClassChangeError with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class java.lang.IndexOutOfBoundsException

Class java.lang.IndexOutOfBoundsException

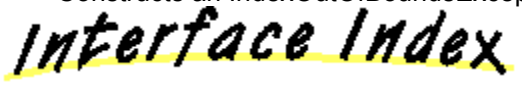
```
java.lang.Object
    |
    +----java.lang.Throwable
            |
            +----java.lang.Exception
                    |
                    +----java.lang.RuntimeException
                            |
                            +----java.lang.IndexOutOfBoundsException
```

---

public class **IndexOutOfBoundsException**
extends RuntimeException

Signals that an index of some sort is out of bounds.

IndexOutOfBoundsException()
Constructs an IndexOutOfBoundsException with no detail message.



IndexOutOfBoundsException(String)
Constructs a IndexOutOfBoundsException with the specified detail message.

IndexOutOfBoundsException

```
public IndexOutOfBoundsException()
```

Constructs an IndexOutOfBoundsException with no detail message. A detail message is a String that describes this particular exception.

IndexOutOfBoundsException

```
public IndexOutOfBoundsException(String s)
```

Constructs a IndexOutOfBoundsException with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class java.lang.InstantiationError

Class java.lang.InstantiationError

```
java.lang.Object
    |
    +----java.lang.Throwable
            |
            +----java.lang.Error
                    |
                    +----java.lang.LinkageError
                            |
                            +----java.lang.IncompatibleClassChangeError
                                    |
                                    +----java.lang.InstantiationError
```

---

public class **InstantiationError**
extends IncompatibleClassChangeError

Signals that the interpreter has tried to instantiate an abstract class or an interface.

InstantiationError()
Constructs an InstantiationError with no detail message.

InstantiationError(String)
Constructs an InstantiationError with the specified detail message.

InstantiationError

```
public InstantiationError()
```

Constructs an InstantiationError with no detail message. A detail message is a String that describes this particular exception.

InstantiationError

```
public InstantiationError(String s)
```

Constructs an InstantiationError with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the String that contains the detail message

Class java.lang.InstantiationException

Class java.lang.InstantiationException

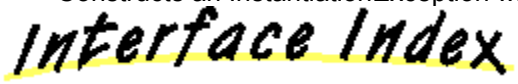
```
java.lang.Object
    |
    +----java.lang.Throwable
            |
            +----java.lang.Exception
                    |
                    +----java.lang.InstantiationException
```

---

public class **InstantiationException**
extends Exception

Signals that an attempt has been made to instantiate an abstract class or an interface.

InstantiationException()
Constructs an InstantiationException with no detail message.



InstantiationException(String)
Constructs an InstantiationException with the specified detail message.

InstantiationException

```
public InstantiationException()
```

Constructs an InstantiationException with no detail message. A detail message is a String that describes this particular exception.

InstantiationException

```
public InstantiationException(String s)
```

Constructs an InstantiationException with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the String containing a detail message

Class java.lang.Integer

Class java.lang.Integer

```
java.lang.Object
    |
    +----java.lang.Number
            |
            +----java.lang.Integer
```

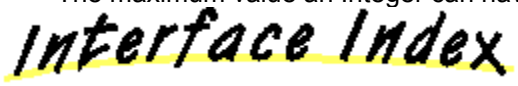
---

public final class **Integer**
extends Number

The Integer class is a wrapper for integer values. In Java, integers are not objects and most of the Java utility classes require the use of objects. Thus, if you needed to store an integer in a hashtable, you would have to "wrap" an Integer instance around it.

MAX_VALUE

The maximum value an Integer can have.

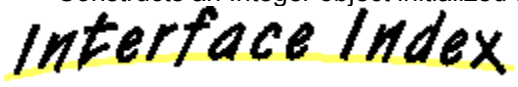


MIN_VALUE

The minimum value an Integer can have.

Integer(int)

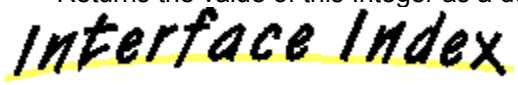
Constructs an Integer object initialized to the specified int value.



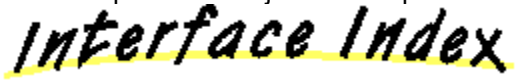
Integer(String)

Constructs an Integer object initialized to the value specified by the String parameter.

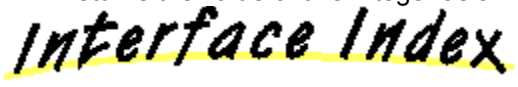
doubleValue()
Returns the value of this Integer as a double.



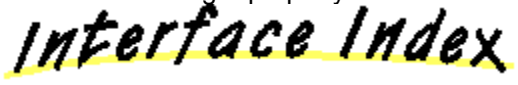
equals(Object)
Compares this object to the specified object.



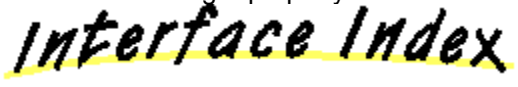
floatValue()
Returns the value of this Integer as a float.



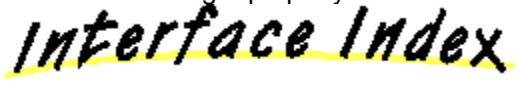
getInteger(String)
Gets an Integer property.



getInteger(String, int)
Gets an Integer property.



getInteger(String, Integer)
Gets an Integer property.

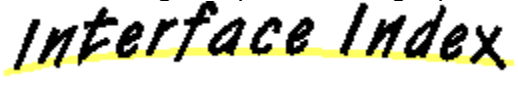


hashCode()
Returns a hashcode for this Integer.

intValue()
Returns the value of this Integer as an int.

longValue()
Returns the value of this Integer as a long.

parseInt(String, int)
Assuming the specified String represents an integer, returns that integer's value.



parseInt(String)
Assuming the specified String represents an integer, returns that integer's value.

toString(int, int)
Returns a new String object representing the specified integer in the specified radix.

toString(int)
Returns a new String object representing the specified integer.

toString()
Returns a String object representing this Integer's value.

valueOf(String, int)

Assuming the specified String represents an integer, returns a new Integer object initialized to that value.

valueOf(String)

Assuming the specified String represents an integer, returns a new Integer object initialized to that value.

MIN_VALUE

```
public final static int MIN_VALUE
```

The minimum value an Integer can have. The lowest minimum value an Integer can have is 0x80000000.

MAX_VALUE

```
public final static int MAX_VALUE
```

The maximum value an Integer can have. The greatest maximum value an Integer can have is 0x7fffffff.

Integer

```
public Integer(int value)
```

Constructs an Integer object initialized to the specified int value.

Parameters:

value - the initial value of the Integer

Integer

```
public Integer(String s) throws NumberFormatException
```

Constructs an Integer object initialized to the value specified by the String parameter. The radix is assumed to be 10.

Parameters:

s - the String to be converted to an Integer

Throws: NumberFormatException

If the String does not contain a parsable integer.

Interface Index

Interface Index **toString**

```
public static String toString(int i,
                              int radix)
```

Returns a new String object representing the specified integer in the specified radix.

Parameters:

i - the integer to be converted

radix - the radix

See Also:

MIN_RADIX, MAX_RADIX

Interface Index **toString**

```
public static String toString(int i)
```

Returns a new String object representing the specified integer. The radix is assumed to be 10.

Parameters:

i - the integer to be converted

Interface Index **parseInt**

```
public static int parseInt(String s,
                           int radix) throws NumberFormatException
```

Assuming the specified String represents an integer, returns that integer's value. Throws an exception if the String cannot be parsed as an int.

Parameters:

s - the String containing the integer
radix - the radix to be used

Throws: NumberFormatException
If the String does not contain a parsable integer.

parseInt

```
public static int parseInt(String s) throws NumberFormatException
```

Assuming the specified String represents an integer, returns that integer's value. Throws an exception if the String cannot be parsed as an int. The radix is assumed to be 10.

Parameters:
s - the String containing the integer

Throws: NumberFormatException
If the string does not contain a parsable integer.

valueOf

```
public static Integer valueOf(String s,
                              int radix) throws NumberFormatException
```

Assuming the specified String represents an integer, returns a new Integer object initialized to that value. Throws an exception if the String cannot be parsed as an int.

Parameters:
s - the String containing the integer
radix - the radix to be used

Throws: NumberFormatException
If the String does not contain a parsable integer.

valueOf

```
public static Integer valueOf(String s) throws NumberFormatException
```

Assuming the specified String represents an integer, returns a new Integer object initialized to that value. Throws an exception if the String cannot be parsed as an int. The radix is assumed to be 10.

Parameters:
s - the String containing the integer

Throws: NumberFormatException
If the String does not contain a parsable integer.

 intValue

```
public int intValue()
```

Returns the value of this Integer as an int.
Overrides:
intValue in class Number

 longValue

```
public long longValue()
```

Returns the value of this Integer as a long.
Overrides:
longValue in class Number

 floatValue

```
public float floatValue()
```

Returns the value of this Integer as a float.
Overrides:
floatValue in class Number

 doubleValue

```
public double doubleValue()
```

Returns the value of this Integer as a double.
Overrides:
doubleValue in class Number

 toString

```
public String toString()
```

Returns a String object representing this Integer's value.
Overrides:
toString in class Object

hashCode

```
public int hashCode()
```

Returns a hashcode for this Integer.
Overrides:
hashCode in class Object

equals

```
public boolean equals(Object obj)
```

Compares this object to the specified object.
Parameters:
obj - the object to compare with
Returns:
true if the objects are the same; false otherwise.
Overrides:
equals in class Object

getInteger

```
public static Integer getInteger(String nm)
```

Gets an Integer property. If the property does not exist, it will return 0.
Parameters:
nm - the property name

getInteger

```
public static Integer getInteger(String nm,
                                 int val)
```

Gets an Integer property. If the property does not exist, it will return val. Deals with Hexadecimal and octal numbers.

Parameters:

nm - the String name
val - the Integer value

getInteger

```
public static Integer getInteger(String nm,
                                 Integer val)
```

Gets an Integer property. If the property does not exist, it will return val. Deals with Hexadecimal and octal numbers.

Parameters:

nm - the property name
val - the integer value

Class java.lang.InternalError

Class java.lang.InternalError

```
java.lang.Object
    |
    +----java.lang.Throwable
            |
            +----java.lang.Error
                    |
                    +----java.lang.VirtualMachineError
                            |
                            +----java.lang.InternalError
```

---

public class **InternalError**
extends VirtualMachineError

Signals that an internal error has occurred.

InternalError()
Constructs an InternalError with no detail message.

InternalError(String)
Constructs an InternalError with the specified detail message.

InternalError

```
public InternalError()
```

Constructs an InternalError with no detail message. A detail message is a String that describes this particular exception.

```
public InternalError(String s)
```

Constructs an InternalError with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class java.lang.InterruptedException

Class java.lang.InterruptedException

```
java.lang.Object
    |
    +----java.lang.Throwable
            |
            +----java.lang.Exception
                    |
                    +----java.lang.InterruptedException
```

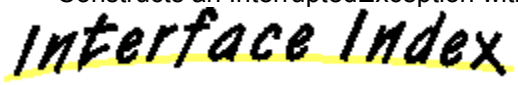
---

public class **InterruptedException**
extends Exception

An exception indicated that some thread has interrupted this thread.

See Also:
interrupt, interrupted

InterruptedException()
Constructs an InterruptedException with no detail message.



InterruptedException(String)
Constructs an InterruptedException with the specified detail message.

InterruptedException

```
public InterruptedException()
```

Constructs an InterruptedException with no detail message. A detail message is a String that describes this particular exception.

InterruptedException

```
public InterruptedException(String s)
```

Constructs an InterruptedException with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class java.lang.LinkageError

Class java.lang.LinkageError

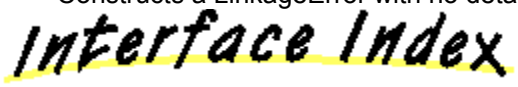
```
java.lang.Object
    |
    +----java.lang.Throwable
            |
            +----java.lang.Error
                    |
                    +----java.lang.LinkageError
```

---

public class **LinkageError**
extends Error

LinkageError and its subclasses indicate that a class has some dependency on another class; however the latter class has incompatibly changed after the compilation of the former class.

LinkageError()
Constructs a LinkageError with no detail message.



LinkageError(String)
Constructs a LinkageError with the specified detail message.

LinkageError

```
public LinkageError()
```

Constructs a LinkageError with no detail message. A detail message is a String that describes this particular exception.

```
public LinkageError(String s)
```

Constructs a LinkageError with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class java.lang.Long

Class java.lang.Long

```
java.lang.Object
    |
    +----java.lang.Number
            |
            +----java.lang.Long
```

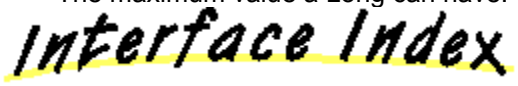
---

public final class **Long**
extends Number

The Long class provides an object wrapper for Long data values and serves as a place for long-oriented operations. A wrapper is useful because most of Java's utility classes require the use of objects. Since longs are not objects in Java, they need to be "wrapped" in a Long instance.

MAX_VALUE

The maximum value a Long can have.

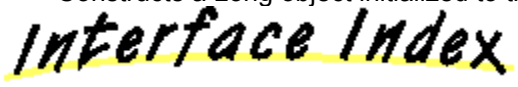


MIN_VALUE

The minimum value a Long can have.

Long(long)

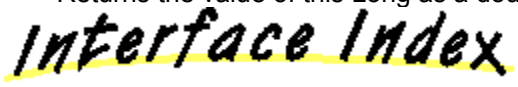
Constructs a Long object initialized to the specified value.



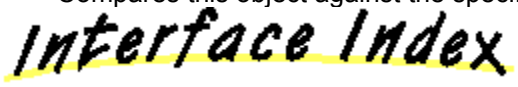
Long(String)

Constructs a Long object initialized to the value specified by the String parameter.

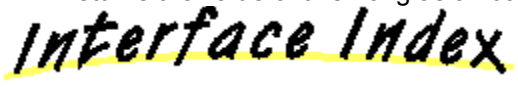
doubleValue()
Returns the value of this Long as a double.



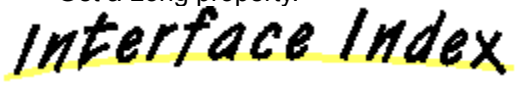
equals(Object)
Compares this object against the specified object.



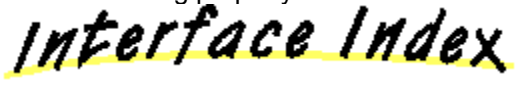
floatValue()
Returns the value of this Long as a float.



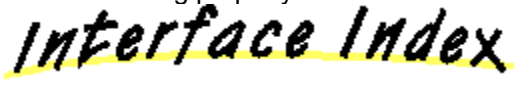
getLong(String)
Get a Long property.



getLong(String, long)
Get a Long property.



getLong(String, Long)
Get a Long property.



hashCode()
Computes a hashcode for this Long.

intValue()
Returns the value of this Long as an int.

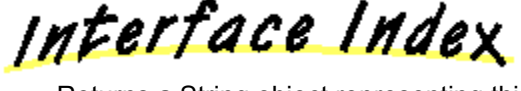
longValue()
Returns the value of this Long as a long.

parseLong(String, int)
Assuming the specified String represents a long, returns that long's value.

parseLong(String)
Assuming the specified String represents a long, return that long's value.

toString(long, int)
Returns a new String object representing the specified long in the specified radix.

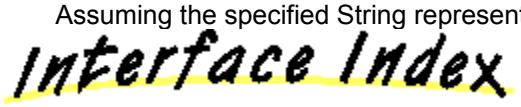
toString(long)
Returns a new String object representing the specified integer.



toString()
Returns a String object representing this Long's value.

valueOf(String, int)

Assuming the specified String represents a long, returns a new Long object initialized to that value.



valueOf(String)

Assuming the specified String represents a long, returns a new Long object initialized to that value.

MIN_VALUE

```
public final static long MIN_VALUE
```

The minimum value a Long can have. The lowest minimum value that a Long can have is 0x8000000000000000.

MAX_VALUE

```
public final static long MAX_VALUE
```

The maximum value a Long can have. The larget maximum value that a Long can have is 0x7fffffffffffffff.

Long

```
public Long(long value)
```

Constructs a Long object initialized to the specified value.

Parameters:

value - the initial value of the Long

Long

```
public Long(String s) throws NumberFormatException
```

Constructs a Long object initialized to the value specified by the String parameter. The radix is assumed to be 10.

Parameters:

s - the String to be converted to a Long

Throws: NumberFormatException

If the String does not contain a parsable long.

toString

```
public static String toString(long i,
                              int radix)
```

Returns a new String object representing the specified long in the specified radix.

Parameters:

i - the long to be converted

radix - the radix

See Also:

MIN_RADIX, MAX_RADIX

toString

```
public static String toString(long i)
```

Returns a new String object representing the specified integer. The radix is assumed to be 10.

Parameters:

i - the long to be converted

parseLong

```
public static long parseLong(String s,
                             int radix) throws NumberFormatException
```

Assuming the specified String represents a long, returns that long's value. Throws an exception if the String cannot be parsed as a long.

Parameters:

s - the String containing the integer

radix - the radix to be used

Throws: NumberFormatException
If the String does not contain a parsable integer.

parseLong

```
public static long parseLong(String s) throws NumberFormatException
```

Assuming the specified String represents a long, return that long's value. Throws an exception if the String cannot be parsed as a long. The radix is assumed to be 10.
Parameters:
s - the String containing the long
Throws: NumberFormatException
If the string does not contain a parsable long.

valueOf

```
public static Long valueOf(String s,
                           int radix) throws NumberFormatException
```

Assuming the specified String represents a long, returns a new Long object initialized to that value. Throws an exception if the String cannot be parsed as a long.
Parameters:
s - the String containing the long.
radix - the radix to be used
Throws: NumberFormatException
If the String does not contain a parsable long.

valueOf

```
public static Long valueOf(String s) throws NumberFormatException
```

Assuming the specified String represents a long, returns a new Long object initialized to that value. Throws an exception if the String cannot be parsed as a long. The radix is assumed to be 10.
Parameters:
s - the String containing the long
Throws: NumberFormatException
If the String does not contain a parsable long.

 intValue

```
public int intValue()
```

Returns the value of this Long as an int.
Overrides:
intValue in class Number

 longValue

```
public long longValue()
```

Returns the value of this Long as a long.
Overrides:
longValue in class Number

 floatValue

```
public float floatValue()
```

Returns the value of this Long as a float.
Overrides:
floatValue in class Number

 doubleValue

```
public double doubleValue()
```

Returns the value of this Long as a double.
Overrides:
doubleValue in class Number

 toString

```
public String toString()
```

Returns a String object representing this Long's value.
Overrides:
toString in class Object

hashCode

```
public int hashCode()
```

Computes a hashcode for this Long.
Overrides:
hashCode in class Object

equals

```
public boolean equals(Object obj)
```

Compares this object against the specified object.
Parameters:
obj - the object to compare with
Returns:
true if the objects are the same; false otherwise.
Overrides:
equals in class Object

getLong

```
public static Long getLong(String nm)
```

Get a Long property. If the property does not exist, it will return 0.
Parameters:
nm - the property name

getLong

```
public static Long getLong(String nm,
                           long val)
```

Get a Long property. If the property does not exist, it will return val. Deals with Hexadecimal and octal numbers.

Parameters:

nm - the String name
val - the Long value

getLong

```
public static Long getLong(String nm,
                           Long val)
```

Get a Long property. If the property does not exist, it will return val. Deals with Hexadecimal and octal numbers.

Parameters:

nm - the property name
val - the Long value

Class java.lang.Math

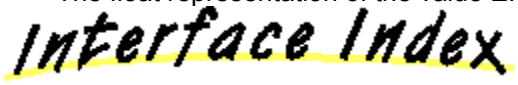
Class java.lang.Math

```
java.lang.Object
    |
    +----java.lang.Math
```

---

public final class **Math**
extends Object

The standard Math library. For the methods in this Class, error handling for out-of-range or immeasurable results are platform dependent. This class cannot be subclassed or instantiated because all methods and variables are static.



E

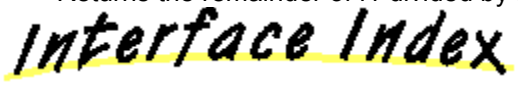
The float representation of the value E.

PI

The float representation of the value Pi.

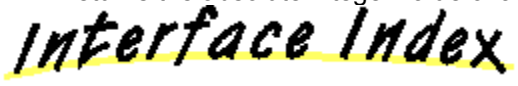
IEEEremainder(double, double)

Returns the remainder of f1 divided by f2 as defined by IEEE 754.



abs(int)

Returns the absolute integer value of a.

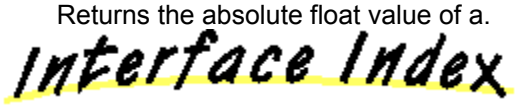


abs(long)

Returns the absolute long value of a.

abs(float)

Returns the absolute float value of a.

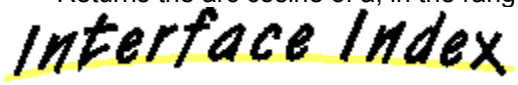


abs(double)

Returns the absolute double value of a.

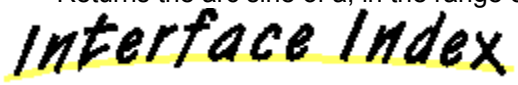
acos(double)

Returns the arc cosine of a, in the range of 0.0 through Pi.



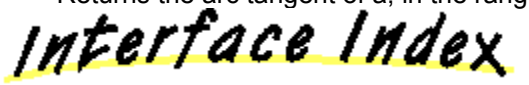
asin(double)

Returns the arc sine of a, in the range of -Pi/2 through Pi/2.



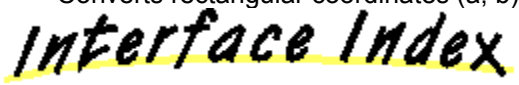
atan(double)

Returns the arc tangent of a, in the range of -Pi/2 through Pi/2.



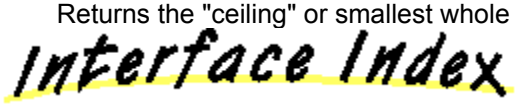
atan2(double, double)

Converts rectangular coordinates (a, b) to polar (r, theta).



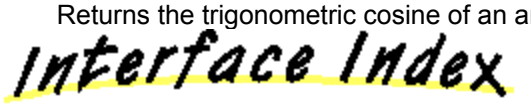
ceil(double)

Returns the "ceiling" or smallest whole number greater than or equal to a.



cos(double)

Returns the trigonometric cosine of an angle.

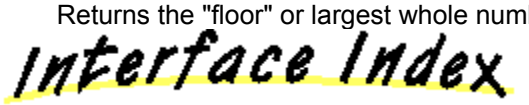


exp(double)

Returns the exponential number e(2.718...) raised to the power of a.

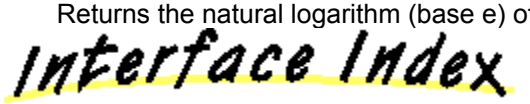
floor(double)

Returns the "floor" or largest whole number less than or equal to a.



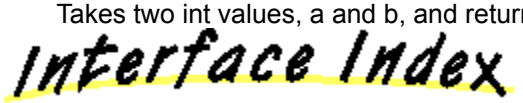
log(double)

Returns the natural logarithm (base e) of a.



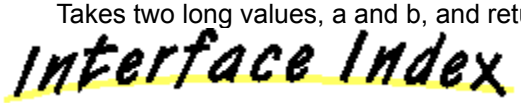
max(int, int)

Takes two int values, a and b, and returns the greater number of the two.



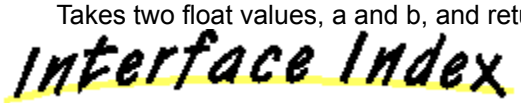
max(long, long)

Takes two long values, a and b, and returns the greater number of the two.



max(float, float)

Takes two float values, a and b, and returns the greater number of the two.



max(double, double)

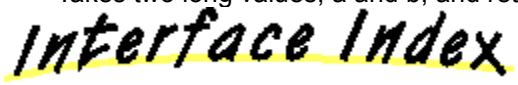
Takes two double values, a and b, and returns the greater number of the two.

min(int, int)

Takes two integer values, a and b, and returns the smallest number of the two.

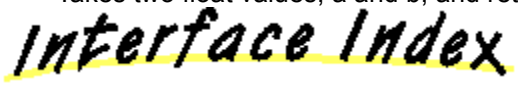
min(long, long)

Takes two long values, a and b, and returns the smallest number of the two.



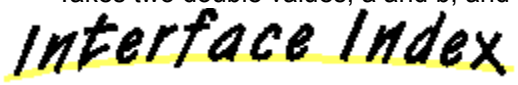
min(float, float)

Takes two float values, a and b, and returns the smallest number of the two.



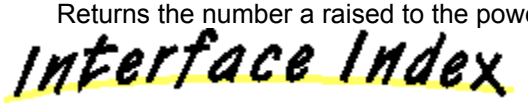
min(double, double)

Takes two double values, a and b, and returns the smallest number of the two.



pow(double, double)

Returns the number a raised to the power of b.

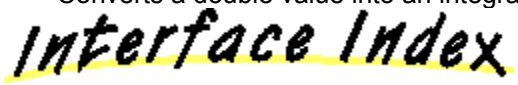


random()

Generates a random number between 0.0 and 1.0.

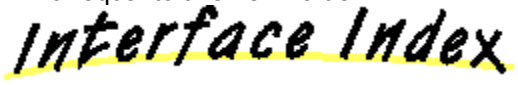
rint(double)

Converts a double value into an integral value in double format.



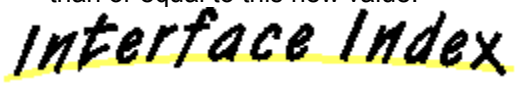
round(float)

Rounds off a float value by first adding 0.5 to it and then returning the largest integer that is less than or equal to this new value.



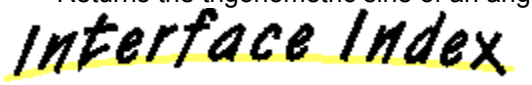
round(double)

Rounds off a double value by first adding 0.5 to it and then returning the largest integer that is less than or equal to this new value.



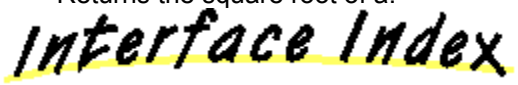
sin(double)

Returns the trigonometric sine of an angle.



sqrt(double)

Returns the square root of a.



tan(double)

Returns the trigonometric tangent of an angle.

E

```
public final static double E
```

The float representation of the value E. E is equivalent to 2.7182818284590452354f in Java.

Interface Index **PI**

```
public final static double PI
```

The float representation of the value Pi. Pi is equivalent to 3.14159265358979323846f in Java.

Interface Index

Interface Index **sin**

```
public static double sin(double a)
```

Returns the trigonometric sine of an angle.
Parameters:
a - an assigned angle that is measured in radians

Interface Index **cos**

```
public static double cos(double a)
```

Returns the trigonometric cosine of an angle.
Parameters:
a - an assigned angle that is measured in radians

Interface Index **tan**

```
public static double tan(double a)
```

Returns the trigonometric tangent of an angle.
Parameters:
a - an assigned angle that is measured in radians

Interface Index **asin**

```
public static double asin(double a)
```

Returns the arc sine of a, in the range of -Pi/2 through Pi/2.
Parameters:
a - (-1.0)

Interface Index **acos**

```
public static double acos(double a)
```

Returns the arc cosine of a, in the range of 0.0 through Pi.
Parameters:
a - (-1.0)

Interface Index **atan**

```
public static double atan(double a)
```

Returns the arc tangent of a, in the range of -Pi/2 through Pi/2.
Parameters:
a - an assigned value
Returns:
the arc tangent of a.

Interface Index **exp**

```
public static double exp(double a)
```

Returns the exponential number e(2.718...) raised to the power of a.
Parameters:
a - an assigned value

Interface Index **log**

```
public static double log(double a) throws ArithmeticException
```

Returns the natural logarithm (base e) of a.

Parameters:
a - a is a number greater than 0.0
Throws: ArithmeticException
If a is less than 0.0 .

sqrt

```
public static double sqrt(double a) throws ArithmeticException
```

Returns the square root of a.
Parameters:
a - a is a number greater than or equal to 0.0
Throws: ArithmeticException
If a is a value less than 0.0 .

IEEEremainder

```
public static double IEEEremainder(double f1,
                                   double f2)
```

Returns the remainder of f1 divided by f2 as defined by IEEE 754.
Parameters:
f1 - the dividend
f2 - the divisor

ceil

```
public static double ceil(double a)
```

Returns the "ceiling" or smallest whole number greater than or equal to a.
Parameters:
a - an assigned value

floor

```
public static double floor(double a)
```

Returns the "floor" or largest whole number less than or equal to a.

Parameters:

a - an assigned value

rint

```
public static double rint(double a)
```

Converts a double value into an integral value in double format.

Parameters:

a - an assigned double value

atan2

```
public static double atan2(double a,
                           double b)
```

Converts rectangular coordinates (a, b) to polar (r, theta). This method computes the phase theta by computing an arc tangent of b/a in the range of -Pi to Pi.

Parameters:

a - an assigned value

b - an assigned value

Returns:

the polar coordinates (r, theta).

pow

```
public static double pow(double a,
                         double b) throws ArithmeticException
```

Returns the number a raised to the power of b. If (a == 0.0), then b must be greater than 0.0; otherwise you will throw an exception. An exception will also occur if (a

Parameters:

a - an assigned value with the exceptions: (a == 0.0) -> (b > 0.0) & (a (b == a whole number)

b - an assigned value with the exceptions: (a == 0.0) -> (b > 0.0) & (a (b == a whole number)

Throws: ArithmeticException

If (a == 0.0) and (b **Throws:** ArithmeticException

If (a

round

```
public static int round(float a)
```

Rounds off a float value by first adding 0.5 to it and then returning the largest integer that is less than or equal to this new value.

Parameters:

a - the value to be rounded off

round

```
public static long round(double a)
```

Rounds off a double value by first adding 0.5 to it and then returning the largest integer that is less than or equal to this new value.

Parameters:

a - the value to be rounded off

random

```
public static synchronized double random()
```

Generates a random number between 0.0 and 1.0.

Random number generators are often referred to as pseudorandom number generators because the numbers produced tend to repeat themselves after a period of time.

Returns:

a pseudorandom double between 0.0 and 1.0.

abs

```
public static int abs(int a)
```

Returns the absolute integer value of a.

Parameters:

a - an assigned integer value

abs

```
public static long abs(long a)
```

Returns the absolute long value of a.
Parameters:
a - an assigned long value.

abs

```
public static float abs(float a)
```

Returns the absolute float value of a.
Parameters:
a - an assigned float value

abs

```
public static double abs(double a)
```

Returns the absolute double value of a.
Parameters:
a - an assigned double value

max

```
public static int max(int a,
                      int b)
```

Takes two int values, a and b, and returns the greater number of the two.
Parameters:
a - an integer value to be compared
b - an integer value to be compared

max

```
public static long max(long a,
                       long b)
```

Takes two long values, a and b, and returns the greater number of the two.

Parameters:

a - a long value to be compared
b - a long value to be compared

max

```
public static float max(float a,
                        float b)
```

Takes two float values, a and b, and returns the greater number of the two.

Parameters:

a - a float value to be compared
b - a float value to be compared

max

```
public static double max(double a,
                         double b)
```

Takes two double values, a and b, and returns the greater number of the two.

Parameters:

a - a double value to be compared
b - a double value to be compared

min

```
public static int min(int a,
                      int b)
```

Takes two integer values, a and b, and returns the smallest number of the two.

Parameters:

a - an integer value to be compared
b - an integer value to be compared

min

```
public static long min(long a,
                       long b)
```

Takes two long values, a and b, and returns the smallest number of the two.

Parameters:

a - a long value to be compared
b - a long value to be compared

min

```
public static float min(float a,
                        float b)
```

Takes two float values, a and b, and returns the smallest number of the two.

Parameters:

a - a float value to be compared
b - a float value to be compared

min

```
public static double min(double a,
                         double b)
```

Takes two double values, a and b, and returns the smallest number of the two.

Parameters:

a - a double value to be compared
b - a double value to be compared

Class java.lang.NegativeArraySizeException

Class java.lang.NegativeArraySizeException

```
java.lang.Object
    |
    +----java.lang.Throwable
            |
            +----java.lang.Exception
                    |
                    +----java.lang.RuntimeException
                            |
                            +----java.lang.NegativeArraySizeException
```

---

public class **NegativeArraySizeException**
extends RuntimeException

Signals that an attempt has been made to create an array with negative size.

Interface Index

Interface Index

NegativeArraySizeException()
Constructs a NegativeArraySizeException with no detail message.

Interface Index

NegativeArraySizeException(String)
Constructs a NegativeArraySizeException with the specified detail message.

Interface Index

Interface Index

NegativeArraySizeException

```
public NegativeArraySizeException()
```

Constructs a NegativeArraySizeException with no detail message. A detail message is a String that describes this particular exception.

NegativeArraySizeException

```
public NegativeArraySizeException(String s)
```

Constructs a NegativeArraySizeException with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class java.lang.NoClassDefFoundError

Class java.lang.NoClassDefFoundError

```
java.lang.Object
    |
    +----java.lang.Throwable
            |
            +----java.lang.Error
                    |
                    +----java.lang.LinkageError
                            |
                            +----java.lang.NoClassDefFoundError
```

---

public class **NoClassDefFoundError**
extends LinkageError

Signals that a class could not be found.

NoClassDefFoundError()
Constructs a NoClassDefFoundError with no detail message.

NoClassDefFoundError(String)
Constructs a NoClassDefFoundError with the specified detail message.

NoClassDefFoundError

```
public NoClassDefFoundError()
```

Constructs a NoClassDefFoundError with no detail message. A detail message is a String that describes this particular exception.

```
public NoClassDefFoundError(String s)
```

Constructs a NoClassDefFoundError with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class java.lang.NoSuchFieldError

Class java.lang.NoSuchFieldError

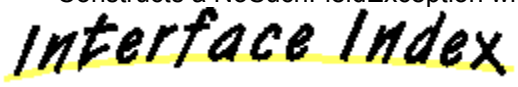
```
java.lang.Object
    |
    +----java.lang.Throwable
            |
            +----java.lang.Error
                    |
                    +----java.lang.LinkageError
                            |
                            +----java.lang.IncompatibleClassChangeError
                                    |
                                    +----java.lang.NoSuchFieldError
```

---

public class **NoSuchFieldError**
extends IncompatibleClassChangeError

Signals that a particular field could not be found.

NoSuchFieldError()
Constructs a NoSuchFieldException without a detail message.



NoSuchFieldError(String)
Constructs a NoSuchFieldException with a detail message.

NoSuchFieldError

```
public NoSuchFieldError()
```

Constructs a NoSuchFieldException without a detail message. A detail message is a String that describes this particular exception.


```
public NoSuchFieldError(String s)
```

Constructs a NoSuchFieldException with a detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class java.lang.NoSuchMethodError

Class java.lang.NoSuchMethodError

```
java.lang.Object
    |
    +----java.lang.Throwable
            |
            +----java.lang.Error
                    |
                    +----java.lang.LinkageError
                            |
                            +----java.lang.IncompatibleClassChangeError
                                    |
                                    +----java.lang.NoSuchMethodError
```

---

public class **NoSuchMethodError**
extends IncompatibleClassChangeError

Signals that a particular method could not be found.

NoSuchMethodError()

NoSuchMethodError(String)
Constructs a NoSuchMethodException with a detail message.

NoSuchMethodError

```
public NoSuchMethodError()
```

NoSuchMethodError

```
public NoSuchMethodError(String s)
```

Constructs a NoSuchMethodException with a detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class java.lang.NoSuchMethodException

Class java.lang.NoSuchMethodException

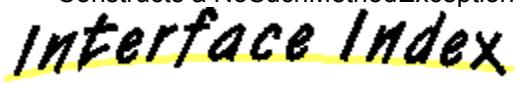
```
java.lang.Object
    |
    +----java.lang.Throwable
             |
             +----java.lang.Exception
                      |
                      +----java.lang.NoSuchMethodException
```

---

public class **NoSuchMethodException**
extends Exception

Signals that a particular method could not be found.

NoSuchMethodException()
Constructs a NoSuchMethodException without a detail message.



NoSuchMethodException(String)
Constructs a NoSuchMethodException with a detail message.

NoSuchMethodException

```
public NoSuchMethodException()
```

Constructs a NoSuchMethodException without a detail message. A detail message is a String that describes this particular exception.

NoSuchMethodException

```
public NoSuchMethodException(String s)
```

Constructs a NoSuchMethodException with a detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class java.lang.NullPointerException

Class java.lang.NullPointerException

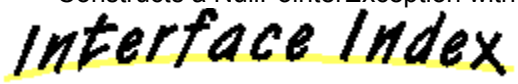
```
java.lang.Object
    |
    +----java.lang.Throwable
            |
            +----java.lang.Exception
                    |
                    +----java.lang.RuntimeException
                            |
                            +----java.lang.NullPointerException
```

---

public class **NullPointerException**
extends RuntimeException

Signals the illegal use of a null pointer.

NullPointerException()
Constructs a NullPointerException with no detail message.



NullPointerException(String)
Constructs a NullPointerException with the specified detail message.

NullPointerException

```
public NullPointerException()
```

Constructs a NullPointerException with no detail message. A detail message is a String that describes this particular exception.

```
public NullPointerException(String s)
```

Constructs a NullPointerException with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class java.lang.Number

Class java.lang.Number

```
java.lang.Object
    |
    +----java.lang.Number
```

---

public class **Number**
extends Object

Number is an abstract superclass for numeric scalar types. Integer, Long, Float and Double are subclasses of Number that bind to a particular numeric representation.

See Also:
Integer, Long, Float, Double

Number()

doubleValue()
Returns the value of the number as a double.

floatValue()
Returns the value of the number as a float.

intValue()
Returns the value of the number as an int.

longValue()

Returns the value of the number as a long.

Interface Index

Interface Index **Number**

```
public Number()
```

Interface Index

Interface Index **intValue**

```
public abstract int intValue()
```

Returns the value of the number as an int. This may involve rounding if the number is not already an integer.

Interface Index **longValue**

```
public abstract long longValue()
```

Returns the value of the number as a long. This may involve rounding if the number is not already a long.

Interface Index **floatValue**

```
public abstract float floatValue()
```

Returns the value of the number as a float. This may involve rounding if the number is not already a float.

Interface Index **doubleValue**

```
public abstract double doubleValue()
```

Returns the value of the number as a double. This may involve rounding if the number is not already a double.

Class java.lang.NumberFormatException

Class java.lang.NumberFormatException

```
java.lang.Object
    |
    +----java.lang.Throwable
            |
            +----java.lang.Exception
                    |
                    +----java.lang.RuntimeException
                            |
                            +----java.lang.IllegalArgumentException
                                    |
                                    +----java.lang.NumberFormatException
```

---

public class **NumberFormatException**
extends IllegalArgumentException

Signals that an invalid number format has occurred.
See Also:
toString

Interface Index

Interface Index

NumberFormatException()
Constructs a NumberFormatException with no detail message.

Interface Index

NumberFormatException(String)
Constructs a NumberFormatException with the specified detail message.

Interface Index

NumberFormatException

```
public NumberFormatException()
```

Constructs a NumberFormatException with no detail message. A detail message is a String that describes this particular exception.

NumberFormatException

```
public NumberFormatException(String s)
```

Constructs a NumberFormatException with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class java.lang.Object

Class java.lang.Object

java.lang.Object

---

public class **Object**

The root of the Class hierarchy. Every Class in the system has Object as its ultimate parent. Every variable and method defined here is available in every Object.

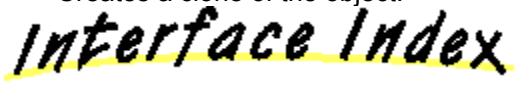
See Also:

Class

Object()

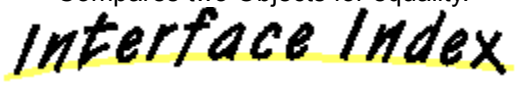
clone()

Creates a clone of the object.



equals(Object)

Compares two Objects for equality.



finalize()

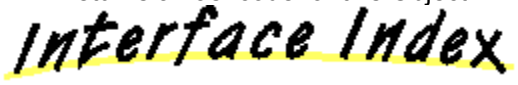
Code to perform when this object is garbage collected.

getClass()

Returns the Class of this Object.

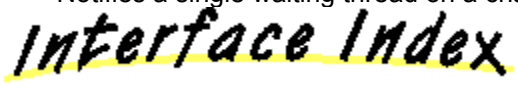
hashCode()

Returns a hashcode for this Object.



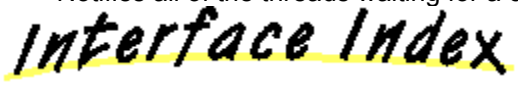
notify()

Notifies a single waiting thread on a change in condition of another thread.



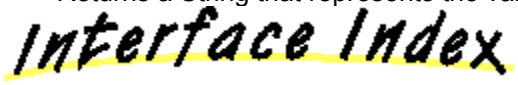
notifyAll()

Notifies all of the threads waiting for a condition to change.



toString()

Returns a String that represents the value of this Object.



wait(long)

Causes a thread to wait until it is notified or the specified timeout expires.

wait(long, int)

More accurate wait.

wait()

Causes a thread to wait forever until it is notified.

Object

```
public Object()
```

getClass

```
public final Class getClass()
```

Returns the Class of this Object. Java has a runtime representation for classes- a descriptor of type Class- which the method getClass() returns for any Object.

hashCode

```
public int hashCode()
```

Returns a hashcode for this Object. Each Object in the Java system has a hashcode. The hashcode

is a number that is usually different for different Objects. It is used when storing Objects in hashtables. Note: hashcodes can be negative as well as positive.

See Also:

Hashtable

equals

```
public boolean equals(Object obj)
```

Compares two Objects for equality. Returns a boolean that indicates whether this Object is equivalent to the specified Object. This method is used when an Object is stored in a hashtable.

Parameters:

obj - the Object to compare with

Returns:

true if these Objects are equal; false otherwise.

See Also:

Hashtable

clone

```
protected Object clone() throws CloneNotSupportedException
```

Creates a clone of the object. A new instance is allocated and a bitwise clone of the current object is place in the new object.

Returns:

a clone of this Object.

Throws: OutOfMemoryError

If there is not enough memory.

Throws: CloneNotSupportedException

Object explicitly does not want to be cloned, or it does not support the Cloneable interface.

toString

```
public String toString()
```

Returns a String that represents the value of this Object. It is recommended that all subclasses override this method.

notify

```
public final void notify()
```

Notifies a single waiting thread on a change in condition of another thread. The thread effecting the change notifies the waiting thread using notify(). Threads that want to wait for a condition to change before proceeding can call wait().

The method notify() can only be called from within a synchronized method.
Throws: IllegalMonitorStateException
If the current thread is not the owner of the Object's monitor.
See Also:
wait, notifyAll

notifyAll

```
public final void notifyAll()
```

Notifies all of the threads waiting for a condition to change. Threads that are waiting are generally waiting for another thread to change some condition. Thus, the thread effecting a change that more than one thread is waiting for notifies all the waiting threads using the method notifyAll(). Threads that want to wait for a condition to change before proceeding can call wait().

The method notifyAll() can only be called from within a synchronized method.
Throws: IllegalMonitorStateException
If the current thread is not the owner of the Object's monitor.
See Also:
wait, notify

wait

```
public final void wait(long timeout) throws InterruptedException
```

Causes a thread to wait until it is notified or the specified timeout expires.

The method wait() can only be called from within a synchronized method.
Parameters:
timeout - the maximum time to wait in milliseconds
Throws: IllegalMonitorStateException
If the current thread is not the owner of the Object's monitor.
Throws: InterruptedException
Another thread has interrupted this thread.

wait

```
public final void wait(long timeout,
                       int nanos) throws InterruptedException
```

More accurate wait. *The method wait() can only be called from within a synchronized method.*

Parameters:

timeout - the maximum time to wait in milliseconds
nano - additional time, in nanoseconds range 0-999999

Throws: IllegalMonitorStateException

If the current thread is not the owner of the Object's monitor.

Throws: InterruptedException

Another thread has interrupted this thread.


```
public final void wait() throws InterruptedException
```

Causes a thread to wait forever until it is notified.

The method wait() can only be called from within a synchronized method

Throws: IllegalMonitorStateException

If the current thread is not the owner of the Object's monitor.

Throws: InterruptedException

Another thread has interrupted this thread.


```
protected void finalize() throws Throwable
```

Code to perform when this object is garbage collected. The default is that nothing needs to be performed. Any exception thrown by a finalize method causes the finalization to halt. But otherwise, it is ignored.

Class java.lang.OutOfMemoryError

Class java.lang.OutOfMemoryError

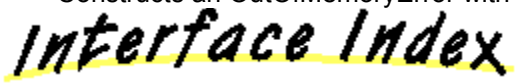
```
java.lang.Object
    |
    +----java.lang.Throwable
            |
            +----java.lang.Error
                    |
                    +----java.lang.VirtualMachineError
                            |
                            +----java.lang.OutOfMemoryError
```

---

public class **OutOfMemoryError**
extends VirtualMachineError

Signals that you are out of memory.

OutOfMemoryError()
Constructs an OutOfMemoryError with no detail message.



OutOfMemoryError(String)
Constructs an OutOfMemoryError with the specified detail message.

OutOfMemoryError

```
public OutOfMemoryError()
```

Constructs an OutOfMemoryError with no detail message. A detail message is a String that describes this particular exception.

```
public OutOfMemoryError(String s)
```

Constructs an OutOfMemoryError with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class java.lang.Process

Class java.lang.Process

```
java.lang.Object
    |
    +----java.lang.Process
```

---

public class **Process**
extends Object

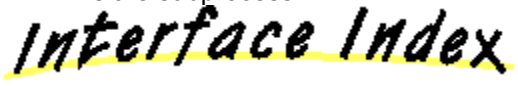
An instance of class Process is returned by variants of the exec () method in class System. From the Process instance, it is possible to: get the standin and/or standout of the subprocess, kill the subprocess, wait for it to terminate, and to retrieve the final exit value of the process.

Dropping the last reference to a Process instance does not kill the subprocess. There is no requirement that the subprocess execute asynchronously with the existing Java process.

Process()

destroy()

Kills the subprocess.

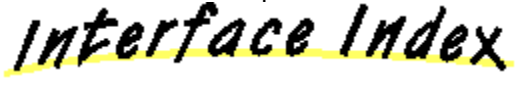


exitValue()

Returns the exit value for the subprocess.

getErrorStream()

Returns the an InputStream connected to the error stream of the child process.



getInputStream()

Returns a Stream connected to the output of the child process.

Interface Index **getOutputStream**()

Returns a Stream connected to the input of the child process.

Interface Index **waitFor**()

Waits for the subprocess to complete.

Interface Index

Interface Index **Process**

```
public Process()
```

Interface Index

Interface Index **getOutputStream**

```
public abstract OutputStream getOutputStream()
```

Returns a Stream connected to the input of the child process. This stream is traditionally buffered.

Interface Index **getInputStream**

```
public abstract InputStream getInputStream()
```

Returns a Stream connected to the output of the child process. This stream is traditionally buffered.

Interface Index **getErrorStream**

```
public abstract InputStream getErrorStream()
```

Returns the an InputStream connected to the error stream of the child process. This stream is traditionally unbuffered.

Interface Index **waitFor**

```
public abstract int waitFor() throws InterruptedException
```

Waits for the subprocess to complete. If the subprocess has already terminated, the exit value is simply returned. If the subprocess has not yet terminated the calling thread will be blocked until the subprocess exits.
Throws: InterruptedException
Another thread has interrupted this thread.

exitValue

```
public abstract int exitValue()
```

Returns the exit value for the subprocess.
Throws: IllegalThreadStateException
If the subprocess has not yet terminated.

destroy

```
public abstract void destroy()
```

Kills the subprocess.

Interface java.lang.Runnable

Interface java.lang.Runnable

public interface **Runnable**
extends Object

This interface is designed to provide a common protocol for Objects that wish to execute code while they are active. For example, Runnable is implemented by class Thread. Being active simply means that a thread has been started and has not yet been stopped.

In addition, Runnable provides the means for a class to be active while not subclassing Thread. A class that implements Runnable can run without subclassing Thread by instantiating a Thread instance and passing itself in as the target. In most cases, the Runnable interface should be used if you are only planning to override the run() method and no other Thread methods. This is important because classes should not be subclassed unless the programmer intends on modifying or enhancing the fundamental behavior of the class.

See Also:

Thread

Interface Index

Interface Index

run()

The method that is executed when a Runnable object is activated.

Interface Index

Interface Index

run

```
public abstract void run()
```

The method that is executed when a Runnable object is activated. The run() method is the "soul" of a Thread. It is in this method that all of the action of a Thread takes place.

See Also:

run

Class java.lang.Runtime

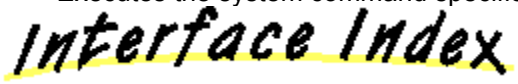
Class java.lang.Runtime

```
java.lang.Object
    |
    +----java.lang.Runtime
```

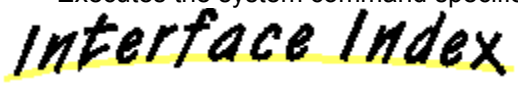
---

public class **Runtime**
extends Object

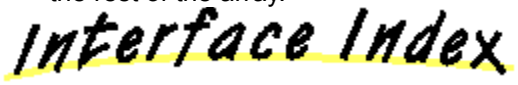
exec(String)
Executes the system command specified in the parameter.



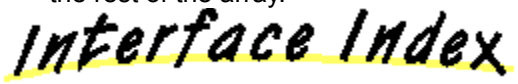
exec(String, String[])
Executes the system command specified in the parameter.



exec(String[])
Executes the system command specified by cmdarray[0] with arguments specified by the strings in the rest of the array.

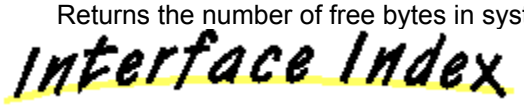


exec(String[], String[])
Executes the system command specified by cmdarray[0] with arguments specified by the strings in the rest of the array.



exit(int)
Exits the virtual machine with an exit code.

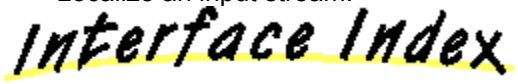
freeMemory()
Returns the number of free bytes in system memory.



gc()
Runs the garbage collector.

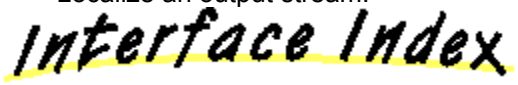
getLocalizedInputStream(InputStream)

Localize an input stream.



getLocalizedOutputStream(OutputStream)

Localize an output stream.

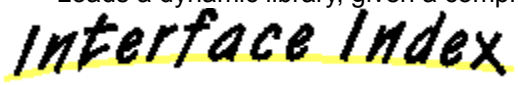


getRuntime()

Returns the runtime.

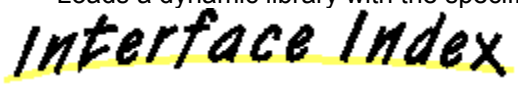
load(String)

Loads a dynamic library, given a complete path name.



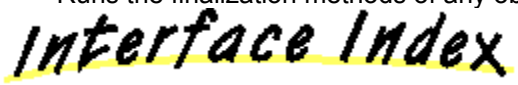
loadLibrary(String)

Loads a dynamic library with the specified library name.



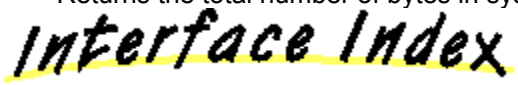
runFinalization()

Runs the finalization methods of any objects pending finalization.



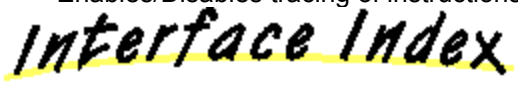
totalMemory()

Returns the total number of bytes in system memory.



traceInstructions(boolean)

Enables/Disables tracing of instructions.



traceMethodCalls(boolean)

Enables/Disables tracing of method calls.

getRuntime

```
public static Runtime getRuntime()
```

Returns the runtime.

exit

```
public void exit(int status)
```

Exits the virtual machine with an exit code. This method does not return, use with caution.

Parameters:

status - exit status, 0 if successful, other values indicate various error types.

exec

```
public Process exec(String command) throws IOException
```

Executes the system command specified in the parameter. Returns a Process which has methods for optaining the stdin, stdout, and stderr of the subprocess. This method fails if executed by untrusted code.

Parameters:

command - a specified system command

Returns:

an instance of class Process

exec

```
public Process exec(String command,
                    String envp[]) throws IOException
```

Executes the system command specified in the parameter. Returns a Process which has methods for optaining the stdin, stdout, and stderr of the subprocess. This method fails if executed by untrusted code.

Parameters:

command - a specified system command

Returns:

an instance of class Process

exec

```
public Process exec(String cmdarray[]) throws IOException
```

Executes the system command specified by cmdarray[0] with arguments specified by the strings in the rest of the array. Returns a Process which has methods for optaining the stdin, stdout, and stderr of the subprocess. This method fails if executed by untrusted code.

Parameters:

an - array containing the command to call and its arguments

envp - array containing environment in format name=value

Returns:

an instance of class Process

exec

```
public Process exec(String cmdarray[],
                    String envp[]) throws IOException
```

Executes the system command specified by cmdarray[0] with arguments specified by the strings in the rest of the array. Returns a Process which has methods for optaining the stdin, stdout, and stderr of the subprocess. This method fails if executed by untrusted code.

Parameters:

an - array containing the command to call and its arguments

envp - array containing environment in format name=value

Returns:

an instance of class Process

freeMemory

```
public long freeMemory()
```

Returns the number of free bytes in system memory. This number is not always accurate because it is just an estimation of the available memory. More memory may be freed by calling System.gc() .

totalMemory

```
public long totalMemory()
```

Returns the total number of bytes in system memory.

gc

```
public void gc()
```

Runs the garbage collector.

runFinalization

```
public void runFinalization()
```

Runs the finalization methods of any objects pending finalization. Usually you will not need to call this method since finalization methods will be called asynchronously by the finalization thread. However, under some circumstances (like running out of a finalized resource) it can be useful to run finalization methods synchronously.

traceInstructions

```
public void traceInstructions(boolean on)
```

Enables/Disables tracing of instructions.
Parameters:
on - start tracing if true

traceMethodCalls

```
public void traceMethodCalls(boolean on)
```

Enables/Disables tracing of method calls.
Parameters:
on - start tracing if true

load

```
public synchronized void load(String filename)
```

Loads a dynamic library, given a complete path name. If you use this from java_g it will automagically insert "_g" before the ".so". Example:
`Runtime.getRuntime().load("/home/avh/lib/libX11.so");`
Parameters:
filename - the file to load
Throws: UnsatisfiedLinkError
If the file does not exist.
See Also:
getRuntime

loadLibrary

```
public synchronized void loadLibrary(String libname)
```

Loads a dynamic library with the specified library name. The call to LoadLibrary() should be made in the static initializer of the first class that is loaded. Linking in the same library more than once is ignored.
Parameters:

libname - the name of the library

Throws: UnsatisfiedLinkError

If the library does not exist.

getLocalizedInputStream

```
public InputStream getLocalizedInputStream(InputStream in)
```

Localize an input stream. A localized input stream will automatically translate the input from the local format to UNICODE.

getLocalizedOutputStream

```
public OutputStream getLocalizedOutputStream(OutputStream out)
```

Localize an output stream. A localized output stream will automatically translate the output from UNICODE to the local format.

Class java.lang.RuntimeException

Class java.lang.RuntimeException

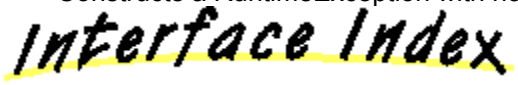
```
java.lang.Object
    |
    +----java.lang.Throwable
            |
            +----java.lang.Exception
                    |
                    +----java.lang.RuntimeException
```

---

public class **RuntimeException**
extends Exception

An exception that can reasonably occur during the execution of a Java program by the Virtual machine.

RuntimeException()
Constructs a RuntimeException with no detail message.



RuntimeException(String)
Constructs a RuntimeException with the specified detail message.

RuntimeException

```
public RuntimeException()
```

Constructs a RuntimeException with no detail message. A detail message is a String that describes this particular exception.

RuntimeException

```
public RuntimeException(String s)
```

Constructs a RuntimeException with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class java.lang.SecurityException

Class java.lang.SecurityException

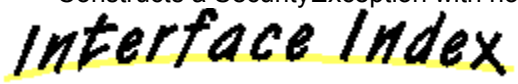
```
java.lang.Object
    |
    +----java.lang.Throwable
            |
            +----java.lang.Exception
                    |
                    +----java.lang.RuntimeException
                            |
                            +----java.lang.SecurityException
```

---

public class **SecurityException**
extends RuntimeException

Signals that a security exception has occurred.

SecurityException()
Constructs a SecurityException with no detail message.



SecurityException(String)
Constructs a SecurityException with the specified detail message.

SecurityException

```
public SecurityException()
```

Constructs a SecurityException with no detail message. A detail message is a String that describes this particular exception.

```
public SecurityException(String s)
```

Constructs a SecurityException with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class java.lang.SecurityManager

Class java.lang.SecurityManager

```
java.lang.Object
    |
    +----java.lang.SecurityManager
```

---

public class **SecurityManager**
extends Object

An abstract class that can be subclassed to implement a security policy. It allows the inspection of the classloaders on the execution stack.

inCheck

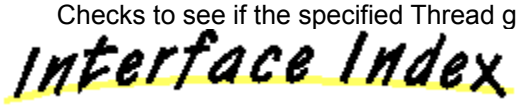
SecurityManager()
Constructs a new SecurityManager.

checkAccept(String, int)
Checks to see if a socket connection to the specified port on the specified host has been accepted.

checkAccess(Thread)
Checks to see if the specified Thread is allowed to modify the Thread group.

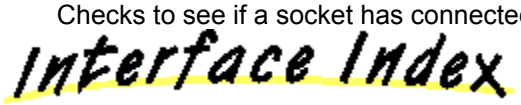
checkAccess(ThreadGroup)

Checks to see if the specified Thread group is allowed to modify this group.



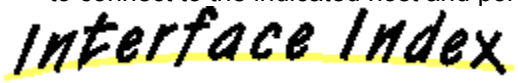
checkConnect(String, int)

Checks to see if a socket has connected to the specified port on the the specified host.



checkConnect(String, int, Object)

Checks to see if the current execution context and the indicated execution context are both allowed to connect to the indicated host and port.

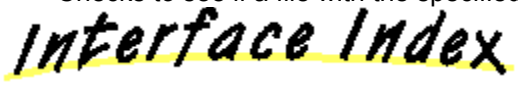


checkCreateClassLoader()

Checks to see if the ClassLoader has been created.

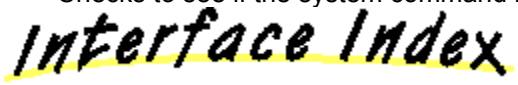
checkDelete(String)

Checks to see if a file with the specified system dependent file name can be deleted.



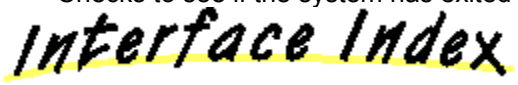
checkExec(String)

Checks to see if the system command is executed by trusted code.



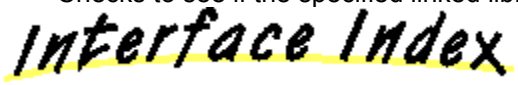
checkExit(int)

Checks to see if the system has exited the virtual machine with an exit code.



checkLink(String)

Checks to see if the specified linked library exists.

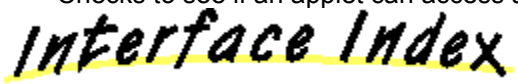


checkListen(int)

Checks to see if a server socket is listening to the specified local port that it is bounded to.

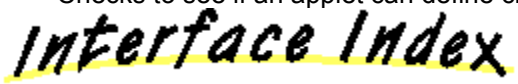
checkPackageAccess(String)

Checks to see if an applet can access a package.



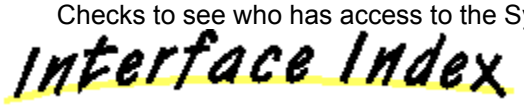
checkPackageDefinition(String)

Checks to see if an applet can define classes in a package.



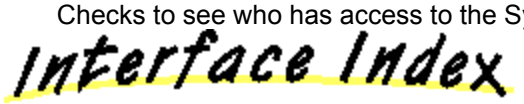
checkPropertiesAccess()

Checks to see who has access to the System properties.



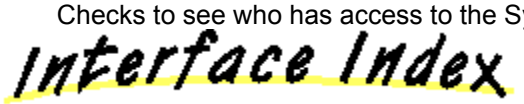
checkPropertyAccess(String)

Checks to see who has access to the System property named by *key*.



checkPropertyAccess(String, String)

Checks to see who has access to the System property named by *key* and *def*.

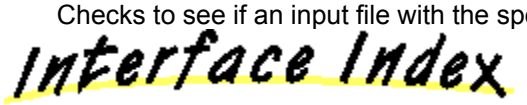


checkRead(FileDescriptor)

Checks to see if an input file with the specified file descriptor object gets created.

checkRead(String)

Checks to see if an input file with the specified system dependent file name gets created.

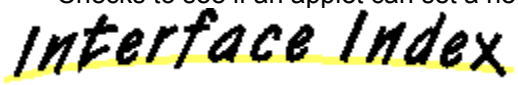


checkRead(String, Object)

Checks to see if the current context or the indicated context are both allowed to read the given file name.

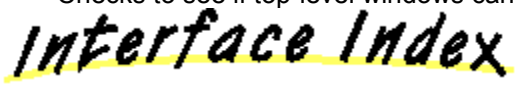
checkSetFactory()

Checks to see if an applet can set a networking-related object factory.



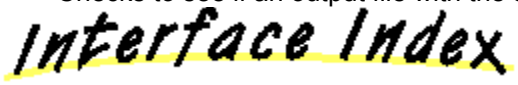
checkTopLevelWindow(Object)

Checks to see if top-level windows can be created by the caller.



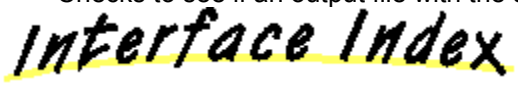
checkWrite(FileDescriptor)

Checks to see if an output file with the specified file descriptor object gets created.



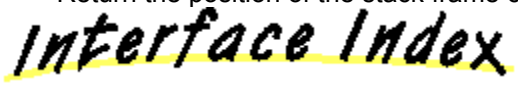
checkWrite(String)

Checks to see if an output file with the specified system dependent file name gets created.



classDepth(String)

Return the position of the stack frame containing the first occurrence of the named class.



classLoaderDepth()

currentClassLoader()

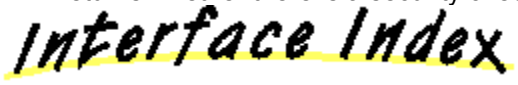
The current ClassLoader on the execution stack.

getClassContext()

Gets the context of this Class.

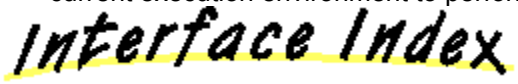
getInCheck()

Returns whether there is a security check in progress.



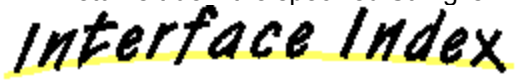
getSecurityContext()

Returns an implementation-dependent Object which encapsulates enough information about the current execution environment to perform some of the security checks later.



inClass(String)

Returns true if the specified String is in this Class.



inClassLoader()

Returns a boolean indicating whether or not the current ClassLoader is equal to null.

Interface Index

inCheck

```
protected boolean inCheck
```

Interface Index

Interface Index

SecurityManager

```
protected SecurityManager()
```

Constructs a new SecurityManager.
Throws: SecurityException
If the security manager cannot be created.

Interface Index

Interface Index

getInCheck

```
public boolean getInCheck()
```

Returns whether there is a security check in progress.

Interface Index

getClassContext

```
protected Class[] getClassContext()
```

Gets the context of this Class.

Interface Index

currentClassLoader

```
protected ClassLoader currentClassLoader()
```

The current ClassLoader on the execution stack.

classDepth

```
protected int classDepth(String name)
```

Return the position of the stack frame containing the first occurrence of the named class.
Parameters:
name - classname of the class to search for

classLoaderDepth

```
protected int classLoaderDepth()
```

inClass

```
protected boolean inClass(String name)
```

Returns true if the specified String is in this Class.
Parameters:
name - the name of the class

inClassLoader

```
protected boolean inClassLoader()
```

Returns a boolean indicating whether or not the current ClassLoader is equal to null.

getSecurityContext

```
public Object getSecurityContext()
```

Returns an implementation-dependent Object which encapsulates enough information about the current execution environment to perform some of the security checks later.

checkCreateClassLoader

```
public void checkCreateClassLoader()
```

Checks to see if the ClassLoader has been created.
Throws: SecurityException
If a security error has occurred.

checkAccess

```
public void checkAccess(Thread g)
```

Checks to see if the specified Thread is allowed to modify the Thread group.
Parameters:
g - the Thread to be checked
Throws: SecurityException
If the current Thread is not allowed to access this Thread group.

checkAccess

```
public void checkAccess(ThreadGroup g)
```

Checks to see if the specified Thread group is allowed to modify this group.
Parameters:
g - the Thread group to be checked
Throws: SecurityException
If the current Thread group is not allowed to access this Thread group.

checkExit

```
public void checkExit(int status)
```

Checks to see if the system has exited the virtual machine with an exit code.
Parameters:
status - exit status, 0 if successful, other values indicate various error types.
Throws: SecurityException
If a security error has occurred.

Interface Index **checkExec**

```
public void checkExec(String cmd)
```

Checks to see if the system command is executed by trusted code.
Parameters:
cmd - the specified system command
Throws: SecurityException
If a security error has occurred.

Interface Index **checkLink**

```
public void checkLink(String lib)
```

Checks to see if the specified linked library exists.
Parameters:
lib - the name of the library
Throws: SecurityException
If the library does not exist.

Interface Index **checkRead**

```
public void checkRead(FileDescriptor fd)
```

Checks to see if an input file with the specified file descriptor object gets created.
Parameters:
fd - the system dependent file descriptor
Throws: SecurityException
If a security error has occurred.

Interface Index **checkRead**

```
public void checkRead(String file)
```

Checks to see if an input file with the specified system dependent file name gets created.
Parameters:
file - the system dependent file name
Throws: SecurityException
If the file is not found.

 checkRead

```
public void checkRead(String file,
                      Object context)
```

Checks to see if the current context or the indicated context are both allowed to read the given file name.

Parameters:

file - the system dependent file name

context - the alternate execution context which must also be checked

Throws: SecurityException

If the file is not found.

 checkWrite

```
public void checkWrite(FileDescriptor fd)
```

Checks to see if an output file with the specified file descriptor object gets created.

Parameters:

fd - the system dependent file descriptor

Throws: SecurityException

If a security error has occurred.

 checkWrite

```
public void checkWrite(String file)
```

Checks to see if an output file with the specified system dependent file name gets created.

Parameters:

file - the system dependent file name

Throws: SecurityException

If the file is not found.

 checkDelete

```
public void checkDelete(String file)
```

Checks to see if a file with the specified system dependent file name can be deleted.

Parameters:

file - the system dependent file name

Throws: SecurityException

If the file is not found.

checkConnect

```
public void checkConnect(String host,
                         int port)
```

Checks to see if a socket has connected to the specified port on the the specified host.

Parameters:

host - the host name port to connect to

port - the protocol port to connect to

Throws: SecurityException

If a security error has occurred.

checkConnect

```
public void checkConnect(String host,
                         int port,
                         Object context)
```

Checks to see if the current execution context and the indicated execution context are both allowed to connect to the indicated host and port.

checkListen

```
public void checkListen(int port)
```

Checks to see if a server socket is listening to the specified local port that it is bounded to.

Parameters:

port - the protocol port to connect to

Throws: SecurityException

If a security error has occurred.

checkAccept

```
public void checkAccept(String host,
                        int port)
```

Checks to see if a socket connection to the specified port on the specified host has been accepted.

Parameters:

host - the host name to connect to

port - the protocol port to connect to

Throws: SecurityException

If a security error has occurred.

checkPropertiesAccess

```
public void checkPropertiesAccess()
```

Checks to see who has access to the System properties.

Throws: SecurityException

If a security error has occurred.

checkPropertyAccess

```
public void checkPropertyAccess(String key)
```

Checks to see who has access to the System property named by *key*.

Parameters:

key - the System property that the caller wants to examine

Throws: SecurityException

If a security error has occurred.

checkPropertyAccess

```
public void checkPropertyAccess(String key,
                                String def)
```

Checks to see who has access to the System property named by *key* and *def*.

Parameters:

key - the System property that the caller wants to examine

def - default value to return if this property is not defined

Throws: SecurityException

If a security error has occurred.

checkTopLevelWindow

```
public boolean checkTopLevelWindow(Object window)
```

Checks to see if top-level windows can be created by the caller. A return of false means that the window creation is allowed but the window should indicate some sort of visual warning. Returning true means the creation is allowed with no special restrictions. To disallow the creation entirely, this method should throw a SecurityException.

Parameters:

window - the new window that's being created.

checkPackageAccess

```
public void checkPackageAccess(String pkg)
```

Checks to see if an applet can access a package.

checkPackageDefinition

```
public void checkPackageDefinition(String pkg)
```

Checks to see if an applet can define classes in a package.

checkSetFactory

```
public void checkSetFactory()
```

Checks to see if an applet can set a networking-related object factory.

Class java.lang.StackOverflowError

Class java.lang.StackOverflowError

```
java.lang.Object
    |
    +----java.lang.Throwable
            |
            +----java.lang.Error
                    |
                    +----java.lang.VirtualMachineError
                            |
                            +----java.lang.StackOverflowError
```

---

public class **StackOverflowError**
extends VirtualMachineError

Signals that a stack overflow has occurred.

StackOverflowError()
Constructs a StackOverflowError with no detail message.

StackOverflowError(String)
Constructs a StackOverflowError with the specified detail message.

StackOverflowError

```
public StackOverflowError()
```

Constructs a StackOverflowError with no detail message. A detail message is a String that describes this particular exception.

```
public StackOverflowError(String s)
```

Constructs a StackOverflowError with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class java.lang.String

Class java.lang.String

```
java.lang.Object
   |
   +----java.lang.String
```

---

public final class **String**
extends Object

A general class of objects to represent character Strings. Strings are constant, their values cannot be changed after creation. The compiler makes sure that each String constant actually results in a String object. Because String objects are immutable they can be shared. For example:

```
String str = "abc";
```

is equivalent to:

```
char data[] = {'a', 'b', 'c'};
String str = new String(data);
```

Here are some more examples of how strings can be used:

```
System.out.println("abc");
String cde = "cde";
System.out.println("abc" + cde);
String c = "abc".substring(2,3);
String d = cde.substring(1, 2);
```

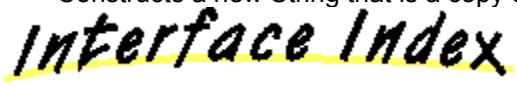
See Also:
StringBuffer

String()

Constructs a new empty String.

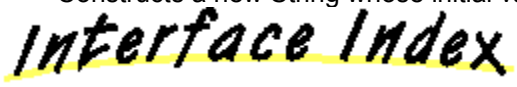
String(String)

Constructs a new String that is a copy of the specified String.



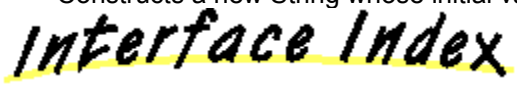
String(char[])

Constructs a new String whose initial value is the specified array of characters.



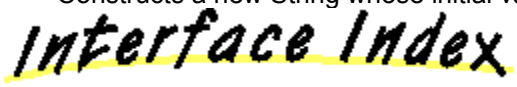
String(char[], int, int)

Constructs a new String whose initial value is the specified sub array of characters.



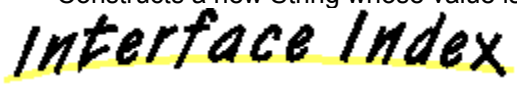
String(byte[], int, int, int)

Constructs a new String whose initial value is the specified sub array of bytes.



String(byte[], int)

Constructs a new String whose value is the specified array of bytes.

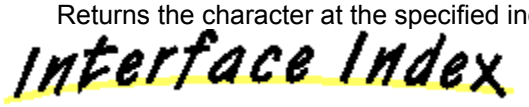


String(StringBuffer)

Construct a new string whose value is the current contents of the given string buffer

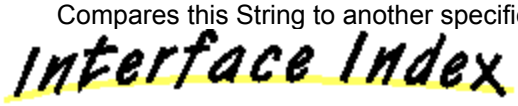
charAt(int)

Returns the character at the specified index.



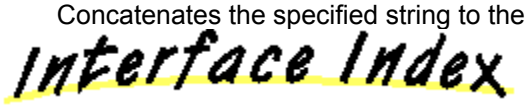
compareTo(String)

Compares this String to another specified String.



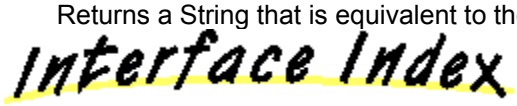
concat(String)

Concatenates the specified string to the end of this String.



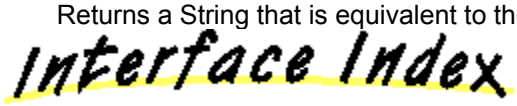
copyValueOf(char[], int, int)

Returns a String that is equivalent to the specified character array.



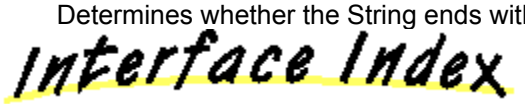
copyValueOf(char[])

Returns a String that is equivalent to the specified character array.



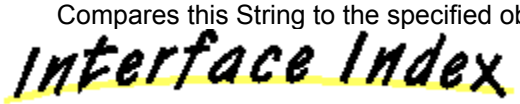
endsWith(String)

Determines whether the String ends with some suffix.



equals(Object)

Compares this String to the specified object.

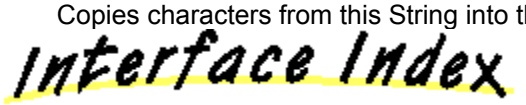


equalsIgnoreCase(String)

Compares this String to another object.

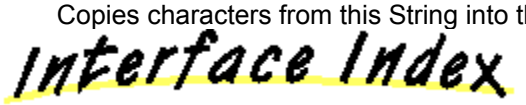
getBytes(int, int, byte[], int)

Copies characters from this String into the specified byte array.



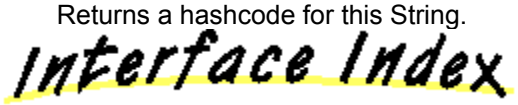
getChars(int, int, char[], int)

Copies characters from this String into the specified character array.



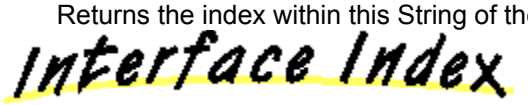
hashCode()

Returns a hashcode for this String.



indexOf(int)

Returns the index within this String of the first occurrence of the specified character.

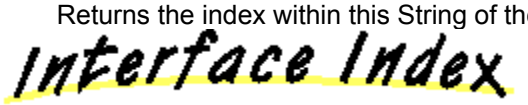


indexOf(int, int)

Returns the index within this String of the first occurrence of the specified character, starting the search at fromIndex.

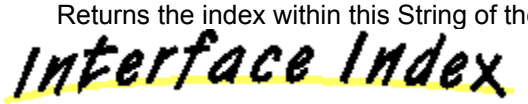
indexOf(String)

Returns the index within this String of the first occurrence of the specified substring.



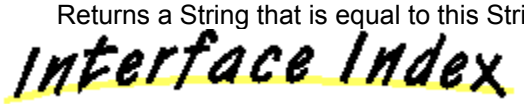
indexOf(String, int)

Returns the index within this String of the first occurrence of the specified substring.



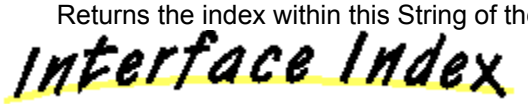
intern()

Returns a String that is equal to this String but which is guaranteed to be from the unique String pool.



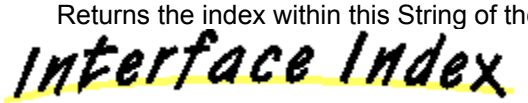
lastIndexOf(int)

Returns the index within this String of the last occurrence of the specified character.



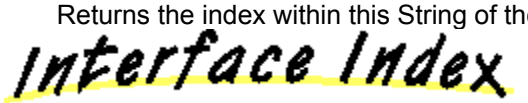
lastIndexOf(int, int)

Returns the index within this String of the last occurrence of the specified character.



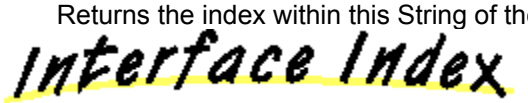
lastIndexOf(String)

Returns the index within this String of the last occurrence of the specified substring.



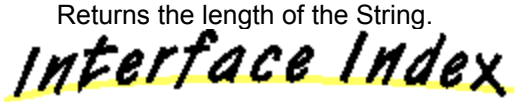
lastIndexOf(String, int)

Returns the index within this String of the last occurrence of the specified substring.



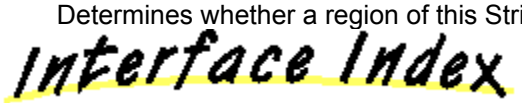
length()

Returns the length of the String.



regionMatches(int, String, int, int)

Determines whether a region of this String matches the specified region of the specified String.

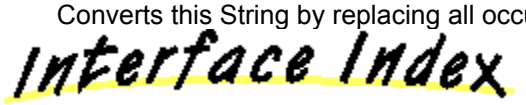


regionMatches(boolean, int, String, int, int)

Determines whether a region of this String matches the specified region of the specified String.

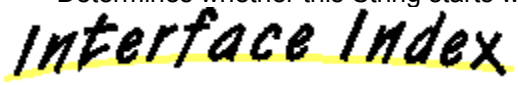
replace(char, char)

Converts this String by replacing all occurences of oldChar with newChar.



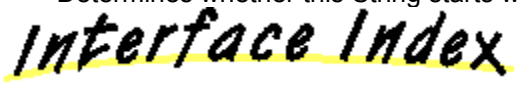
startsWith(String, int)

Determines whether this String starts with some prefix.



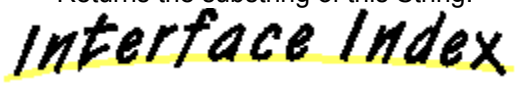
startsWith(String)

Determines whether this String starts with some prefix.



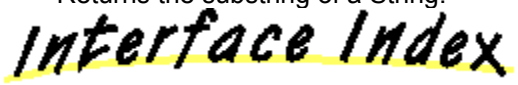
substring(int)

Returns the substring of this String.



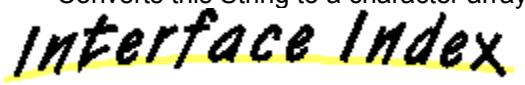
substring(int, int)

Returns the substring of a String.



toCharArray()

Converts this String to a character array.

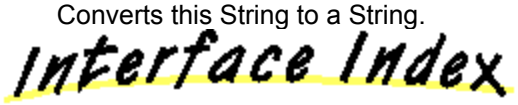


toLowerCase()

Converts all of the characters in this String to lower case.

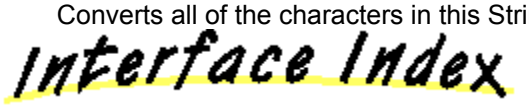
toString()

Converts this String to a String.



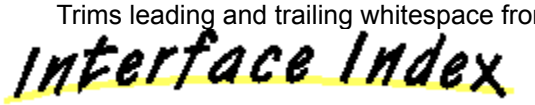
toUpperCase()

Converts all of the characters in this String to upper case.



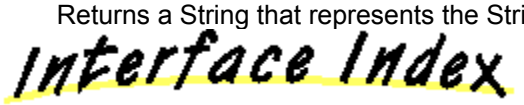
trim()

Trims leading and trailing whitespace from this String.



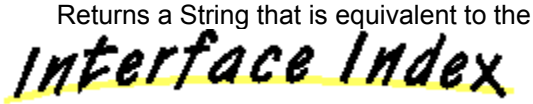
valueOf(Object)

Returns a String that represents the String value of the object.



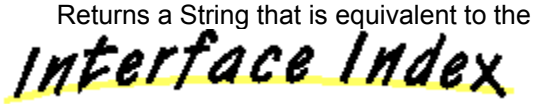
valueOf(char[])

Returns a String that is equivalent to the specified character array.



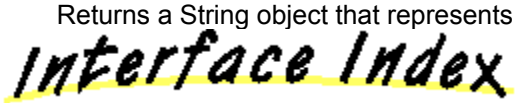
valueOf(char[], int, int)

Returns a String that is equivalent to the specified character array.



valueOf(boolean)

Returns a String object that represents the state of the specified boolean.

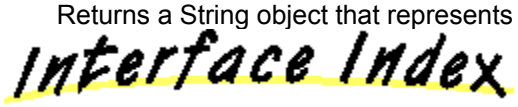


valueOf(char)

Returns a String object that contains a single character

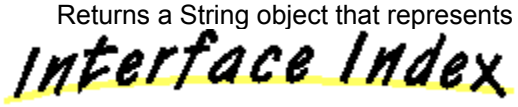
valueOf(int)

Returns a String object that represents the value of the specified integer.



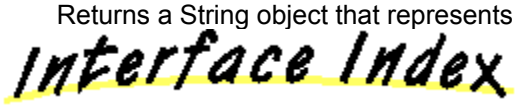
valueOf(long)

Returns a String object that represents the value of the specified long.



valueOf(float)

Returns a String object that represents the value of the specified float.



valueOf(double)

Returns a String object that represents the value of the specified double.

String

```
public String()
```

Constructs a new empty String.

String

```
public String(String value)
```

Constructs a new String that is a copy of the specified String.

Parameters:

value - the initial value of the String

String

```
public String(char value[])
```

Constructs a new String whose initial value is the specified array of characters.

Parameters:

value - the initial value of the String

```
public String(char value[],
              int offset,
              int count)
```

Constructs a new String whose initial value is the specified sub array of characters. The length of the new string will be count characters starting at offset within the specified character array.

Parameters:

value - the initial value of the String, an array of characters
offset - the offset into the value of the String
count - the length of the value of the String

Throws: StringIndexOutOfBoundsException

If the offset and count arguments are invalid.


```
public String(byte ascii[],
              int hibyte,
              int offset,
              int count)
```

Constructs a new String whose initial value is the specified sub array of bytes. The high-byte of each character can be specified, it should usually be 0. The length of the new String will be count characters starting at offset within the specified character array.

Parameters:

ascii - the bytes that will be converted to characters
hibyte - the high byte of each Unicode character
offset - the offset into the ascii array
count - the length of the String

Throws: StringIndexOutOfBoundsException

If the offset and count arguments are invalid.


```
public String(byte ascii[],
              int hibyte)
```

Constructs a new String whose value is the specified array of bytes. The byte array transformed into Unicode chars using hibyte as the upper byte of each character.

Parameters:

ascii - the byte that will be converted to characters
hibyte - the top 8 bits of each 16 bit Unicode character

String

```
public String(StringBuffer buffer)
```

Construct a new string whose value is the current contents of the given string buffer
Parameters:
buffer - the stringbuffer to be converted

length

```
public int length()
```

Returns the length of the String. The length of the String is equal to the number of 16 bit Unicode characters in the String.

charAt

```
public char charAt(int index)
```

Returns the character at the specified index. An index ranges from 0 to length() - 1.
Parameters:
index - the index of the desired character
Throws: StringIndexOutOfBoundsException
If the index is not in the range 0 to length()-1.

getChars

```
public void getChars(int srcBegin,
                     int srcEnd,
                     char dst[],
                     int dstBegin)
```

Copies characters from this String into the specified character array. The characters of the specified substring (determined by srcBegin and srcEnd) are copied into the character array, starting at the array's dstBegin location.

Parameters:
srcBegin - index of the first character in the string
srcEnd - end of the characters that are copied
dst - the destination array
dstBegin - the start offset in the destination array

getBytes

```
public void getBytes(int srcBegin,
                     int srcEnd,
                     byte dst[],
                     int dstBegin)
```

Copies characters from this String into the specified byte array. Copies the characters of the specified substring (determined by srcBegin and srcEnd) into the byte array, starting at the array's dstBegin location.

Parameters:
srcBegin - index of the first character in the String
srcEnd - end of the characters that are copied
dst - the destination array
dstBegin - the start offset in the destination array

equals

```
public boolean equals(Object anObject)
```

Compares this String to the specified object. Returns true if the object is equal to this String; that is, has the same length and the same characters in the same sequence.

Parameters:
anObject - the object to compare this String against

Returns:
true if the Strings are equal; false otherwise.

Overrides:
equals in class Object

equalsIgnoreCase

```
public boolean equalsIgnoreCase(String anotherString)
```

Compares this String to another object. Returns true if the object is equal to this String; that is, has the same length and the same characters in the same sequence. Upper case characters are folded to lower case before they are compared.

Parameters:
anotherString - the String to compare this String against
Returns:
true if the Strings are equal, ignoring case; false otherwise.

compareTo

```
public int compareTo(String anotherString)
```

Compares this String to another specified String. Returns an integer that is less than, equal to, or greater than zero. The integer's value depends on whether this String is less than, equal to, or greater than anotherString.
Parameters:
anotherString - the String to be compared

regionMatches

```
public boolean regionMatches(int toffset,
                             String other,
                             int ooffset,
                             int len)
```

Determines whether a region of this String matches the specified region of the specified String.
Parameters:
toffset - where to start looking in this String
other - the other String
ooffset - where to start looking in the other String
len - the number of characters to compare
Returns:
true if the region matches with the other; false otherwise.

regionMatches

```
public boolean regionMatches(boolean ignoreCase,
                             int toffset,
                             String other,
                             int ooffset,
                             int len)
```

Determines whether a region of this String matches the specified region of the specified String. If the boolean ignoreCase is true, upper case characters are considered equivalent to lower case letters.
Parameters:

ignoreCase - if true, case is ignored
toffset - where to start looking in this String
other - the other String
ooffset - where to start looking in the other String
len - the number of characters to compare

Returns:

true if the region matches with the other; false otherwise.

startsWith

```
public boolean startsWith(String prefix,
                          int toffset)
```

Determines whether this String starts with some prefix.

Parameters:

prefix - the prefix
toffset - where to begin looking in the the String

Returns:

true if the String starts with the specified prefix; false otherwise.

startsWith

```
public boolean startsWith(String prefix)
```

Determines whether this String starts with some prefix.

Parameters:

prefix - the prefix

Returns:

true if the String starts with the specified prefix; false otherwise.

endsWith

```
public boolean endsWith(String suffix)
```

Determines whether the String ends with some suffix.

Parameters:

suffix - the suffix

Returns:

true if the String ends with the specified suffix; false otherwise.

hashCode

```
public int hashCode()
```

Returns a hashcode for this String. This is a large number composed of the character values in the String.

Overrides:

hashCode in class Object

indexOf

```
public int indexOf(int ch)
```

Returns the index within this String of the first occurrence of the specified character. This method returns -1 if the index is not found.

Parameters:

ch - the character to search for

indexOf

```
public int indexOf(int ch,
                   int fromIndex)
```

Returns the index within this String of the first occurrence of the specified character, starting the search at fromIndex. This method returns -1 if the index is not found.

Parameters:

ch - the character to search for

fromIndex - the index to start the search from

lastIndexOf

```
public int lastIndexOf(int ch)
```

Returns the index within this String of the last occurrence of the specified character. The String is searched backwards starting at the last character. This method returns -1 if the index is not found.

Parameters:

ch - the character to search for

lastIndexOf

```
public int lastIndexOf(int ch,
                       int fromIndex)
```

Returns the index within this String of the last occurrence of the specified character. The String is searched backwards starting at fromIndex. This method returns -1 if the index is not found.

Parameters:

ch - the character to search for

fromIndex - the index to start the search from

indexOf

```
public int indexOf(String str)
```

Returns the index within this String of the first occurrence of the specified substring. This method returns -1 if the index is not found.

Parameters:

str - the substring to search for

indexOf

```
public int indexOf(String str,
                   int fromIndex)
```

Returns the index within this String of the first occurrence of the specified substring. The search is started at fromIndex. This method returns -1 if the index is not found.

Parameters:

str - the substring to search for

fromIndex - the index to start the search from

lastIndexOf

```
public int lastIndexOf(String str)
```

Returns the index within this String of the last occurrence of the specified substring. The String is searched backwards. This method returns -1 if the index is not found.

Parameters:

str - the substring to search for

lastIndexOf

```
public int lastIndexOf(String str,
                       int fromIndex)
```

Returns the index within this String of the last occurrence of the specified substring. The String is searched backwards starting at fromIndex. This method returns -1 if the index is not found.

Parameters:

str - the substring to search for
fromIndex - the index to start the search from

substring

```
public String substring(int beginIndex)
```

Returns the substring of this String. The substring is specified by a beginIndex (inclusive) and the end of the string.

Parameters:

beginIndex - the beginning index, inclusive

substring

```
public String substring(int beginIndex,
                        int endIndex)
```

Returns the substring of a String. The substring is specified by a beginIndex (inclusive) and an endIndex (exclusive).

Parameters:

beginIndex - the beginning index, inclusive
endIndex - the ending index, exclusive

Throws: StringIndexOutOfBoundsException

If the beginIndex or the endIndex is out of range.

concat

```
public String concat(String str)
```

Concatenates the specified string to the end of this String.

Parameters:

str - the String which is concatenated to the end of this String

replace

```
public String replace(char oldChar,
                      char newChar)
```

Converts this String by replacing all occurences of oldChar with newChar.

Parameters:

oldChar - the old character

newChar - the new character

toLowerCase

```
public String toLowerCase()
```

Converts all of the characters in this String to lower case.

Returns:

the String, converted to lowercase.

See Also:

toLowerCase, toUpperCase

toUpperCase

```
public String toUpperCase()
```

Converts all of the characters in this String to upper case.

Returns:

the String, converted to uppercase.

See Also:

toUpperCase, toLowerCase

trim

```
public String trim()
```

Trims leading and trailing whitespace from this String.
Returns:
the String, with whitespace removed.

toString

```
public String toString()
```

Converts this String to a String.
Returns:
the String itself.
Overrides:
toString in class Object

toCharArray

```
public char[] toCharArray()
```

Converts this String to a character array. This creates a new array.
Returns:
an array of characters.

valueOf

```
public static String valueOf(Object obj)
```

Returns a String that represents the String value of the object. The object may choose how to represent itself by implementing the toString() method.
Parameters:
obj - the object to be converted

valueOf

```
public static String valueOf(char data[])
```

Returns a String that is equivalent to the specified character array. Uses the original array as the body of the String (ie. it doe not copy it to a new array).

Parameters:

data - the character array

valueOf

```
public static String valueOf(char data[],
                             int offset,
                             int count)
```

Returns a String that is equivalent to the specified character array.

Parameters:

data - the character array
offset - the offset into the value of the String
count - the length of the value of the String

copyValueOf

```
public static String copyValueOf(char data[],
                                 int offset,
                                 int count)
```

Returns a String that is equivalent to the specified character array. It creates a new array and copies the characters into it.

Parameters:

data - the character array
offset - the offset into the value of the String
count - the length of the value of the String

copyValueOf

```
public static String copyValueOf(char data[])
```

Returns a String that is equivalent to the specified character array. It creates a new array and copies the characters into it.

Parameters:

data - the character array

```
public static String valueOf(boolean b)
```

Returns a String object that represents the state of the specified boolean.

Parameters:

b - the boolean


```
public static String valueOf(char c)
```

Returns a String object that contains a single character

Parameters:

c - the character

Returns:

the resulting String.


```
public static String valueOf(int i)
```

Returns a String object that represents the value of the specified integer.

Parameters:

i - the integer


```
public static String valueOf(long l)
```

Returns a String object that represents the value of the specified long.

Parameters:

l - the long

```
public static String valueOf(float f)
```

Returns a String object that represents the value of the specified float.
Parameters:
f - the float

valueOf

```
public static String valueOf(double d)
```

Returns a String object that represents the value of the specified double.
Parameters:
d - the double

intern

```
public String intern()
```

Returns a String that is equal to this String but which is guaranteed to be from the unique String pool.
For example:
`s1.intern() == s2.intern()  s1.equals(s2).`

Class java.lang.StringBuffer

Class java.lang.StringBuffer

```
java.lang.Object
    |
    +----java.lang.StringBuffer
```

---

public final class **StringBuffer**
extends Object

This Class is a growable buffer for characters. It is mainly used to create Strings. The compiler uses it to implement the "+" operator. For example:

```
"a" + 4 + "c"
```

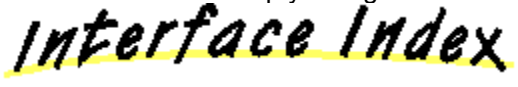
is compiled to:

```
new StringBuffer().append("a").append(4).append("c").toString()
```

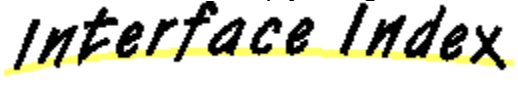
Note that the method toString() does not create a copy of the internal buffer. Instead the buffer is marked as shared. Any further changes to the buffer will cause a copy to be made.

See Also:
String, ByteArrayOutputStream

StringBuffer()
Constructs an empty String buffer.



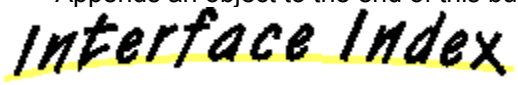
StringBuffer(int)
Constructs an empty String buffer with the specified initial length.



StringBuffer(String)
Constructs a String buffer with the specified initial value.

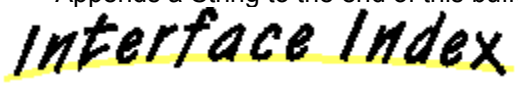
append(Object)

Appends an object to the end of this buffer.



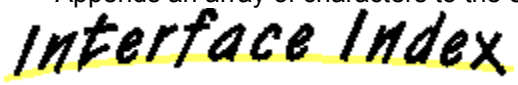
append(String)

Appends a String to the end of this buffer.



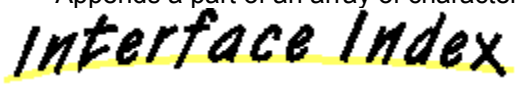
append(char[])

Appends an array of characters to the end of this buffer.



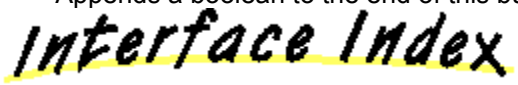
append(char[], int, int)

Appends a part of an array of characters to the end of this buffer.



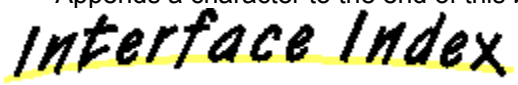
append(boolean)

Appends a boolean to the end of this buffer.



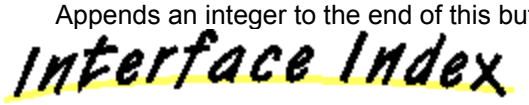
append(char)

Appends a character to the end of this buffer.



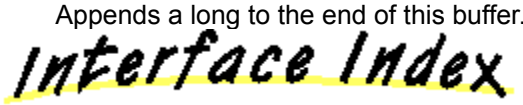
append(int)

Appends an integer to the end of this buffer.



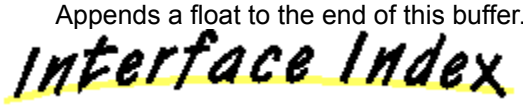
append(long)

Appends a long to the end of this buffer.



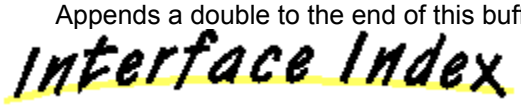
append(float)

Appends a float to the end of this buffer.



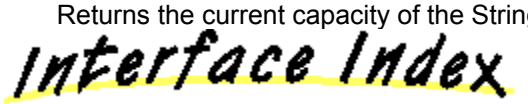
append(double)

Appends a double to the end of this buffer.



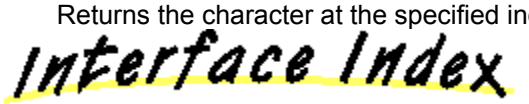
capacity()

Returns the current capacity of the String buffer.



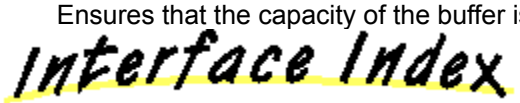
charAt(int)

Returns the character at the specified index.



ensureCapacity(int)

Ensures that the capacity of the buffer is at least equal to the specified minimum.



getChars(int, int, char[], int)

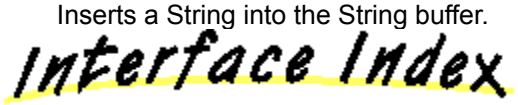
Copies the characters of the specified substring (determined by srcBegin and srcEnd) into the character array, starting at the array's dstBegin location.

insert(int, Object)

Inserts an object into the String buffer.

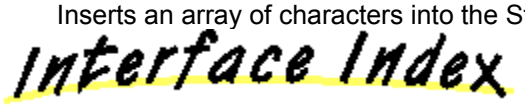
insert(int, String)

Inserts a String into the String buffer.



insert(int, char[])

Inserts an array of characters into the String buffer.

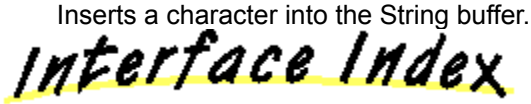


insert(int, boolean)

Inserts a boolean into the String buffer.

insert(int, char)

Inserts a character into the String buffer.

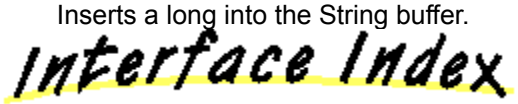


insert(int, int)

Inserts an integer into the String buffer.

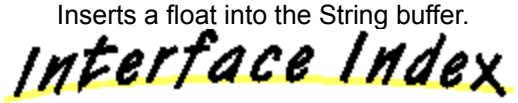
insert(int, long)

Inserts a long into the String buffer.



insert(int, float)

Inserts a float into the String buffer.

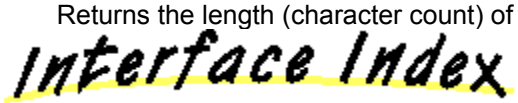


insert(int, double)

Inserts a double into the String buffer.

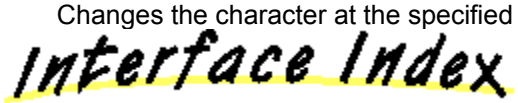
length()

Returns the length (character count) of the buffer.



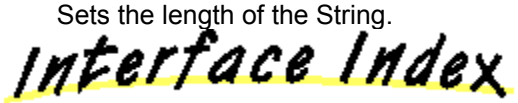
setCharAt(int, char)

Changes the character at the specified index to be ch.



setLength(int)

Sets the length of the String.



toString()

Converts to a String representing the data in the buffer.

StringBuffer

```
public StringBuffer()
```

Constructs an empty String buffer.

StringBuffer

```
public StringBuffer(int length)
```

Constructs an empty String buffer with the specified initial length.
Parameters:
length - the initial length

StringBuffer

```
public StringBuffer(String str)
```

Constructs a String buffer with the specified initial value.
Parameters:
str - the initial value of the buffer

length

```
public int length()
```

Returns the length (character count) of the buffer.

capacity

```
public int capacity()
```

Returns the current capacity of the String buffer. The capacity is the amount of storage available for newly inserted characters; beyond which an allocation will occur.

ensureCapacity

```
public synchronized void ensureCapacity(int minimumCapacity)
```

Ensures that the capacity of the buffer is at least equal to the specified minimum.
Parameters:
minimumCapacity - the minimum desired capacity

setLength

```
public synchronized void setLength(int newLength)
```

Sets the length of the String. If the length is reduced, characters are lost. If the length is extended, the values of the new characters are set to 0.
Parameters:
newLength - the new length of the buffer
Throws: StringIndexOutOfBoundsException
If the length is invalid.

charAt

```
public synchronized char charAt(int index)
```

Returns the character at the specified index. An index ranges from 0..length()-1.
Parameters:
index - the index of the desired character
Throws: StringIndexOutOfBoundsException
If the index is invalid.

getChars

```
public synchronized void getChars(int srcBegin,
                                  int srcEnd,
                                  char dst[],
                                  int dstBegin)
```

Copies the characters of the specified substring (determined by srcBegin and srcEnd) into the character array, starting at the array's dstBegin location. Both srcBegin and srcEnd must be legal indexes into the buffer.

Parameters:
srcBegin - begin copy at this offset in the String
srcEnd - stop copying at this offset in the String
dst - the array to copy the data into
dstBegin - offset into dst
Throws: StringIndexOutOfBoundsException
If there is an invalid index into the buffer.

setCharAt

```
public synchronized void setCharAt(int index,
                                   char ch)
```

Changes the character at the specified index to be ch.
Parameters:
index - the index of the character
ch - the new character
Throws: StringIndexOutOfBoundsException
If the index is invalid.

append

```
public synchronized StringBuffer append(Object obj)
```

Appends an object to the end of this buffer.
Parameters:
obj - the object to be appended
Returns:
the StringBuffer itself, NOT a new one.

append

```
public synchronized StringBuffer append(String str)
```

Appends a String to the end of this buffer.
Parameters:
str - the String to be appended
Returns:
the StringBuffer itself, NOT a new one.

```
public synchronized StringBuffer append(char str[])
```

Appends an array of characters to the end of this buffer.

Parameters:

str - the characters to be appended

Returns:

the StringBuffer itself, NOT a new one.


```
public synchronized StringBuffer append(char str[],
                                        int offset,
                                        int len)
```

Appends a part of an array of characters to the end of this buffer.

Parameters:

str - the characters to be appended

offset - where to start

len - the number of characters to add

Returns:

the StringBuffer itself, NOT a new one.


```
public StringBuffer append(boolean b)
```

Appends a boolean to the end of this buffer.

Parameters:

b - the boolean to be appended

Returns:

the StringBuffer itself, NOT a new one.


```
public synchronized StringBuffer append(char c)
```

Appends a character to the end of this buffer.

Parameters:
ch - the character to be appended
Returns:
the StringBuffer itself, NOT a new one.

append

```
public StringBuffer append(int i)
```

Appends an integer to the end of this buffer.
Parameters:
i - the integer to be appended
Returns:
the StringBuffer itself, NOT a new one.

append

```
public StringBuffer append(long l)
```

Appends a long to the end of this buffer.
Parameters:
l - the long to be appended
Returns:
the StringBuffer itself, NOT a new one.

append

```
public StringBuffer append(float f)
```

Appends a float to the end of this buffer.
Parameters:
f - the float to be appended
Returns:
the StringBuffer itself, NOT a new one.

append

```
public StringBuffer append(double d)
```

Appends a double to the end of this buffer.

Parameters:

d - the double to be appended

Returns:

the StringBuffer itself, NOT a new one.

insert

```
public synchronized StringBuffer insert(int offset,
                                        Object obj)
```

Inserts an object into the String buffer.

Parameters:

offset - the offset at which to insert

obj - the object to insert

Returns:

the StringBuffer itself, NOT a new one.

Throws: StringIndexOutOfBoundsException

If the offset is invalid.

insert

```
public synchronized StringBuffer insert(int offset,
                                        String str)
```

Inserts a String into the String buffer.

Parameters:

offset - the offset at which to insert

str - the String to insert

Returns:

the StringBuffer itself, NOT a new one.

Throws: StringIndexOutOfBoundsException

If the offset is invalid.

insert

```
public synchronized StringBuffer insert(int offset,
                                        char str[])
```

Inserts an array of characters into the String buffer.

Parameters:
offset - the offset at which to insert
str - the characters to insert
Returns:
the StringBuffer itself, NOT a new one.
Throws: StringIndexOutOfBoundsException
If the offset is invalid.

insert

```
public StringBuffer insert(int offset,
                           boolean b)
```

Inserts a boolean into the String buffer.
Parameters:
offset - the offset at which to insert
b - the boolean to insert
Returns:
the StringBuffer itself, NOT a new one.
Throws: StringIndexOutOfBoundsException
If the offset is invalid.

insert

```
public synchronized StringBuffer insert(int offset,
                                        char c)
```

Inserts a character into the String buffer.
Parameters:
offset - the offset at which to insert
ch - the character to insert
Returns:
the StringBuffer itself, NOT a new one.
Throws: StringIndexOutOfBoundsException
If the offset invalid.

insert

```
public StringBuffer insert(int offset,
                           int i)
```

Inserts an integer into the String buffer.

Parameters:
offset - the offset at which to insert
i - the integer to insert
Returns:
the StringBuffer itself, NOT a new one.
Throws: StringIndexOutOfBoundsException
If the offset is invalid.

insert

```
public StringBuffer insert(int offset,
                           long l)
```

Inserts a long into the String buffer.
Parameters:
offset - the offset at which to insert
l - the long to insert
Returns:
the StringBuffer itself, NOT a new one.
Throws: StringIndexOutOfBoundsException
If the offset is invalid.

insert

```
public StringBuffer insert(int offset,
                           float f)
```

Inserts a float into the String buffer.
Parameters:
offset - the offset at which to insert
f - the float to insert
Returns:
the StringBuffer itself, NOT a new one.
Throws: StringIndexOutOfBoundsException
If the offset is invalid.

insert

```
public StringBuffer insert(int offset,
                           double d)
```

Inserts a double into the String buffer.

Parameters:
offset - the offset at which to insert
d - the double to insert
Returns:
the StringBuffer itself, NOT a new one.
Throws: StringIndexOutOfBoundsException
If the offset is invalid.

toString

```
public String toString()
```

Converts to a String representing the data in the buffer.
Overrides:
toString in class Object

Class java.lang.StringIndexOutOfBoundsException

Class java.lang.StringIndexOutOfBoundsException

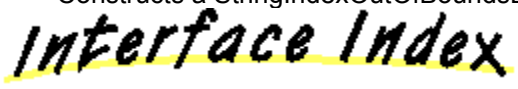
```
java.lang.Object
    |
    +----java.lang.Throwable
             |
             +----java.lang.Exception
                      |
                      +----java.lang.RuntimeException
                               |
                               +----java.lang.IndexOutOfBoundsException
                                        |
                                        +----
java.lang.StringIndexOutOfBoundsException
```

---

public class **StringIndexOutOfBoundsException**
extends IndexOutOfBoundsException

Signals that a String index is out of range.
See Also:
charAt

StringIndexOutOfBoundsException()
Constructs a StringIndexOutOfBoundsException with no detail message.



StringIndexOutOfBoundsException(String)
Constructs a StringIndexOutOfBoundsException with the specified detail message.

Interface Index **StringIndexOutOfBoundsException**(int)
Constructs a StringIndexOutOfBoundsException initialized with the specified index.

Interface Index

Interface Index **StringIndexOutOfBoundsException**

```
public StringIndexOutOfBoundsException()
```

Constructs a StringIndexOutOfBoundsException with no detail message. A detail message is a String that describes this particular exception.

Interface Index **StringIndexOutOfBoundsException**

```
public StringIndexOutOfBoundsException(String s)
```

Constructs a StringIndexOutOfBoundsException with the specified detail message. A detail message is a String that describes this particular exception.
Parameters:
s - the String containing a detail message about the error

Interface Index **StringIndexOutOfBoundsException**

```
public StringIndexOutOfBoundsException(int index)
```

Constructs a StringIndexOutOfBoundsException initialized with the specified index.
Parameters:
index - the offending index

Class java.lang.System

Class java.lang.System

```
java.lang.Object
    |
    +----java.lang.System
```

---

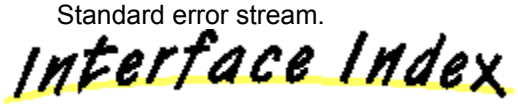
public final class **System**
extends Object

This Class provides a system-independent interface to system functionality. One of the more useful things provided by this Class are the standard input and output streams. The standard input streams are used for reading character data. The standard output streams are used for printing. For example:

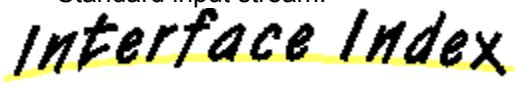
```
System.out.println("Hello World!");
```

This Class cannot be instantiated or subclassed because all of the methods and variables are static.

err
Standard error stream.



in
Standard input stream.



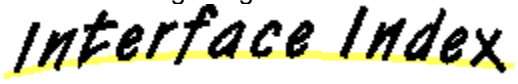
out
Standard output stream.

arraycopy(Object, int, Object, int, int)
Copies an array from the source array, beginning at the specified position, to the specified position of the destination array.

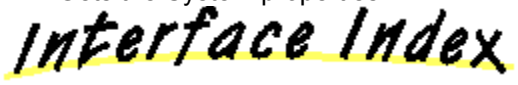
currentTimeMillis()
Returns the current time in milliseconds GMT since the epoch (00:00:00 UTC, January 1, 1970).

exit(int)
Exits the virtual machine with an exit code.

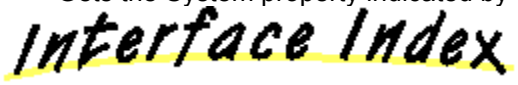
gc()
Runs the garbage collector.



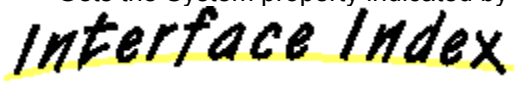
getProperties()
Gets the System properties.



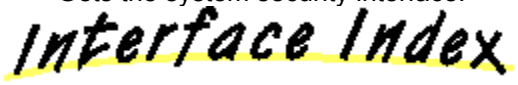
getProperty(String)
Gets the System property indicated by the specified key.



getProperty(String, String)
Gets the System property indicated by the specified key and def.

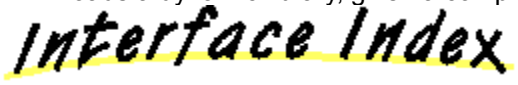


getSecurityManager()
Gets the system security interface.

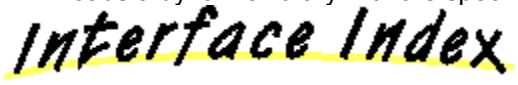


getenv(String)
Obsolete.

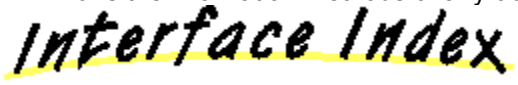
load(String)
Loads a dynamic library, given a complete path name.



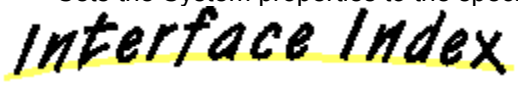
loadLibrary(String)
Loads a dynamic library with the specified library name.



runFinalization()
Runs the finalization methods of any objects pending finalization.



setProperties(Properties)
Sets the System properties to the specified properties.



setSecurityManager(SecurityManager)
Sets the System security.

in

```
public static InputStream in
```

Standard input stream. This stream is used for reading in character data.

out

```
public static PrintStream out
```

Standard output stream. This stream is used for printing messages.

err

```
public static PrintStream err
```

Standard error stream. This stream can be used to print error messages. Many applications read in data from an InputStream and output messages via the PrintStream out statement. Often applications rely on command line redirection to specify source and destination files. A problem with redirecting standard output is the incapability of writing messages to the screen if the output has been redirected to a file. This problem can be overcome by sending some output to PrintStream out and other output to PrintStream err. The difference between PrintStream err and PrintStream out is that PrintStream err is often used for displaying error messages but may be used for any purpose.

setSecurityManager

```
public static void setSecurityManager(SecurityManager s)
```

Sets the System security. This value can only be set once.

Parameters:

s - the security manager

Throws: SecurityException

If the SecurityManager has already been set.

getSecurityManager

```
public static SecurityManager getSecurityManager()
```

Gets the system security interface.

currentTimeMillis

```
public static long currentTimeMillis()
```

Returns the current time in milliseconds GMT since the epoch (00:00:00 UTC, January 1, 1970). It is a signed 64 bit integer, and so it will not overflow until the year 292280995.

See Also:

Date

arraycopy

```
public static void arraycopy(Object src,
                             int src_position,
                             Object dst,
                             int dst_position,
                             int length)
```

Copies an array from the source array, beginning at the specified position, to the specified position of the destination array. This method does not allocate memory for the destination array. The memory must already be allocated.

Parameters:

src - the source data
srcpos - start position in the source data
dest - the destination
destpos - start position in the destination data
length - the number of array elements to be copied

Throws: ArrayIndexOutOfBoundsException

If copy would cause access of data outside array bounds.

Throws: ArrayStoreException

If an element in the src array could could not be stored into the destination array due to a type mismatch

getProperties

```
public static Properties getProperties()
```

Gets the System properties.

setProperties

```
public static void setProperties(Properties props)
```

Sets the System properties to the specified properties.

Parameters:

props - the properties to be set

getProperty

```
public static String getProperty(String key)
```

Gets the System property indicated by the specified key.

Parameters:

key - the name of the system property

getProperty

```
public static String getProperty(String key,
                                 String def)
```

Gets the System property indicated by the specified key and def.

Parameters:

key - the name of the system property

def - the default value to use if this property is not set

getenv

```
public static String getenv(String name)
```

Obsolete. Gets an environment variable. An environment variable is a system dependent external variable that has a string value.

Parameters:

name - the name of the environment variable

Returns:

the value of the variable, or null if the variable is not defined.

exit

```
public static void exit(int status)
```

Exits the virtual machine with an exit code. This method does not return, use with caution.

Parameters:

status - exit status, 0 if successful, other values indicate various error types.

See Also:

exit

gc

```
public static void gc()
```

Runs the garbage collector.

See Also:

gc

runFinalization

```
public static void runFinalization()
```

Runs the finalization methods of any objects pending finalization.

See Also:

gc

load

```
public static void load(String filename)
```

Loads a dynamic library, given a complete path name.

Parameters:

filename - the file to load

Throws: UnsatisfiedLinkError

If the file does not exist.

See Also:

load

loadLibrary

```
public static void loadLibrary(String libname)
```

Loads a dynamic library with the specified library name.

Parameters:

libname - the name of the library

Throws: UnsatisfiedLinkError

If the library does not exist.

See Also:

loadLibrary

Class java.lang.Thread

Class java.lang.Thread

```
java.lang.Object
   |
   +----java.lang.Thread
```

---

public class **Thread**
extends Object
implements Runnable

A Thread is a single sequential flow of control within a process. This simply means that while executing within a program, each thread has a beginning, a sequence, a point of execution occurring at any time during runtime of the thread and of course, an ending. Thread objects are the basis for multi-threaded programming. Multi-threaded programming allows a single program to conduct concurrently running threads that perform different tasks.

To create a new thread of execution, declare a new class which is a subclass of Thread and then override the run() method with code that you want executed in this Thread. An instance of the Thread subclass should be created next with a call to the start() method following the instance. The start() method will create the thread and execute the run() method. For example:

```
class PrimeThread extends Thread {
    public void run() {
        // compute primes...
    }
}
```

To start this thread you need to do the following:

```
PrimeThread p = new PrimeThread();
p.start();
...
```

Another way to create a thread is by using the Runnable interface. This way any object that implements the Runnable interface can be run in a thread. For example:

```
class Primes implements Runnable {
    public void run() {
        // compute primes...
    }
}
```

To start this thread you need to do the following:

```
Primes p = new Primes();
new Thread(p).start();
...
```

The virtual machine runs until all Threads that are not daemon Threads have died. A Thread dies when its run() method returns, or when the stop() method is called.

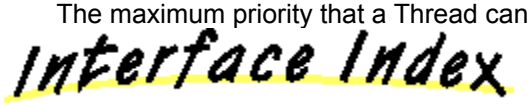
When a new Thread is created, it inherits the priority and the daemon flag from its parent (i.e.: the Thread that created it).

See Also:

Runnable

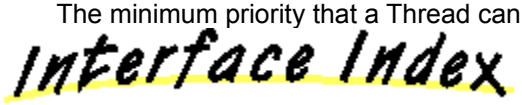
MAX_PRIORITY

The maximum priority that a Thread can have.



MIN_PRIORITY

The minimum priority that a Thread can have.



NORM_PRIORITY

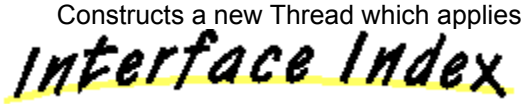
The default priority that is assigned to a Thread.

Thread()

Constructs a new Thread.

Thread(Runnable)

Constructs a new Thread which applies the run() method of the specified target.

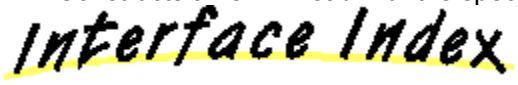


Thread(ThreadGroup, Runnable)

Constructs a new Thread in the specified Thread group that applies the run() method of the specified target.

Thread(String)

Constructs a new Thread with the specified name.



Thread(ThreadGroup, String)

Constructs a new Thread in the specified Thread group with the specified name.

Thread(Runnable, String)

Constructs a new Thread with the specified name and applies the run() method of the specified target.

Thread(ThreadGroup, Runnable, String)

Constructs a new Thread in the specified Thread group with the specified name and applies the run() method of the specified target.

activeCount()

Returns the current number of active Threads in this Thread group.

checkAccess()

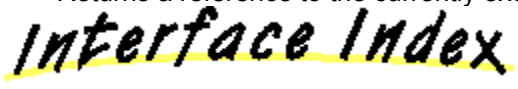
Checks whether the current Thread is allowed to modify this Thread.

countStackFrames()

Returns the number of stack frames in this Thread.

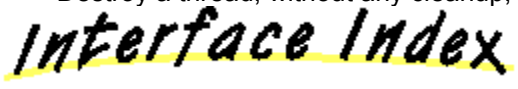
currentThread()

Returns a reference to the currently executing Thread object.



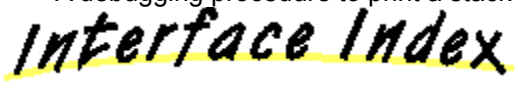
destroy()

Destroy a thread, without any cleanup, i.e.



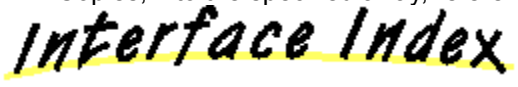
dumpStack()

A debugging procedure to print a stack trace for the current Thread.



enumerate(Thread[])

Copies, into the specified array, references to every active Thread in this Thread's group.

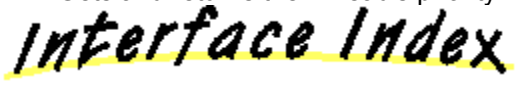


getName()

Gets and returns this Thread's name.

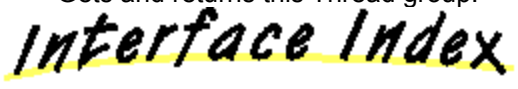
getPriority()

Gets and returns the Thread's priority.



getThreadGroup()

Gets and returns this Thread group.



interrupt()

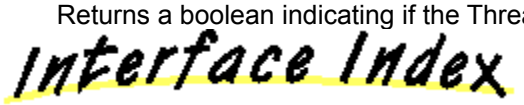
Send an interrupt to a thread.

interrupted()

Ask if you have been interrupted.

isAlive()

Returns a boolean indicating if the Thread is active.

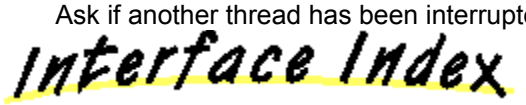


isDaemon()

Returns the daemon flag of the Thread.

isInterrupted()

Ask if another thread has been interrupted.

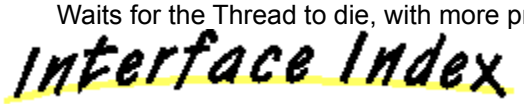


join(long)

Waits for this Thread to die.

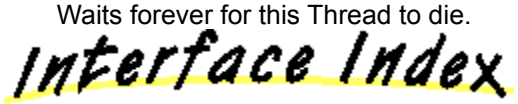
join(long, int)

Waits for the Thread to die, with more precise time.



join()

Waits forever for this Thread to die.

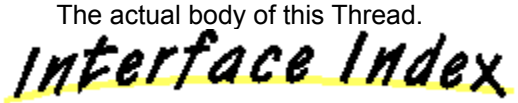


resume()

Resumes this Thread execution.

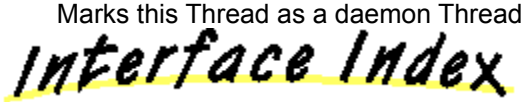
run()

The actual body of this Thread.



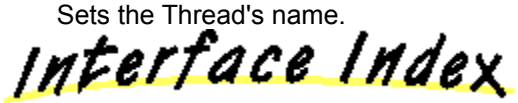
setDaemon(boolean)

Marks this Thread as a daemon Thread or a user Thread.



setName(String)

Sets the Thread's name.

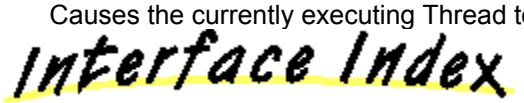


setPriority(int)

Sets the Thread's priority.

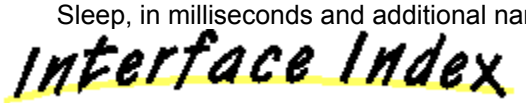
sleep(long)

Causes the currently executing Thread to sleep for the specified number of milliseconds.



sleep(long, int)

Sleep, in milliseconds and additional nanosecond.



start()

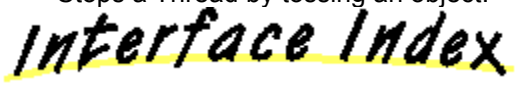
Starts this Thread.

stop()

Stops a Thread by tossing an object.

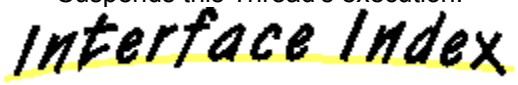
stop(Throwable)

Stops a Thread by tossing an object.



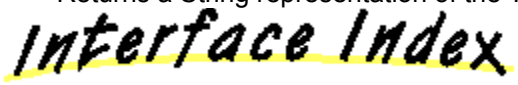
suspend()

Suspends this Thread's execution.



toString()

Returns a String representation of the Thread, including the thread's name, priority and thread group.



yield()

Causes the currently executing Thread object to yield.

MIN_PRIORITY

```
public final static int MIN_PRIORITY
```

The minimum priority that a Thread can have. The most minimal priority is equal to 1.

NORM_PRIORITY

```
public final static int NORM_PRIORITY
```

The default priority that is assigned to a Thread. The default priority is equal to 5.

MAX_PRIORITY

```
public final static int MAX_PRIORITY
```

The maximum priority that a Thread can have. The maximal priority value a Thread can have is 10.

```
public Thread()
```

Constructs a new Thread. Threads created this way must have overridden their run() method to actually do anything. An example illustrating this method being used is shown.

```
import java.lang.*;


class plain01 implements Runnable {
  String name;
  plain01() {
    name = null;
  }
  plain01(String s) {
    name = s;
  }
  public void run() {
    if (name == null)
      System.out.println("A new thread created");
    else
      System.out.println("A new thread with name " + name + " created");
  }
}


class threadtest01 {
  public static void main(String args[] ) {
    int failed = 0 ;


   Thread t1 = new Thread();
    if(t1 != null) {
      System.out.println("new Thread() succeed");
    } else {
       System.out.println("new Thread() failed");
       failed++;
    }
}
```

Interface Index Thread

```
public Thread(Runnable target)
```

Constructs a new Thread which applies the run() method of the specified target.

Parameters:

target - the object whose run() method is called

Thread

```
public Thread(ThreadGroup group,
              Runnable target)
```

Constructs a new Thread in the specified Thread group that applies the run() method of the specified target.

Parameters:

group - the Thread group

target - the object whose run() method is called

Thread

```
public Thread(String name)
```

Constructs a new Thread with the specified name.

Parameters:

name - the name of the new Thread

Thread

```
public Thread(ThreadGroup group,
              String name)
```

Constructs a new Thread in the specified Thread group with the specified name.

Parameters:

group - the Thread group

name - the name of the new Thread

Thread

```
public Thread(Runnable target,
              String name)
```

Constructs a new Thread with the specified name and applies the run() method of the specified target.

Parameters:

target - the object whose run() method is called
name - the name of the new Thread

Thread

```
public Thread(ThreadGroup group,
              Runnable target,
              String name)
```

Constructs a new Thread in the specified Thread group with the specified name and applies the run() method of the specified target.

Parameters:

group - the Thread group
target - the object whose run() method is called
name - the name of the new Thread

currentThread

```
public static Thread currentThread()
```

Returns a reference to the currently executing Thread object.

yield

```
public static void yield()
```

Causes the currently executing Thread object to yield. If there are other runnable Threads they will be scheduled next.

sleep

```
public static void sleep(long millis) throws InterruptedException
```

Causes the currently executing Thread to sleep for the specified number of milliseconds.

Parameters:

millis - the length of time to sleep in milliseconds

Throws: InterruptedException

Another thread has interrupted this thread.

sleep

```
public static void sleep(long millis,
                         int nanos) throws InterruptedException
```

Sleep, in milliseconds and additional nanosecond.

Parameters:

millis - the length of time to sleep in milliseconds

nanos - 0-999999 additional nanoseconds to sleep

Throws: InterruptedException

Another thread has interrupted this thread.

start

```
public synchronized void start()
```

Starts this Thread. This will cause the run() method to be called. This method will return immediately.

Throws: IllegalThreadStateException

If the thread was already started.

See Also:

run, stop

run

```
public void run()
```

The actual body of this Thread. This method is called after the Thread is started. You must either override this method by subclassing class Thread, or you must create the Thread with a Runnable target.

See Also:

start, stop

stop

```
public final void stop()
```

Stops a Thread by tossing an object. By default this routine tosses a new instance of ThreadDeath to the target Thread. ThreadDeath is not actually a subclass of Exception, but is a subclass of Object. Users should not normally try to catch ThreadDeath unless they must do some extraordinary cleanup operation. If ThreadDeath is caught it is important to rethrow the object so that the thread will actually die. The top-level error handler will not print out a message if ThreadDeath falls through.

See Also:

start, run

stop

```
public final synchronized void stop(Throwable o)
```

Stops a Thread by tossing an object. Normally, users should just call the stop() method without any argument. However, in some exceptional circumstances used by the stop() method to kill a Thread, another object is tossed. ThreadDeath, is not actually a subclass of Exception, but is a subclass of Throwable

Parameters:

o - the Throwable object to be thrown

See Also:

start, run

interrupt

```
public void interrupt()
```

Send an interrupt to a thread.

interrupted

```
public static boolean interrupted()
```

Ask if you have been interrupted.

isInterrupted

```
public boolean isInterrupted()
```

Ask if another thread has been interrupted.

destroy

```
public void destroy()
```

Destroy a thread, without any cleanup, i.e. just toss its state; any monitors it has locked remain locked. A last resort.

isAlive

```
public final boolean isAlive()
```

Returns a boolean indicating if the Thread is active. Having an active Thread means that the Thread has been started and has not been stopped.

suspend

```
public final void suspend()
```

Suspends this Thread's execution.

resume

```
public final void resume()
```

Resumes this Thread execution. This method is only valid after suspend() has been invoked.

setPriority

```
public final void setPriority(int newPriority)
```

Sets the Thread's priority.
Throws: IllegalArgumentException
If the priority is not within the range MIN_PRIORITY, MAX_PRIORITY.
See Also:
MIN_PRIORITY, MAX_PRIORITY, getPriority

getPriority

```
public final int getPriority()
```

Gets and returns the Thread's priority.

See Also:

setPriority

setName

```
public final void setName(String name)
```

Sets the Thread's name.

Parameters:

name - the new name of the Thread

See Also:

getName

getName

```
public final String getName()
```

Gets and returns this Thread's name.

See Also:

setName

getThreadGroup

```
public final ThreadGroup getThreadGroup()
```

Gets and returns this Thread group.

activeCount

```
public static int activeCount()
```

Returns the current number of active Threads in this Thread group.

enumerate

```
public static int enumerate(Thread tarray[])
```

Copies, into the specified array, references to every active Thread in this Thread's group.

Returns:

the number of Threads put into the array.

countStackFrames

```
public int countStackFrames()
```

Returns the number of stack frames in this Thread. The Thread must be suspended when this method is called.

Throws: IllegalThreadStateException

If the Thread is not suspended.

join

```
public final synchronized void join(long millis) throws
InterruptedException
```

Waits for this Thread to die. A timeout in milliseconds can be specified. A timeout of 0 milliseconds means to wait forever.

Parameters:

millis - the time to wait in milliseconds

Throws: InterruptedException

Another thread has interrupted this thread.

join

```
public final synchronized void join(long millis,
                                    int nanos) throws InterruptedException
```

Waits for the Thread to die, with more precise time.

Throws: InterruptedException

Another thread has interrupted this thread.

join

```
public final void join() throws InterruptedException
```

Waits forever for this Thread to die.
Throws: InterruptedException
Another thread has interrupted this thread.

dumpStack

```
public static void dumpStack()
```

A debugging procedure to print a stack trace for the current Thread.
See Also:
printStackTrace

setDaemon

```
public final void setDaemon(boolean on)
```

Marks this Thread as a daemon Thread or a user Thread. When there are only daemon Threads left running in the system, Java exits.
Parameters:
on - determines whether the Thread will be a daemon Thread
Throws: IllegalThreadStateException
If the Thread is active.
See Also:
isDaemon

isDaemon

```
public final boolean isDaemon()
```

Returns the daemon flag of the Thread.
See Also:
setDaemon

 checkAccess

```
public void checkAccess()
```

Checks whether the current Thread is allowed to modify this Thread.
Throws: SecurityException
If the current Thread is not allowed to access this Thread group.

 toString

```
public String toString()
```

Returns a String representation of the Thread, including the thread's name, priority and thread group.
Overrides:
toString in class Object

Class java.lang.ThreadDeath

Class java.lang.ThreadDeath

```
java.lang.Object
    |
    +----java.lang.Throwable
            |
            +----java.lang.Error
                    |
                    +----java.lang.ThreadDeath
```

---

public class **ThreadDeath**
extends Error

An instance of ThreadDeath is thrown in the victim thread when thread.stop() is called. This is not a subclass of Exception, but rather a subclass of Error because too many people already catch Exception. Instances of this class should be caught explicitly only if you are interested in cleaning up when being asynchronously terminated. If ThreadDeath is caught, it is important to rethrow the object so that the Thread will actually die. The top-level error handler will not print out a message if ThreadDeath falls through.

Interface Index

Interface Index **ThreadDeath**()

Interface Index

Interface Index **ThreadDeath**

```
public ThreadDeath()
```

Class java.lang.ThreadGroup

Class java.lang.ThreadGroup

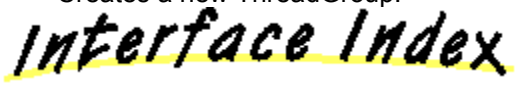
```
java.lang.Object
    |
    +----java.lang.ThreadGroup
```

---

public class **ThreadGroup**
extends Object

A group of Threads. A Thread group can contain a set of Threads as well as a set of other Thread groups. A Thread can access its Thread group, but it can't access the parent of its Thread group. This makes it possible to encapsulate a Thread in a Thread group and stop it from manipulating Threads in the parent group.

ThreadGroup(String)
Creates a new ThreadGroup.



ThreadGroup(ThreadGroup, String)
Creates a new ThreadGroup with a specified name in the specified Thread group.

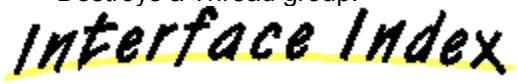
activeCount()
Returns an estimate of the number of active Threads in the Thread group.

activeGroupCount()
Returns an estimate of the number of active groups in the Thread group.

checkAccess()
Checks to see if the current Thread is allowed to modify this group.

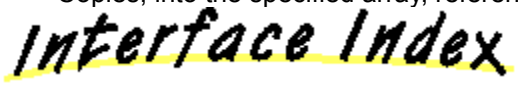
destroy()

Destroys a Thread group.



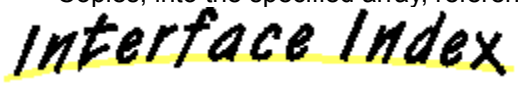
enumerate(Thread[])

Copies, into the specified array, references to every active Thread in this Thread group.



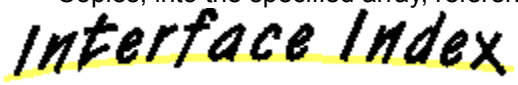
enumerate(Thread[], boolean)

Copies, into the specified array, references to every active Thread in this Thread group.



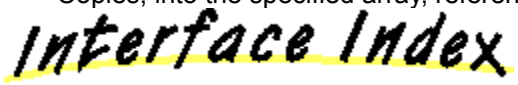
enumerate(ThreadGroup[])

Copies, into the specified array, references to every active Thread group in this Thread group.



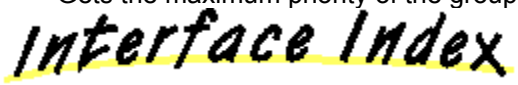
enumerate(ThreadGroup[], boolean)

Copies, into the specified array, references to every active Thread group in this Thread group.



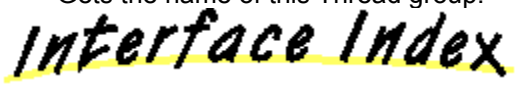
getMaxPriority()

Gets the maximum priority of the group.



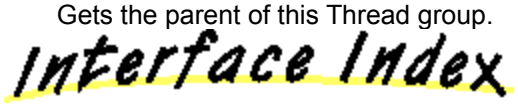
getName()

Gets the name of this Thread group.



getParent()

Gets the parent of this Thread group.

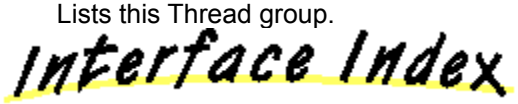


isDaemon()

Returns the daemon flag of the Thread group.

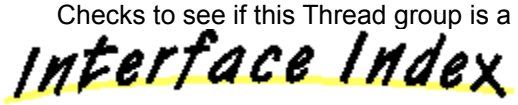
list()

Lists this Thread group.



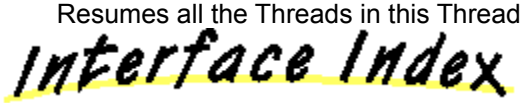
parentOf(ThreadGroup)

Checks to see if this Thread group is a parent of or is equal to another Thread group.



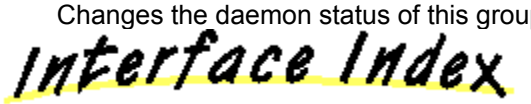
resume()

Resumes all the Threads in this Thread group and all of its sub groups.



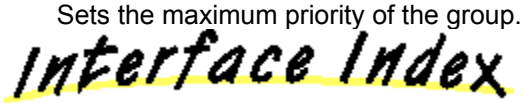
setDaemon(boolean)

Changes the daemon status of this group.



setMaxPriority(int)

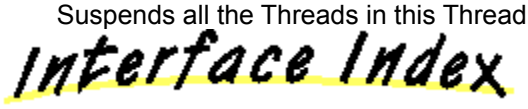
Sets the maximum priority of the group.



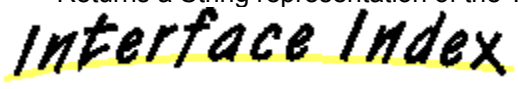
stop()

Stops all the Threads in this Thread group and all of its sub groups.

suspend()
Suspends all the Threads in this Thread group and all of its sub groups.



toString()
Returns a String representation of the Thread group.



uncaughtException(Thread, Throwable)
Called when a thread in this group exists because of an uncaught exception.

ThreadGroup

```
public ThreadGroup(String name)
```

Creates a new ThreadGroup. Its parent will be the Thread group of the current Thread.
Parameters:
name - the name of the new Thread group created

ThreadGroup

```
public ThreadGroup(ThreadGroup parent,
                   String name)
```

Creates a new ThreadGroup with a specified name in the specified Thread group.
Parameters:
parent - the specified parent Thread group
name - the name of the new Thread group being created
Throws: NullPointerException
If the given thread group is equal to null.

getName

```
public final String getName()
```

Gets the name of this Thread group.

getParent

```
public final ThreadGroup getParent()
```

Gets the parent of this Thread group.

getMaxPriority

```
public final int getMaxPriority()
```

Gets the maximum priority of the group. Threads that are part of this group cannot have a higher priority than the maximum priority.

isDaemon

```
public final boolean isDaemon()
```

Returns the daemon flag of the Thread group. A daemon Thread group is automatically destroyed when it is found empty after a Thread group or Thread is removed from it.

setDaemon

```
public final void setDaemon(boolean daemon)
```

Changes the daemon status of this group.

Parameters:

daemon - the daemon boolean which is to be set.

setMaxPriority

```
public final synchronized void setMaxPriority(int pri)
```

Sets the maximum priority of the group. Threads that are already in the group **can** have a higher priority than the set maximum.

Parameters:

pri - the priority of the Thread group

 parentOf

```
public final boolean parentOf(ThreadGroup g)
```

Checks to see if this Thread group is a parent of or is equal to another Thread group.
Parameters:
g - the Thread group to be checked
Returns:
true if this Thread group is equal to or is the parent of another Thread group; false otherwise.

 checkAccess

```
public final void checkAccess()
```

Checks to see if the current Thread is allowed to modify this group.
Throws: SecurityException
If the current Thread is not allowed to access this Thread group.

 activeCount

```
public synchronized int activeCount()
```

Returns an estimate of the number of active Threads in the Thread group.

 enumerate

```
public int enumerate(Thread list[])
```

Copies, into the specified array, references to every active Thread in this Thread group. You can use the activeCount() method to get an estimate of how big the array should be.
Parameters:
list - an array of Threads
Returns:
the number of Threads put into the array

 enumerate

```
public int enumerate(Thread list[],
                     boolean recurse)
```

Copies, into the specified array, references to every active Thread in this Thread group. You can use the activeCount() method to get an estimate of how big the array should be.

Parameters:

list - an array list of Threads

recurse - a boolean indicating whether a Thread has reapearred

Returns:

the number of Threads placed into the array.

activeGroupCount

```
public synchronized int activeGroupCount()
```

Returns an estimate of the number of active groups in the Thread group.

enumerate

```
public int enumerate(ThreadGroup list[])
```

Copies, into the specified array, references to every active Thread group in this Thread group. You can use the activeGroupCount() method to get an estimate of how big the array should be.

Parameters:

list - an array of Thread groups

Returns:

the number of Thread groups placed into the array.

enumerate

```
public int enumerate(ThreadGroup list[],
                     boolean recurse)
```

Copies, into the specified array, references to every active Thread group in this Thread group. You can use the activeGroupCount() method to get an estimate of how big the array should be.

Parameters:

list - an array list of Thread groups

recurse - a boolean indicating if a Thread group has reappeared

Returns:

the number of Thread groups placed into the array.

stop

```
public final synchronized void stop()
```

Stops all the Threads in this Thread group and all of its sub groups.

suspend

```
public final synchronized void suspend()
```

Suspends all the Threads in this Thread group and all of its sub groups.

resume

```
public final synchronized void resume()
```

Resumes all the Threads in this Thread group and all of its sub groups.

destroy

```
public final synchronized void destroy()
```

Destroys a Thread group. This does **NOT** stop the Threads in the Thread group.
Throws: IllegalThreadStateException
If the Thread group is not empty or if the Thread group was already destroyed.

list

```
public synchronized void list()
```

Lists this Thread group. Useful for debugging only.

uncaughtException

```
public void uncaughtException(Thread t,
                              Throwable e)
```

Called when a thread in this group exists because of an uncaught exception.

toString

```
public String toString()
```

Returns a String representation of the Thread group.

Overrides:

toString in class Object

Class java.lang.Throwable

Class java.lang.Throwable

```
java.lang.Object
   |
   +----java.lang.Throwable
```

---

public class **Throwable**
extends Object

An object signalling that an exceptional condition has occurred. All exceptions are a subclass of Exception. An exception contains a snapshot of the execution stack, this snapshot is used to print a stack backtrace. An exception also contains a message string. Here is an example of how to catch an exception:

```
try {
    int a[] = new int[2];
    a[4];
} catch (ArrayIndexOutOfBoundsException e) {
    System.out.println("an exception occurred: " + e.getMessage());
    e.printStackTrace();
}
```

Throwable()
Constructs a new Throwable with no detail message.

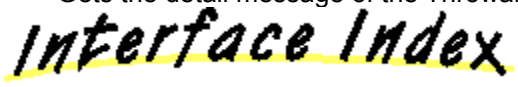
Throwable(String)
Constructs a new Throwable with the specified detail message.

fillInStackTrace()

Fills in the excecution stack trace.

getMessage()

Gets the detail message of the Throwable.



printStackTrace()

Prints the Throwable and the Throwable's stack trace.

printStackTrace(PrintStream)

toString()

Returns a short description of the Throwable.

Throwable

```
public Throwable()
```

Constructs a new Throwable with no detail message. The stack trace is automatically filled in.

Throwable

```
public Throwable(String message)
```

Constructs a new Throwable with the specified detail message. The stack trace is automatically filled in.

Parameters:

message - the detailed message

getMessage

```
public String getMessage()
```

Gets the detail message of the Throwable. A detail message is a String that describes the Throwable that has taken place.

Returns:
the detail message of the throwable.

toString

```
public String toString()
```

Returns a short description of the Throwable.
Overrides:
toString in class Object

printStackTrace

```
public void printStackTrace()
```

Prints the Throwable and the Throwable's stack trace.

printStackTrace

```
public void printStackTrace(PrintStream s)
```

fillInStackTrace

```
public Throwable fillInStackTrace()
```

Fills in the excecution stack trace. This is useful only when rethrowing a Throwable. For example:

```
try {
     a = b / c;
} catch(ArithmeticThrowable e) {
     a = Number.MAX_VALUE;
     throw e.fillInStackTrace();
}
```

Returns:
the Throwable itself.
See Also:
printStackTrace

Class java.lang.UnknownError

Class java.lang.UnknownError

```
java.lang.Object
    |
    +----java.lang.Throwable
            |
            +----java.lang.Error
                    |
                    +----java.lang.VirtualMachineError
                            |
                            +----java.lang.UnknownError
```

---

public class **UnknownError**
extends VirtualMachineError

Signals that an unknown but serious exception has occurred.

UnknownError()
Constructs an UnknownError with no detail message.

UnknownError(String)
Constructs an UnknownError with the specified detail message.

UnknownError

```
public UnknownError()
```

Constructs an UnknownError with no detail message. A detail message is a String that describes this particular exception.

UnknownError

```
public UnknownError(String s)
```

Constructs an UnknownError with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class java.lang.UnsatisfiedLinkError

Class java.lang.UnsatisfiedLinkError

```
java.lang.Object
    |
    +----java.lang.Throwable
            |
            +----java.lang.Error
                    |
                    +----java.lang.LinkageError
                            |
                            +----java.lang.UnsatisfiedLinkError
```

---

public class **UnsatisfiedLinkError**
extends LinkageError

Signals an unsatisfied link.
See Also:
Runtime

UnsatisfiedLinkError()
Constructs an UnsatisfiedLinkError with no detail message.

UnsatisfiedLinkError(String)
Constructs an UnsatisfiedLinkError with the specified detail message.

UnsatisfiedLinkError

```
public UnsatisfiedLinkError()
```

Constructs an UnsatisfiedLinkError with no detail message. A detail message is a String that describes this particular exception.

UnsatisfiedLinkError

```
public UnsatisfiedLinkError(String s)
```

Constructs an UnsatisfiedLinkError with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class java.lang.VerifyError

Class java.lang.VerifyError

```
java.lang.Object
    |
    +----java.lang.Throwable
            |
            +----java.lang.Error
                    |
                    +----java.lang.LinkageError
                            |
                            +----java.lang.VerifyError
```

---

public class **VerifyError**
extends LinkageError

Signals that a Verification Error occured.

VerifyError()

Constructor.

VerifyError(String)

Constructor with a detail message.

VerifyError

```
public VerifyError()
```

Constructor.

```
public VerifyError(String s)
```

Constructor with a detail message.

Class java.lang.VirtualMachineError

Class java.lang.VirtualMachineError

```
java.lang.Object
    |
    +----java.lang.Throwable
            |
            +----java.lang.Error
                    |
                    +----java.lang.VirtualMachineError
```

---

public class **VirtualMachineError**
extends Error

A VirtualMachineError indicates that the virtual machine is broken or has run out of resources.

VirtualMachineError()
Constructs a VirtualMachineError with no detail message.

VirtualMachineError(String)
Constructs a VirtualMachineError with the specified detail message.

VirtualMachineError

```
public VirtualMachineError()
```

Constructs a VirtualMachineError with no detail message. A detail message is a String that describes this particular exception.

```
public VirtualMachineError(String s)
```

Constructs a VirtualMachineError with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class java.net.ContentHandler

Class java.net.ContentHandler

```
java.lang.Object
    |
    +----java.net.ContentHandler
```

---

public class **ContentHandler**
extends Object

A class to read data from a URLConnection and construct an Object. Specific subclasses of ContentHandler handle specific mime types. It is the responsibility of a ContentHandlerFactory to select an appropriate ContentHandler for the mime-type of the URLConnection. Applications should never call ContentHandlers directly, rather they should use URL.getContent() or URLConnection.getContent()

ContentHandler()

getContent(URLConnection)

Given an input stream positioned at the beginning of the representation of an object, reads that stream and recreates the object from it.

ContentHandler

```
public ContentHandler()
```

Interface Index**getContent**

```
public abstract Object getContent(URLConnection urlc) throws IOException
```

Given an input stream positioned at the beginning of the representation of an object, reads that stream and recreates the object from it.

Throws: IOException

An IO error occurred while reading the object.

Interface java.net.ContentHandlerFactory

Interface java.net.ContentHandlerFactory

public interface **ContentHandlerFactory**
extends Object

This interface defines a factory for ContentHandler instances. It is used by the URLStreamHandler class to create ContentHandlers for various streams.

createContentHandler(String)
Creates a new ContentHandler to read an object from a URLStreamHandler.

createContentHandler

```
public abstract ContentHandler createContentHandler(String mimetype)
```

Creates a new ContentHandler to read an object from a URLStreamHandler.

Parameters:

mimetype - The mime type for which a content handler is desired.

Class java.net.DatagramPacket

Class java.net.DatagramPacket

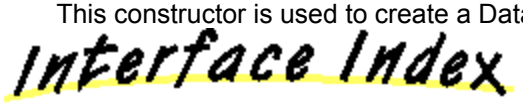
```
java.lang.Object
    |
    +----java.net.DatagramPacket
```

---

public final class **DatagramPacket**
extends Object

A class that represents a datagram packet containing packet data, packet length, internet addresses and port.

DatagramPacket(byte[], int)
This constructor is used to create a DatagramPacket object used for receiving datagrams.



DatagramPacket(byte[], int, InetAddress, int)
This constructor is used construct the DatagramPacket to be sent.

getAddress()

getData()

getLength()

getPort()

Interface Index

Interface Index **DatagramPacket**

```
public DatagramPacket(byte ibuf[],
                      int ilength)
```

This constructor is used to create a DatagramPacket object used for receiving datagrams.
Parameters:
ibuf - is where packet data is to be received.
ilength - is the number of bytes to be received.

Interface Index **DatagramPacket**

```
public DatagramPacket(byte ibuf[],
                      int ilength,
                      InetAddress iaddr,
                      int iport)
```

This constructor is used construct the DatagramPacket to be sent.
Parameters:
ibuf - contains the packet data.
ilength - contains the packet length
iaddr - and iport contains destination ip addr and port number.

Interface Index

Interface Index **getAddress**

```
public InetAddress getAddress()
```

Interface Index **getPort**

```
public int getPort()
```

Interface Index **getData**

```
public byte[] getData()
```

Interface Index **getLength**

```
public int getLength()
```

Class java.net.DatagramSocket

Class java.net.DatagramSocket

```
java.lang.Object
    |
    +----java.net.DatagramSocket
```

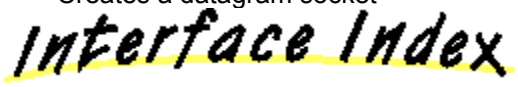
---

public class **DatagramSocket**
extends Object

The datagram socket class implements unreliable datagrams.

DatagramSocket()

Creates a datagram socket



DatagramSocket(int)

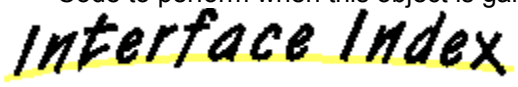
Creates a datagram socket

close()

Close the datagram socket.

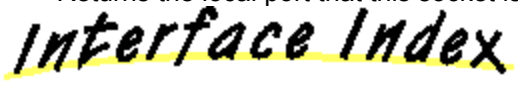
finalize()

Code to perform when this object is garbage collected.



getLocalPort()

Returns the local port that this socket is bound to.



receive(DatagramPacket)

Receives datagram packet.

Interface Index **send**(DatagramPacket)

Sends Datagram Packet to the destination address

Interface Index

Interface Index **DatagramSocket**

```
public DatagramSocket() throws SocketException
```

Creates a datagram socket

Interface Index **DatagramSocket**

```
public DatagramSocket(int port) throws SocketException
```

Creates a datagram socket

Parameters:

local - port to use

Interface Index

Interface Index **send**

```
public void send(DatagramPacket p) throws IOException
```

Sends Datagram Packet to the destination address

Parameters:

DatagramPacket - to be sent. The packet contains the buffer of bytes, length and destination InetAddress and port.

Throws: IOException

i/o error occurred

Interface Index **receive**

```
public synchronized void receive(DatagramPacket p) throws IOException
```

Receives datagram packet.

Parameters:

DatagramPacket - to be received. On return, the DatagramPacket contains the buffer in which the data is received, packet length, sender's address and sender's port number. Blocks until some input is available.

Throws: IOException

i/o error occurred

getLocalPort

```
public int getLocalPort()
```

Returns the local port that this socket is bound to.

close

```
public synchronized void close()
```

Close the datagram socket.

finalize

```
protected synchronized void finalize()
```

Code to perform when this object is garbage collected.

Overrides:

finalize in class Object

Class java.net.InetAddress

Class java.net.InetAddress

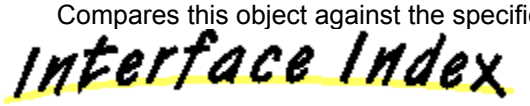
```
java.lang.Object
    |
    +----java.net.InetAddress
```

---

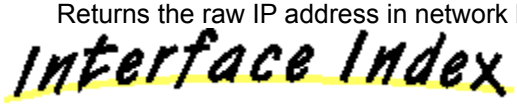
public final class **InetAddress**
extends Object

A class that represents Internet addresses.

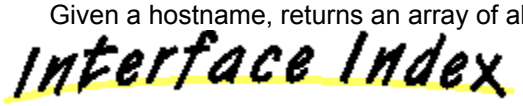
equals(Object)
Compares this object against the specified object.



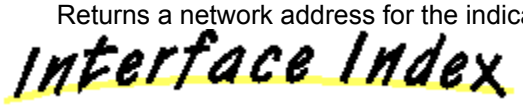
getAddress()
Returns the raw IP address in network byte order.



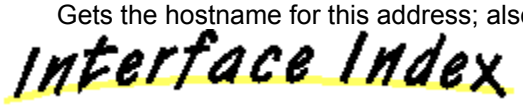
getAllByName(String)
Given a hostname, returns an array of all the corresponding InetAddresses.



getByName(String)
Returns a network address for the indicated host.



getHostName()
Gets the hostname for this address; also the key in the hashtable.



getLocalHost()
Returns the local host.

hashCode()
Returns a hashcode for this InetAddress.

 toString()

Converts the InetAddress to a String.

 getHostName

```
public String getHostName()
```

Gets the hostname for this address; also the key in the hashtable. If the host is equal to null, then this address refers to any of the local machine's available network addresses.

 getAddress

```
public byte[] getAddress()
```

Returns the raw IP address in network byte order. The highest order byte position is in addr[0]. To be prepared for 64-bit IP addresses n array of bytes is returned.

Returns:

raw IP address in network byte order.

 hashCode

```
public int hashCode()
```

Returns a hashcode for this InetAddress.

Overrides:

hashCode in class Object

 equals

```
public boolean equals(Object obj)
```

Compares this object against the specified object.

Parameters:

obj - the object to compare against.

Returns:

true if the objects are the same; false otherwise.

Overrides:

equals in class Object

toString

```
public String toString()
```

Converts the InetAddress to a String.

Overrides:

toString in class Object

getByName

```
public static synchronized InetAddress getByName(String host) throws
UnknownHostException
```

Returns a network address for the indicated host. A host name of null refers to default address for the local machine. A local cache is used to speed access to addresses. If all addresses for a host are needed, use the getAllByName() method.

Parameters:

host - the specified host

Throws: UnknownHostException

If the address is unknown.

getAllByName

```
public static synchronized InetAddress[] getAllByName(String host) throws
UnknownHostException
```

Given a hostname, returns an array of all the corresponding InetAddresses.

Throws: UnknownHostException

If the host name could not be resolved

getLocalHost

```
public static InetAddress getLocalHost() throws UnknownHostException
```

Returns the local host.
Throws: UnknownHostException
If the host name could not be resolved

Class java.net.MalformedURLException

Class java.net.MalformedURLException

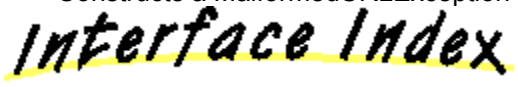
```
java.lang.Object
    |
    +----java.lang.Throwable
            |
            +----java.lang.Exception
                    |
                    +----java.io.IOException
                            |
                            +----java.net.MalformedURLException
```

---

public class **MalformedURLException**
extends IOException

Signals that a malformed URL has occurred.

MalformedURLException()
Constructs a MalformedURLException with no detail message.



MalformedURLException(String)
Constructs a MalformedURLException with the specified detail message.

MalformedURLException

```
public MalformedURLException()
```

Constructs a MalformedURLException with no detail message. A detail message is a String that describes this particular exception.

MalformedURLException

```
public MalformedURLException(String msg)
```

Constructs a MalformedURLException with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

msg - the detail message

Class java.net.ProtocolException

Class java.net.ProtocolException

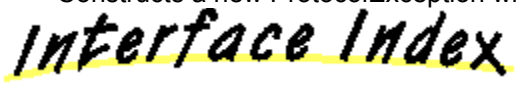
```
java.lang.Object
    |
    +----java.lang.Throwable
            |
            +----java.lang.Exception
                    |
                    +----java.io.IOException
                            |
                            +----java.net.ProtocolException
```

---

public class **ProtocolException**
extends IOException

Signals when connect gets an EPROTO. This exception is specifically caught in class Socket.

ProtocolException(String)
Constructs a new ProtocolException with the specified detail message.



ProtocolException()
Constructs a new ProtocolException with no detail message.

ProtocolException

```
public ProtocolException(String host)
```

Constructs a new ProtocolException with the specified detail message. A detail message is a String that gives a specific description of this error.

Parameters:
host - the detail message

ProtocolException

```
public ProtocolException()
```

Constructs a new ProtocolException with no detail message. A detail message is a String that gives a specific description of this error.

Class java.net.ServerSocket

Class java.net.ServerSocket

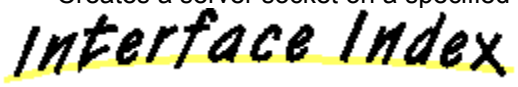
```
java.lang.Object
    |
    +----java.net.ServerSocket
```

---

public final class **ServerSocket**
extends Object

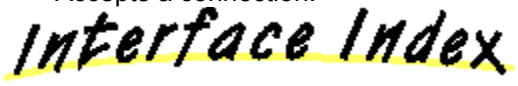
The server Socket class. It uses a SocketImpl to implement the actual socket operations. It is done this way so that you are able to change socket implementations depending on the kind of firewall being used. You can change socket implementations by setting the SocketImplFactory.

ServerSocket(int)
Creates a server socket on a specified port.



ServerSocket(int, int)
Creates a server socket, binds it to the specified local port and listens to it.

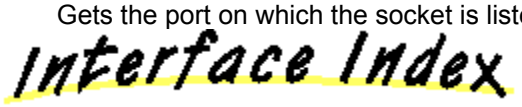
accept()
Accepts a connection.



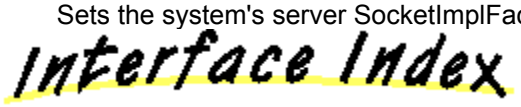
close()
Closes the server socket.

getInetAddress()
Gets the address to which the socket is connected.

getLocalPort()
Gets the port on which the socket is listening.



setSocketFactory(SocketImplFactory)
Sets the system's server SocketImplFactory.



toString()
Returns the implementation address and implementation port of this ServerSocket as a String.

ServerSocket

```
public ServerSocket(int port) throws IOException
```

Creates a server socket on a specified port.
Parameters:
port - the port
Throws: IOException
IO error when opening the socket.

ServerSocket

```
public ServerSocket(int port,
                    int count) throws IOException
```

Creates a server socket, binds it to the specified local port and listens to it. You can connect to an annonymous port by specifying the port number to be 0.
Parameters:
port - the specified port
count - the amountt of time to listen for a connection

getInetAddress

```
public InetAddress getInetAddress()
```

Gets the address to which the socket is connected.

getLocalPort

```
public int getLocalPort()
```

Gets the port on which the socket is listening.

accept

```
public Socket accept() throws IOException
```

Accepts a connection. This method will block until the connection is made.
Throws: IOException
IO error when waiting for the connection.

close

```
public void close() throws IOException
```

Closes the server socket.
Throws: IOException
IO error when closing the socket.

toString

```
public String toString()
```

Returns the implementation address and implementation port of this ServerSocket as a String.
Overrides:
toString in class Object

setSocketFactory

```
public static synchronized void setSocketFactory(SocketImplFactory fac)
throws IOException
```

Sets the system's server SocketImplFactory. The factory can be specified only once.

Parameters:

fac - the desired factory

Throws: SocketException

If the factory has already been defined.

Throws: IOException

IO error when setting the socket factor.

Class java.net.Socket

Class java.net.Socket

```
java.lang.Object
    |
    +----java.net.Socket
```

---

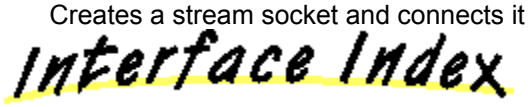
public final class **Socket**
extends Object

The client Socket class. It uses a SocketImpl to implement the actual socket operations. It is done this way so that you are able to change socket implementations depending on the kind of firewall that is used. You can change socket implementations by setting the SocketImplFactory.

Socket(String, int)
Creates a stream socket and connects it to the specified port on the specified host.

Socket(String, int, boolean)
Creates a socket and connects it to the specified port on the specified host.

Socket(InetAddress, int)
Creates a stream socket and connects it to the specified address on the specified port.

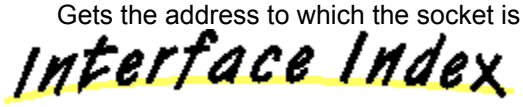


Socket(InetAddress, int, boolean)
Creates a socket and connects it to the specified address on the specified port.

close()
Closes the socket.

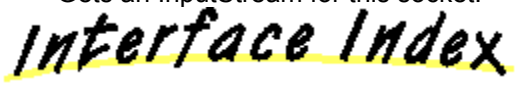
getInetAddress()

Gets the address to which the socket is connected.



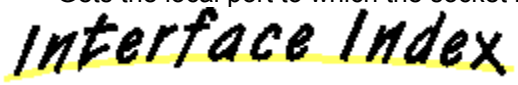
getInputStream()

Gets an InputStream for this socket.



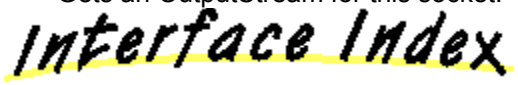
getLocalPort()

Gets the local port to which the socket is connected.



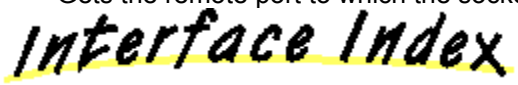
getOutputStream()

Gets an OutputStream for this socket.



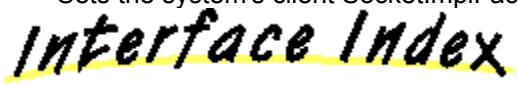
getPort()

Gets the remote port to which the socket is connected.



setSocketImplFactory(SocketImplFactory)

Sets the system's client SocketImplFactory.



toString()

Converts the Socket to a String.

Socket

```
public Socket(String host,
              int port) throws UnknownHostException, IOException
```

Creates a stream socket and connects it to the specified port on the specified host.

Parameters:

host - the host
port - the port

Socket

```
public Socket(String host,
              int port,
              boolean stream) throws IOException
```

Creates a socket and connects it to the specified port on the specified host. The last argument lets

you specify whether you want a stream or datagram socket.

Parameters:

host - the specified host
port - the specified port
stream - a boolean indicating whether this is a stream or datagram socket

Socket

```
public Socket(InetAddress address,
              int port) throws IOException
```

Creates a stream socket and connects it to the specified address on the specified port.

Parameters:

address - the specified address
port - the specified port

Socket

```
public Socket(InetAddress address,
              int port,
              boolean stream) throws IOException
```

Creates a socket and connects it to the specified address on the specified port. The last argument lets you specify whether you want a stream or datagram socket.

Parameters:

address - the specified address
port - the specified port
stream - a boolean indicating whether this is a stream or datagram socket

getInetAddress

```
public InetAddress getInetAddress()
```

Gets the address to which the socket is connected.

getPort

```
public int getPort()
```

Gets the remote port to which the socket is connected.

getLocalPort

```
public int getLocalPort()
```

Gets the local port to which the socket is connected.

getInputStream

```
public InputStream getInputStream() throws IOException
```

Gets an InputStream for this socket.

getOutputStream

```
public OutputStream getOutputStream() throws IOException
```

Gets an OutputStream for this socket.

close

```
public synchronized void close() throws IOException
```

Closes the socket.

toString

```
public String toString()
```

Converts the Socket to a String.

Overrides:

toString in class Object

setSocketImplFactory

```
  public static synchronized void setSocketImplFactory(SocketImplFactory fac)
throws IOException
```

Sets the system's client SocketImplFactory. The factory can be specified only once.

Parameters:

fac - the desired factory

Throws: SocketException

If the factory is already defined.

Class java.net.SocketException

Class java.net.SocketException

```
java.lang.Object
    |
    +----java.lang.Throwable
            |
            +----java.lang.Exception
                    |
                    +----java.io.IOException
                            |
                            +----java.net.SocketException
```

---

public class **SocketException**
extends IOException

Signals that an error occurred while attempting to use a socket.

Interface Index

Interface Index **SocketException**

```
public SocketException(String msg)
```

Constructs a new SocketException with the specified detail message. A detail message is a String that gives a specific description of this error.

Parameters:
msg - the detail message

SocketException

```
public SocketException()
```

Constructs a new SocketException with no detail message. A detail message is a String that gives a specific description of this error.

Class java.net.SocketImpl

Class java.net.SocketImpl

```
java.lang.Object
    |
    +----java.net.SocketImpl
```

---

public class **SocketImpl**
extends Object

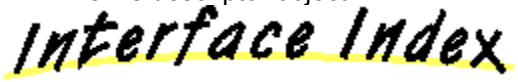
This is the Socket implementation class. It is an abstract class that must be subclassed to provide an actual implementation.

address

The internet address where the socket will make a connection.

fd

The file descriptor object



localport

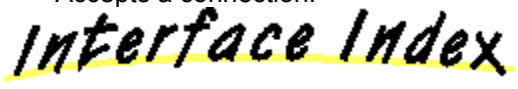
port

The port where the socket will make a connection.

SocketImpl()

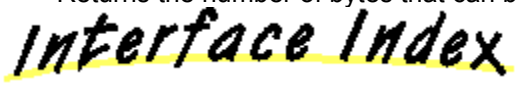
accept(SocketImpl)

Accepts a connection.



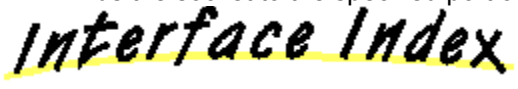
available()

Returns the number of bytes that can be read without blocking.



bind(InetAddress, int)

Binds the socket to the specified port on the specified host.



close()

Closes the socket.

connect(String, int)

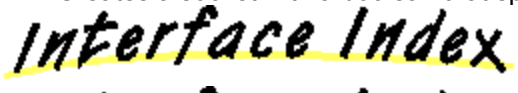
Connects the socket to the specified port on the specified host.

connect(InetAddress, int)

Connects the socket to the specified address on the specified port.

create(boolean)

Creates a socket with a boolean that specifies whether this is a stream socket or a datagram socket.



getFileDescriptor()

getInetAddress()

getInputStream()

Gets an InputStream for this socket.

getLocalPort()

getOutputStream()

Gets an OutputStream for this socket.

getPort()

listen(int)

Listens for connections over a specified amount of time.

toString()

Returns the address and port of this Socket as a String.

Interface Index

Interface Index **fd**

```
protected FileDescriptor fd
```

The file descriptor object

Interface Index **address**

```
protected InetAddress address
```

The internet address where the socket will make a connection.

Interface Index **port**

```
protected int port
```

The port where the socket will make a connection.

Interface Index **localport**

```
protected int localport
```

Interface Index

Interface Index **SocketImpl**

```
public SocketImpl()
```

Interface Index

Interface Index **create**

```
protected abstract void create(boolean stream) throws IOException
```

Creates a socket with a boolean that specifies whether this is a stream socket or a datagram socket.

Parameters:

stream - a boolean indicating whether this is a stream or datagram socket

connect

```
protected abstract void connect(String host,
                                int port) throws IOException
```

Connects the socket to the specified port on the specified host.

Parameters:

host - the specified host of the connection

port - the port where the connection is made

connect

```
protected abstract void connect(InetAddress address,
                                int port) throws IOException
```

Connects the socket to the specified address on the specified port.

Parameters:

address - the specified address of the connection

port - the specified port where connection is made

bind

```
protected abstract void bind(InetAddress host,
                             int port) throws IOException
```

Binds the socket to the specified port on the specified host.

Parameters:

host - the host

port - the port

listen

```
protected abstract void listen(int count) throws IOException
```

Listens for connections over a specified amount of time.
Parameters:
count - the amount of time this socket will listen for connections

accept

```
protected abstract void accept(SocketImpl s) throws IOException
```

Accepts a connection.
Parameters:
s - the accepted connection

getInputStream

```
protected abstract InputStream getInputStream() throws IOException
```

Gets an InputStream for this socket.

getOutputStream

```
protected abstract OutputStream getOutputStream() throws IOException
```

Gets an OutputStream for this socket.

available

```
protected abstract int available() throws IOException
```

Returns the number of bytes that can be read without blocking.

close

```
protected abstract void close() throws IOException
```

Closes the socket.

getFileDescriptor

```
protected FileDescriptor getFileDescriptor()
```

getInetAddress

```
protected InetAddress getInetAddress()
```

getPort

```
protected int getPort()
```

getLocalPort

```
protected int getLocalPort()
```

toString

```
public String toString()
```

Returns the address and port of this Socket as a String.

Overrides:

toString in class Object

Interface java.net.SocketImplFactory

Interface java.net.SocketImplFactory

public interface **SocketImplFactory**
extends Object

This interface defines a factory for SocketImpl instances. It is used by the socket class to create socket implementations that implement various policies.

createSocketImpl()
Creates a new SocketImpl instance.

createSocketImpl

```
public abstract SocketImpl createSocketImpl()
```

Creates a new SocketImpl instance.

Class java.net.UnknownHostException

Class java.net.UnknownHostException

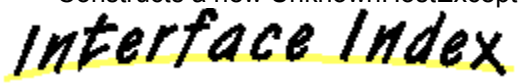
```
java.lang.Object
    |
    +----java.lang.Throwable
            |
            +----java.lang.Exception
                    |
                    +----java.io.IOException
                            |
                            +----java.net.UnknownHostException
```

---

public class **UnknownHostException**
extends IOException

Signals that the address of the server specified by a network client could not be resolved.

UnknownHostException(String)
Constructs a new UnknownHostException with the specified detail message.



UnknownHostException()
Constructs a new UnknownHostException with no detail message.

UnknownHostException

```
public UnknownHostException(String host)
```

Constructs a new UnknownHostException with the specified detail message. A detail message is a String that gives a specific description of this error.

Parameters:
host - the detail message

UnknownHostException

```
public UnknownHostException()
```

Constructs a new UnknownHostException with no detail message. A detail message is a String that gives a specific description of this error.

Class java.net.UnknownServiceException

Class java.net.UnknownServiceException

```
java.lang.Object
    |
    +----java.lang.Throwable
            |
            +----java.lang.Exception
                    |
                    +----java.io.IOException
                            |
                            +----java.net.UnknownServiceException
```

---

public class **UnknownServiceException**
extends IOException

Signals that an unknown service exception has occurred.

UnknownServiceException()
Constructs a new UnknownServiceException with no detail message.

UnknownServiceException(String)
Constructs a new UnknownServiceException with the specified detail message.

UnknownServiceException

```
public UnknownServiceException()
```

Constructs a new UnknownServiceException with no detail message. A detail message is a String that gives a specific description of this error.

UnknownServiceException

```
public UnknownServiceException(String msg)
```

Constructs a new UnknownServiceException with the specified detail message. A detail message is a String that gives a specific description of this error.

Parameters:

msg - the detail message

Class java.net.URL

Class java.net.URL

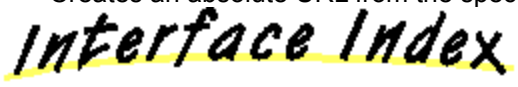
```
java.lang.Object
    |
    +----java.net.URL
```

---

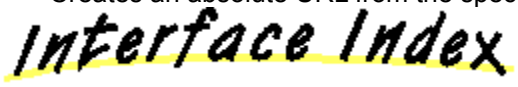
public final class **URL**
extends Object

Class URL represents a Uniform Reference Locator -- a reference to an object on the World Wide Web. This is a constant object, once it is created, its fields cannot be changed.

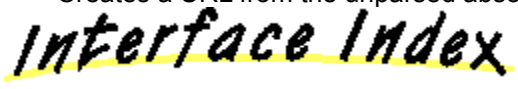
URL(String, String, int, String)
Creates an absolute URL from the specified protocol, host, port and file.



URL(String, String, String)
Creates an absolute URL from the specified protocol, host, and file.



URL(String)
Creates a URL from the unparsed absolute URL.

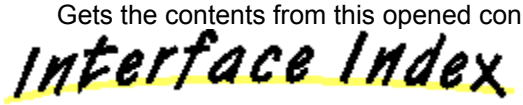


URL(URL, String)
Creates a URL from the unparsed URL in the specified context.If spec is an absolute URL it is used as is.

equals(Object)
Compares two URLs.

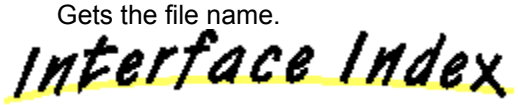
getContent()

Gets the contents from this opened connection.



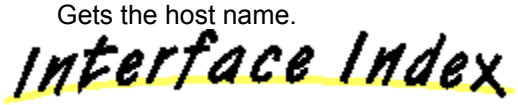
getFile()

Gets the file name.



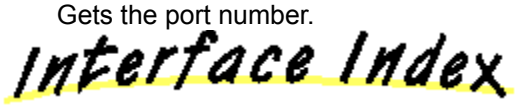
getHost()

Gets the host name.



getPort()

Gets the port number.



getProtocol()

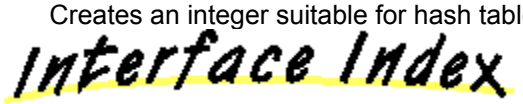
Gets the protocol name.

getRef()

Gets the ref.

hashCode()

Creates an integer suitable for hash table indexing.

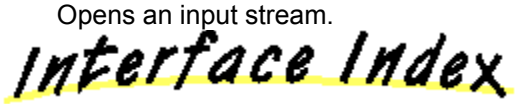


openConnection()

Creates (if not already in existance) a URLConnection object that contains a connection to the remote object referred to by the URL.

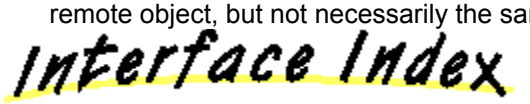
openStream()

Opens an input stream.



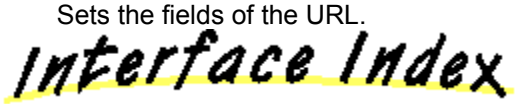
sameFile(URL)

Compares two URLs, excluding the "ref" fields: sameFile is true if the true references the same remote object, but not necessarily the same subpiece of that object.



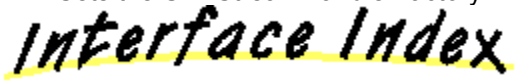
set(String, String, int, String, String)

Sets the fields of the URL.



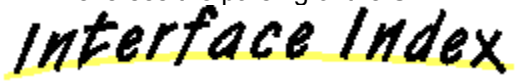
setURLStreamHandlerFactory(URLStreamHandlerFactory)

Sets the URLStreamHandler factory.



toExternalForm()

Reverses the parsing of the URL.



toString()

Converts to a human-readable form.

Interface Index

Interface Index **URL**

```
public URL(String protocol,
           String host,
           int port,
           String file) throws MalformedURLException
```

Creates an absolute URL from the specified protocol, host, port and file.

Parameters:

protocol - the protocol to use
host - the host to connect to
port - the port at that host to connect to
file - the file on that host

Throws: MalformedURLException

If an unknown protocol is found.

Interface Index **URL**

```
public URL(String protocol,
           String host,
           String file) throws MalformedURLException
```

Creates an absolute URL from the specified protocol, host, and file. The port number used will be the default for the protocol.

Parameters:

protocol - the protocol to use
host - the host to connect to
file - the file on that host

Throws: MalformedURLException

If an unknown protocol is found.

Interface Index **URL**

```
public URL(String spec) throws MalformedURLException
```

Creates a URL from the unparsed absolute URL.

Parameters:

spec - the URL String to parse

URL

```
public URL(URL context,
           String spec) throws MalformedURLException
```

Creates a URL from the unparsed URL in the specified context.If spec is an absolute URL it is used as is. Otherwise it isparsed in terms of the context. Context may be null (indicating no context).

Parameters:

context - the context to parse the URL to

spec - the URL String to parse

Throws: MalformedURLException

If the protocol is equal to null.

set

```
protected void set(String protocol,
                   String host,
                   int port,
                   String file,
                   String ref)
```

Sets the fields of the URL. This is not a public method so that only URLStreamHandlers can modify URL fields. URLs are otherwise constant. REMIND: this method will be moved to URLStreamHandler

Parameters:

protocol - the protocol to use

host - the host name to connecto to

port - the protocol port to connect to

file - the specified file name on that host

ref - the reference

getPort

```
public int getPort()
```

Gets the port number. Returns -1 if the port is not set.

getProtocol

```
public String getProtocol()
```

Gets the protocol name.

Interface Index **getHost**

```
public String getHost()
```

Gets the host name.

Interface Index **getFile**

```
public String getFile()
```

Gets the file name.

Interface Index **getRef**

```
public String getRef()
```

Gets the ref.

Interface Index **equals**

```
public boolean equals(Object obj)
```

Compares two URLs.

Parameters:

obj - the URL to compare against.

Returns:

true if and only if they are equal, false otherwise.

Overrides:

equals in class Object

Interface Index **hashCode**

```
public int hashCode()
```

Creates an integer suitable for hash table indexing.

Overrides:
hashCode in class Object

sameFile

```
public boolean sameFile(URL other)
```

Compares two URLs, excluding the "ref" fields: sameFile is true if the true references the same remote object, but not necessarily the same subpiece of that object.

Parameters:
other - the URL to compare against.

Returns:
true if and only if they are equal, false otherwise.

toString

```
public String toString()
```

Converts to a human-readable form.

Returns:
the textual representation.

Overrides:
toString in class Object

toExternalForm

```
public String toExternalForm()
```

Reverses the parsing of the URL.

Returns:
the textual representation of the fully qualified URL (i.e. after the context and canonicalization have been applied).

openConnection

```
public URLConnection openConnection() throws IOException
```

Creates (if not already in existance) a URLConnection object that contains a connection to the remote object referred to by the URL. Invokes the appropriate protocol handler. Failure is indicated by throwing an exception.

Throws: IOException

If an I/O exception has occurred.

See Also:

URLConnection

openStream

```
public final InputStream openStream() throws IOException
```

Opens an input stream.

Throws: IOException

If an I/O exception has occurred.

getContent

```
public final Object getContent() throws IOException
```

Gets the contents from this opened connection.

Throws: IOException

If an I/O exception has occurred.

setURLStreamHandlerFactory

```
public static synchronized void
setURLStreamHandlerFactory(URLStreamHandlerFactory fac)
```

Sets the URLStreamHandler factory.

Parameters:

fac - the desired factory

Throws: Error

If the factory has already been defined.

Class java.net.URLConnection

Class java.net.URLConnection

```
java.lang.Object
    |
    +----java.net.URLConnection
```

---

public class **URLConnection**
extends Object

A class to represent an active connection to an object represented by a URL. It is an abstract class that must be subclassed to implement a connection.

allowUserInteraction

connected

doInput

doOutput

ifModifiedSince

url

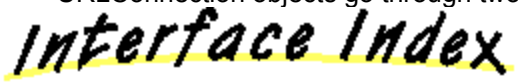
useCaches

URLConnection(URL)

Constructs a URL connection to the specified URL.

connect()

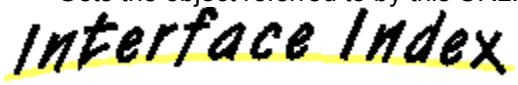
URLConnection objects go through two phases: first they are created, then they are connected.



getAllowUserInteraction()

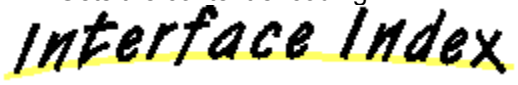
getContent()

Gets the object referred to by this URL.



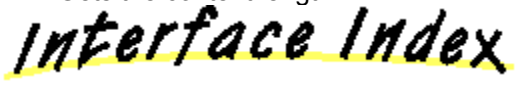
getContentEncoding()

Gets the content encoding.



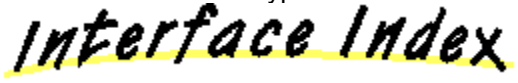
getContentLength()

Gets the content length.



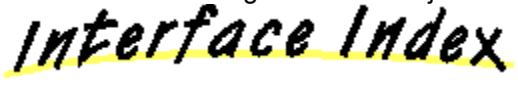
getContentType()

Gets the content type.



getDate()

Gets the sending date of the object.

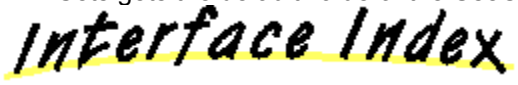


getDefaultAllowUserInteraction()

getDefaultRequestProperty (String)

getDefaultUseCaches()

Sets/gets the default value of the UseCaches flag.

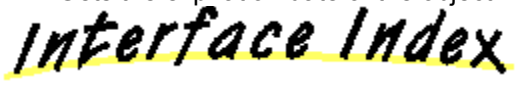


getDoInput()

getDoOutput()

getExpiration()

Gets the expriation date of the object.

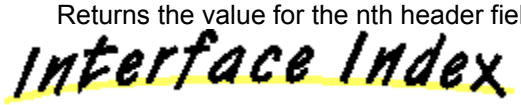


getHeaderField(String)

Gets a header field by name.

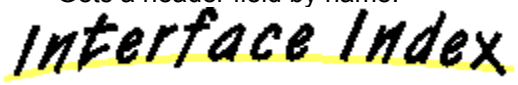
getHeaderField(int)

Returns the value for the nth header field.



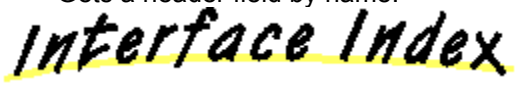
getHeaderFieldDate(String, long)

Gets a header field by name.



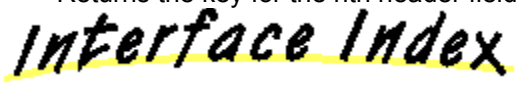
getHeaderFieldInt(String, int)

Gets a header field by name.



getHeaderFieldKey(int)

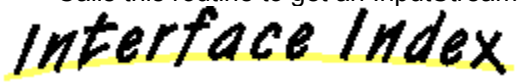
Returns the key for the nth header field.



getIfModifiedSince()

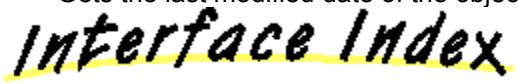
getInputStream()

Calls this routine to get an InputStream that reads from the object.



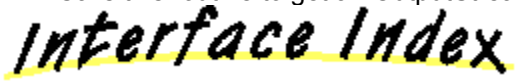
getLastModified()

Gets the last modified date of the object.



getOutputStream()

Calls this routine to get an OutputStream that writes to the object.



getRequestProperty(String)

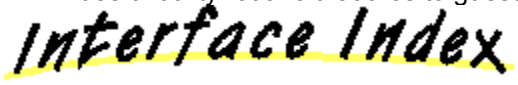
getURL()

Gets the URL for this connection.

getUseCaches()

guessContentTypeFromName (String)

A useful utility routine that tries to guess the content-type of an object based upon its extension.



guessContentTypeFromStream(InputStream)

This method is used to check for files that have some type that can be determined by inspection.

setAllowUserInteraction(boolean)

Some URL connections occasionally need to to interactions with the user.

setContentHandlerFactory(ContentHandlerFactory)

Sets the ContentHandler factory.

setDefaultAllowUserInteraction(boolean)

Sets/gets the default value of the allowUserInteraction flag.

Interface Index **setDefaultRequestProperty**(String, String)
Sets/gets the default value of a general request property.
Interface Index **setDefaultUseCaches**(boolean)
Interface Index **setDoInput**(boolean)
A URL connection can be used for input and/or output.
Interface Index **setDoOutput**(boolean)
A URL connection can be used for input and/or output.
Interface Index **setIfModifiedSince**(long)
Some protocols support skipping fetching unless the object is newer than some amount of time.
Interface Index **setRequestProperty**(String, String)
Sets/gets a general request property.
Interface Index **setUseCaches**(boolean)
Some protocols do caching of documents.
Interface Index **toString**()
Returns the String representation of the URL connection.

Interface Index

Interface Index **url**

```
protected URL url
```

Interface Index **doInput**

```
protected boolean doInput
```

Interface Index **doOutput**

```
protected boolean doOutput
```

Interface Index **allowUserInteraction**

```
protected boolean allowUserInteraction
```

Interface Index **useCaches**

```
protected boolean useCaches
```

Interface Index **ifModifiedSince**

```
protected long ifModifiedSince
```

Interface Index **connected**

```
protected boolean connected
```

Interface Index

Interface Index **URLConnection**

```
protected URLConnection(URL url)
```

Constructs a URL connection to the specified URL.
Parameters:
url - the specified URL

Interface Index

Interface Index **connect**

```
public abstract void connect() throws IOException
```

URLConnection objects go through two phases: first they are created, then they are connected. After being created, and before being connected, various options can be specified (eg. doInput,

UseCaches, ...). After connecting, it is an Error to try to set them. Operations that depend on being connected, like getContentLength, will implicitly perform the connection if necessary. Connecting when already connected does nothing.

getURL

```
public URL getURL()
```

Gets the URL for this connection.

getContentLength

```
public int getContentLength()
```

Gets the content length. Returns -1 if not known.

getContentType

```
public String getContentType()
```

Gets the content type. Returns null if not known.

getContentEncoding

```
public String getContentEncoding()
```

Gets the content encoding. Returns null if not known.

getExpiration

```
public long getExpiration()
```

Gets the expriation date of the object. Returns 0 if not known.

getDate

```
public long getDate()
```

Gets the sending date of the object. Returns 0 if not known.

Interface Index **getLastModified**

```
public long getLastModified()
```

Gets the last modified date of the object. Returns 0 if not known.

Interface Index **getHeaderField**

```
public String getHeaderField(String name)
```

Gets a header field by name. Returns null if not known.
Parameters:
name - the name of the header field

Interface Index **getHeaderFieldInt**

```
public int getHeaderFieldInt(String name,
                             int Default)
```

Gets a header field by name. Returns null if not known. The field is parsed as an integer. This form of getHeaderField exists because some connection types (e.g. http-ng) have pre-parsed headers and this allows them to override this method and short-circuit the parsing.
Parameters:
name - the name of the header field
Default - the value to return if the field is missing or malformed.

Interface Index **getHeaderFieldDate**

```
public long getHeaderFieldDate(String name,
                               long Default)
```

Gets a header field by name. Returns null if not known. The field will be parsed as a date. This form of getHeaderField exists because some connection types (eg. http-ng) have pre-parsed headers. This allows them to override this method and short-circuit the parsing.
Parameters:
name - the name of the header field
Default - the value to return if the field is missing or malformed.

getHeaderFieldKey

```
public String getHeaderFieldKey(int n)
```

Returns the key for the nth header field. Returns null if there are fewer than n fields. This can be used to iterate through all the headers in the message.

getHeaderField

```
public String getHeaderField(int n)
```

Returns the value for the nth header field. Returns null if there are fewer than n fields. This can be used in conjunction with getHeaderFieldKey to iterate through all the headers in the message.

getContent

```
public Object getContent() throws IOException
```

Gets the object referred to by this URL. For example, if it refers to an image the object will be some subclass of Image. The instanceof operator should be used to determine what kind of object was returned.

Returns:

the object that was fetched.

Throws: UnknownServiceException

If the protocol does not support content.

getInputStream

```
public InputStream getInputStream() throws IOException
```

Calls this routine to get an InputStream that reads from the object. Protocol implementors should use this if appropriate.

Throws: UnknownServiceException

If the protocol does not support input.

getOutputStream

```
public OutputStream getOutputStream() throws IOException
```

Calls this routine to get an OutputStream that writes to the object. Protocol implementors should use this if appropriate.

Throws: UnknownServiceException
If the protocol does not support output.

toString

```
public String toString()
```

Returns the String representation of the URL connection.

Overrides:
toString in class Object

setDoInput

```
public void setDoInput(boolean doinput)
```

A URL connection can be used for input and/or output. Set the DoInput flag to true if you intend to use the URL connection for input, false if not. The default is true unless DoOutput is explicitly set to true, in which case DoInput defaults to false.

getDoInput

```
public boolean getDoInput()
```

setDoOutput

```
public void setDoOutput(boolean dooutput)
```

A URL connection can be used for input and/or output. Set the DoOutput flag to true if you intend to use the URL connection for output, false if not. The default is false.

getDoOutput

```
public boolean getDoOutput()
```

setAllowUserInteraction

```
public void setAllowUserInteraction(boolean allowuserinteraction)
```

Some URL connections occasionally need to to interactions with the user. For example, the http protocol may need to pop up an authentication dialog. But this is only appropriate if the application is running in a context where there *is* a user. The allowUserInteraction flag allows these interactions when true. When it is false, they are not allowed and an exception is tossed. The default value can be set/gotten using setDefaultAllowUserInteraction, which defaults to false.

getAllowUserInteraction

```
public boolean getAllowUserInteraction()
```

setDefaultAllowUserInteraction

```
public static void setDefaultAllowUserInteraction(boolean
defaultallowuserinteraction)
```

Sets/gets the default value of the allowUserInteraction flag. This default is "sticky", being a part of the static state of all URLConnections. This flag applies to the next, and all following URLConnections that are created.

getDefaultAllowUserInteraction

```
public static boolean getDefaultAllowUserInteraction()
```

setUseCaches

```
public void setUseCaches(boolean usecaches)
```

Some protocols do caching of documents. Occasionally, it is important to be able to "tunnel through" and ignore the caches (e.g. the "reload" button in a browser). If the UseCaches flag on a connection is true, the connection is allowed to use whatever caches it can. If false, caches are to be ignored. The default value comes from DefaultUseCaches, which defaults to true.

getUseCaches

```
public boolean getUseCaches()
```

setIfModifiedSince

```
public void setIfModifiedSince(long ifmodifiedsince)
```

Some protocols support skipping fetching unless the object is newer than some amount of time. The ifModifiedSince field may be set/gotten to define this time.

getIfModifiedSince

```
public long getIfModifiedSince()
```

getDefaultUseCaches

```
public boolean getDefaultUseCaches()
```

Sets/gets the default value of the UseCaches flag. This default is "sticky", being a part of the static state of all URLConnections. This flag applies to the next, and all following, URLConnections that are created.

setDefaultUseCaches

```
public void setDefaultUseCaches(boolean defaultusecaches)
```

setRequestProperty

```
public void setRequestProperty(String key,
                               String value)
```

Sets/gets a general request property.

Parameters:

key - The keyword by which the request is known (eg "accept")

value - The value associated with it.

getRequestProperty

```
public String getRequestProperty(String key)
```

setDefaultRequestProperty

```
public static void setDefaultRequestProperty(String key,
                                              String value)
```

Sets/gets the default value of a general request property. When a URLConnection is created, it is initialized with these properties.

Parameters:

key - The keyword by which the request is known (eg "accept")

value - The value associated with it.

getDefaultRequestProperty

```
public static String getDefaultRequestProperty(String key)
```

setContentHandlerFactory

```
public static synchronized void
setContentHandlerFactory(ContentHandlerFactory fac)
```

Sets the ContentHandler factory.

Parameters:

fac - the desired factory

Throws: Error

If the factory has already been defined.

guessContentTypeFromName

```
protected static String guessContentTypeFromName(String fname)
```

A useful utility routine that tries to guess the content-type of an object based upon its extension.

guessContentTypeFromStream

```
protected static String guessContentTypeFromStream(InputStream is) throws
IOException
```

This method is used to check for files that have some type that can be determined by inspection. The bytes at the beginning of the file are examined loosely. In an ideal world, this routine would not be needed, but in a world where http servers lie about content-types and extensions are often non-standard, direct inspection of the bytes can make the system more robust. The stream must support marks (e.g. have a BufferedInputStream somewhere).

Class java.net.URLEncoder

Class java.net.URLEncoder

```
java.lang.Object
   |
   +----java.net.URLEncoder
```

---

public class **URLEncoder**
extends Object

Turns Strings of text into x-www-form-urlencoded format.

encode(String)
Translates String into x-www-form-urlencoded format.

encode

```
public static String encode(String s)
```

Translates String into x-www-form-urlencoded format.
Parameters:
s - String to be translated
Returns:
the translated String.

Class java.net.URLStreamHandler

Class java.net.URLStreamHandler

```
java.lang.Object
    |
    +----java.net.URLStreamHandler
```

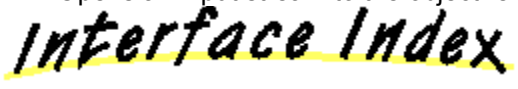
---

public class **URLStreamHandler**
extends Object

Abstract class for URL stream openers. Subclasses of this class know how to create streams for particular protocol types.

URLStreamHandler()

openConnection(URL)
Opens an input stream to the object referenced by the URL.



parseURL(URL, String, int, int)
This method is called to parse the string spec into URL u.

setURL(URL, String, String, int, String, String)
Calls the (protected) set method out of the URL given.

toExternalForm(URL)
Reverses the parsing of the URL.


URLStreamHandler

```
public URLStreamHandler()
```


openConnection

```
protected abstract URLConnection openConnection(URL u) throws IOException
```

Opens an input stream to the object referenced by the URL. This method should be overridden by a subclass.

Parameters:

u - the URL that this connects to


parseURL

```
protected void parseURL(URL u,
                        String spec,
                        int start,
                        int limit)
```

This method is called to parse the string spec into URL u. If there is any inherited context then it has already been copied into u. The parameters `start` and `limit` refer to the range of characters in spec that should be parsed. The default method uses parsing rules that match the http spec, which most URL protocol families follow. If you are writing a protocol handler that has a different syntax, override this routine.

Parameters:

u - the URL to receive the result of parsing the spec

spec - the URL string to parse

start - the character position to start parsing at. This is just past the ':' (if there is one).

limit - the character position to stop parsing at. This is the end of the string or the position of the "#" character if present (the "#" reference syntax is protocol independent).


toExternalForm

```
protected String toExternalForm(URL u)
```

Reverses the parsing of the URL. This should probably be overridden if you override parseURL().

Parameters:

u - the URL

Returns:

the textual representation of the fully qualified URL (i.e. after the context and canonicalization have been applied).


```
protected void setURL(URL u,
                      String protocol,
                      String host,
                      int port,
                      String file,
                      String ref)
```

Calls the (protected) set method out of the URL given. Only classes derived from URLStreamHandler are supposed to be able to call the set() method on a URL.

See Also:

set

Interface java.net.URLStreamHandlerFactory

Interface java.net.URLStreamHandlerFactory

public interface **URLStreamHandlerFactory**
extends Object

This interface defines a factory for URLStreamHandler instances. It is used by the URL class to create URLStreamHandlers for various streams.

createURLStreamHandler(String)
Creates a new URLStreamHandler instance with the specified protocol.

createURLStreamHandler

```
public abstract URLStreamHandler createURLStreamHandler(String protocol)
```

Creates a new URLStreamHandler instance with the specified protocol.

Parameters:

protocol - the protocol to use (ftp, http, nntp, etc.)

Class java.util.BitSet

Class java.util.BitSet

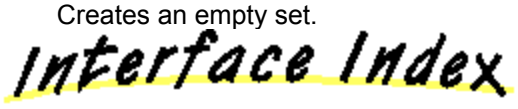
```
java.lang.Object
    |
    +----java.util.BitSet
```

---

public final class **BitSet**
extends Object
implements Cloneable

A set of bits. The set automatically grows as more bits are needed.

BitSet()
Creates an empty set.



BitSet(int)
Creates an empty set with the specified size.

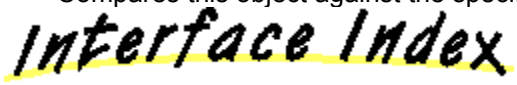
and(BitSet)
Logically ANDs this bit set with the specified set of bits.

clear(int)
Clears a bit.

clone()
Clones the BitSet.

equals(Object)

Compares this object against the specified object.



get(int)

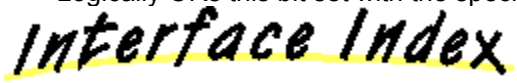
Gets a bit.

hashCode()

Gets the hashcode.

or(BitSet)

Logically ORs this bit set with the specified set of bits.



set(int)

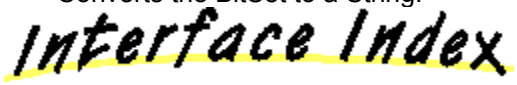
Sets a bit.

size()

Calculates and returns the set's size

toString()

Converts the BitSet to a String.



xor(BitSet)

Logically XORs this bit set with the specified set of bits.

BitSet

```
public BitSet()
```

Creates an empty set.

BitSet

```
public BitSet(int nbits)
```

Creates an empty set with the specified size.

Parameters:

nbits - the size of the set

Interface Index set

```
public void set(int bit)
```

Sets a bit.
Parameters:
bit - the bit to be set

Interface Index clear

```
public void clear(int bit)
```

Clears a bit.
Parameters:
bit - the bit to be cleared

Interface Index get

```
public boolean get(int bit)
```

Gets a bit.
Parameters:
bit - the bit to be gotten

Interface Index and

```
public void and(BitSet set)
```

Logically ANDs this bit set with the specified set of bits.
Parameters:
set - the bit set to be ANDed with

Interface Index **or**

```
public void or(BitSet set)
```

Logically ORs this bit set with the specified set of bits.
Parameters:
set - the bit set to be ORed with

Interface Index **xor**

```
public void xor(BitSet set)
```

Logically XORs this bit set with the specified set of bits.
Parameters:
set - the bit set to be XORed with

Interface Index **hashCode**

```
public int hashCode()
```

Gets the hashcode.
Overrides:
hashCode in class Object

Interface Index **size**

```
public int size()
```

Calculates and returns the set's size

Interface Index **equals**

```
public boolean equals(Object obj)
```

Compares this object against the specified object.
Parameters:
obj - the object to commpare with

Returns:
true if the objects are the same; false otherwise.
Overrides:
equals in class Object

clone

```
public Object clone()
```

Clones the BitSet.
Overrides:
clone in class Object

toString

```
public String toString()
```

Converts the BitSet to a String.
Overrides:
toString in class Object

Class java.util.Date

Class java.util.Date

```
java.lang.Object
   |
   +----java.util.Date
```

---

public class **Date**
extends Object

A wrapper for a date. This class lets you manipulate dates in a system independent way. To print today's date use:

```
Date d = new Date();
System.out.println("today = " + d);
```

To find out what day corresponds to a particular date:

```
Date d = new Date(63, 0, 16);   // January 16, 1963
System.out.println("Day of the week: " + d.getDay());
```

The date can be set and examined according to the local time zone into the year, month, day, hour, minute and second.

While the API is intended to reflect UTC, Coordinated Universal Time, it doesn't do so exactly. This inexact behavior is inherited from the time system of the underlying OS. All modern OS's that I (jag) am aware of assume that 1 day = 24*60*60 seconds. In UTC, about once a year there is an extra second, called a "leap second" added to a day to account for the wobble of the earth. Most computer clocks are not accurate enough to be able to reflect this distinction. Some computer standards are defined in GMT, which is equivalent to UT, Universal Time. GMT is the "civil" name for the standard, UT is the "scientific" name for the same standard. The distinction between UTC and UT is that the first is based on an atomic clock and the second is based on astronomical observations, which for all practical purposes is an invisibly fine hair to split. An interesting source of further information is the US Naval Observatory, particularly the **http://tycho.usno.navy.mil Directorate of Time** and their definitions of **http://tycho.usno.navy.mil/systime.html Systems of Time**.

Date()
Creates today's date/time.

Date(long)

Creates a date.

Date(int, int, int)

Creates a date.

Date(int, int, int, int, int)

Creates a date.

Date(int, int, int, int, int, int)

Creates a date.

Date(String)

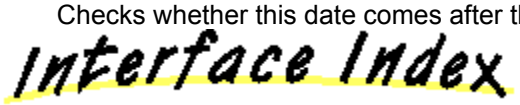
Creates a date from a string according to the syntax accepted by parse().

UTC(int, int, int, int, int, int)

Calculates a UTC value from YMDHMS.

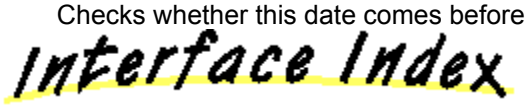
after(Date)

Checks whether this date comes after the specified date.



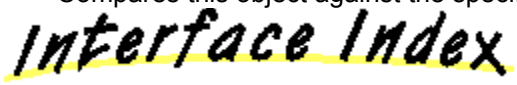
before(Date)

Checks whether this date comes before the specified date.



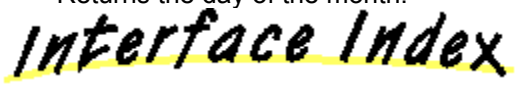
equals(Object)

Compares this object against the specified object.



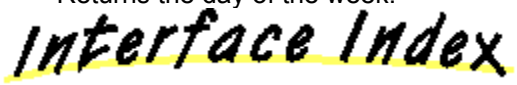
getDate()

Returns the day of the month.



getDay()

Returns the day of the week.



getHours()

Returns the hour.

getMinutes()

Returns the minute.

getMonth()

Returns the month.

getSeconds()

Returns the second.

getTime()

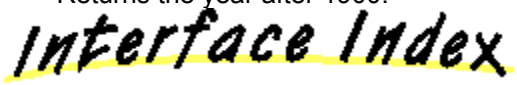
Returns the time in milliseconds since the epoch.

getTimezoneOffset()

Return the time zone offset in minutes for the current locale that is appropriate for this time.

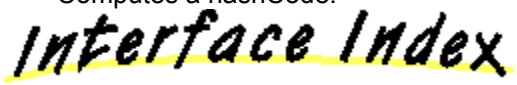
getYear()

Returns the year after 1900.



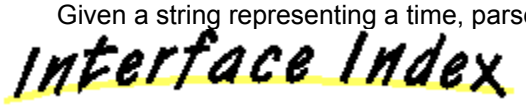
hashCode()

Computes a hashCode.



parse(String)

Given a string representing a time, parse it and return the time value.



setDate(int)

Sets the date.

setHours(int)

Sets the hours.

setMinutes(int)

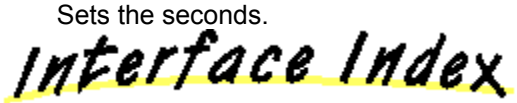
Sets the minutes.

setMonth(int)

Sets the month.

setSeconds(int)

Sets the seconds.



setTime(long)

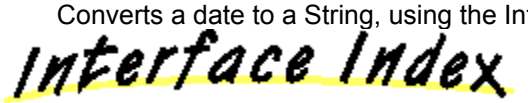
Sets the time.

setYear(int)

Sets the year.

toGMTString()

Converts a date to a String, using the Internet GMT conventions.



toLocaleString()

Converts a date to a String, using the locale conventions.

toString()

Converts a date to a String, using the UNIX ctime conventions.

Date

```
public Date()
```

Creates today's date/time.

Date

```
public Date(long date)
```

Creates a date. The fields are normalized before the Date object is created. The argument does not have to be in the correct range. For example, the 32nd of January is correctly interpreted as the 1st of February. You can use this to figure out what day a particular date falls on.

Parameters:

date - the value of the argument to be created

Date

```
public Date(int year,
            int month,
            int date)
```

Creates a date. The fields are normalized before the Date object is created. The arguments do not have to be in the correct range. For example, the 32nd of January is correctly interpreted as the 1st of February. You can use this to figure out what day a particular date falls on.

Parameters:

year - a year after 1900

month - a month between 0-11

date - day of the month between 1-31

Date

```
public Date(int year,
```

```
            int month,
            int date,
            int hrs,
            int min)
```

Creates a date. The fields are normalized before the Date object is created. The arguments do not have to be in the correct range. For example, the 32nd of January is correctly interpreted as the 1st of February. You can use this to figure out what day a particular date falls on.

Parameters:

year - a year after 1900
month - a month between 0-11
date - day of the month between 1-31
hrs - hours between 0-23
min - minutes between 0-59

Date

```
public Date(int year,
            int month,
            int date,
            int hrs,
            int min,
            int sec)
```

Creates a date. The fields are normalized before the Date object is created. The arguments do not have to be in the correct range. For example, the 32nd of January is correctly interpreted as the 1st of February. You can use this to figure out what day a particular date falls on.

Parameters:

year - a year after 1900
month - a month between 0-11
date - day of the month between 1-31
hrs - hours between 0-23
min - minutes between 0-59
sec - seconds between 0-59

Date

```
public Date(String s)
```

Creates a date from a string according to the syntax accepted by parse().

Interface Index **UTC**

```
public static long UTC(int year,
                       int month,
                       int date,
                       int hrs,
                       int min,
                       int sec)
```

Calculates a UTC value from YMDHMS. Interpretes the parameters in UTC, *not in the local time zone.*

Parameters:

year - a year after 1900
month - a month between 0-11
date - day of the month between 1-31
hrs - hours between 0-23
min - minutes between 0-59
sec - seconds between 0-59

Interface Index ***parse***

```
public static long parse(String s)
```

Given a string representing a time, parse it and return the time value. It accepts many syntaxes, but most importantly, in accepts the IETF standard date syntax: "Sat, 12 Aug 1995 13:30:00 GMT". It understands the continental US time zone abbreviations, but for general use, a timezone offset should be used: "Sat, 12 Aug 1995 13:30:00 GMT+0430" (4 hours, 30 minutes west of the Greenwich meridian). If no time zone is specified, the local time zone is assumed. GMT and UTC are considered equivalent.

Interface Index ***getYear***

```
public int getYear()
```

Returns the year after 1900.

Interface Index ***setYear***

```
public void setYear(int year)
```

Sets the year.

Parameters:

year - the year value

getMonth

```
public int getMonth()
```

Returns the month. This method assigns months with the values 0-11, with January beginning at value 0.

setMonth

```
public void setMonth(int month)
```

Sets the month.
Parameters:
month - the month value (0-11)

getDate

```
public int getDate()
```

Returns the day of the month. This method assigns days with the values of 1 to 31.

setDate

```
public void setDate(int date)
```

Sets the date.
Parameters:
date - the day value

getDay

```
public int getDay()
```

Returns the day of the week. This method assigns days of the week with the values 0-6, with 0 being Sunday.

getHours

```
public int getHours()
```

Returns the hour. This method assigns the value of the hours of the day to range from 0 to 23, with midnight equal to 0.

setHours

```
public void setHours(int hours)
```

Sets the hours.
Parameters:
hours - the hour value

getMinutes

```
public int getMinutes()
```

Returns the minute. This method assigns the minutes of an hour to be any value from 0 to 59.

setMinutes

```
public void setMinutes(int minutes)
```

Sets the minutes.
Parameters:
minutes - the value of the minutes

getSeconds

```
public int getSeconds()
```

Returns the second. This method assigns the seconds of a minute to values of 0-59.

setSeconds

```
public void setSeconds(int seconds)
```

Sets the seconds.
Parameters:
seconds - the second value

Interface Index **getTime**

```
public long getTime()
```

Returns the time in milliseconds since the epoch.

Interface Index **setTime**

```
public void setTime(long time)
```

Sets the time.
Parameters:
time - The new time value in milliseconds since the epoch.

Interface Index **before**

```
public boolean before(Date when)
```

Checks whether this date comes before the specified date.
Parameters:
when - the date to compare
Returns:
true if the original date comes before the specified one; false otherwise.

Interface Index **after**

```
public boolean after(Date when)
```

Checks whether this date comes after the specified date.
Parameters:
when - the date to compare
Returns:

true if the original date comes after the specified one; false otherwise.

equals

```
public boolean equals(Object obj)
```

Compares this object against the specified object.

Parameters:

obj - the object to compare with

Returns:

true if the objects are the same; false otherwise.

Overrides:

equals in class Object

hashCode

```
public int hashCode()
```

Computes a hashCode.

Overrides:

hashCode in class Object

toString

```
public String toString()
```

Converts a date to a String, using the UNIX ctime conventions.

Overrides:

toString in class Object

toLocaleString

```
public String toLocaleString()
```

Converts a date to a String, using the locale conventions.

toGMTString

```
public String toGMTString()
```

Converts a date to a String, using the Internet GMT conventions.

getTimezoneOffset

```
public int getTimezoneOffset()
```

Return the time zone offset in minutes for the current locale that is appropriate for this time. This value would be a constant except for daylight savings time.

Class java.util.Dictionary

Class java.util.Dictionary

```
java.lang.Object
    |
    +----java.util.Dictionary
```

---

public class **Dictionary**
extends Object

The Dictionary class is the abstract parent of Hashtable, which maps keys to values. Any object can be used as a key and/or value.

See Also:

Hashtable, hashCode, equals

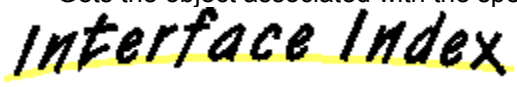
Dictionary()

elements()

Returns an enumeration of the elements.

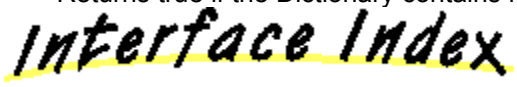
get(Object)

Gets the object associated with the specified key in the Dictionary.



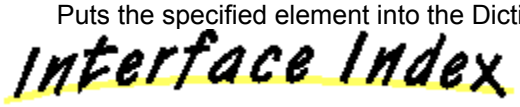
isEmpty()

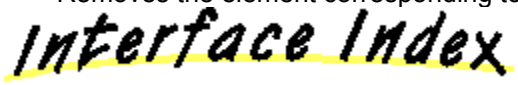
Returns true if the Dictionary contains no elements.



keys()

Returns an enumeration of the Dictionary's keys.

put(Object, Object)
Puts the specified element into the Dictionary, using the specified key.


remove(Object)
Removes the element corresponding to the key.


size()
Returns the number of elements contained within the Dictionary.

Dictionary

```
public Dictionary()
```

size

```
public abstract int size()
```

Returns the number of elements contained within the Dictionary.

isEmpty

```
public abstract boolean isEmpty()
```

Returns true if the Dictionary contains no elements.

keys

```
public abstract Enumeration keys()
```

Returns an enumeration of the Dictionary's keys.
See Also:
elements, Enumeration

elements

```
public abstract Enumeration elements()
```

Returns an enumeration of the elements. Use the Enumeration methods on the returned object to fetch the elements sequentially.

See Also:

keys, Enumeration

get

```
public abstract Object get(Object key)
```

Gets the object associated with the specified key in the Dictionary.

Parameters:

key - the key in the hash table

Returns:

s the element for the key, or null if the key is not defined in the hash table.

See Also:

put

put

```
public abstract Object put(Object key,
                           Object value)
```

Puts the specified element into the Dictionary, using the specified key. The element may be retrieved by doing a get() with the same key. The key and the element cannot be null.

Parameters:

key - the specified hashtable key
value - the specified element

Returns:

the old value of the key, or null if it did not have one.

Throws: NullPointerException

If the value of the specified element is null.

See Also:

get

remove

```
public abstract Object remove(Object key)
```

Removes the element corresponding to the key. Does nothing if the key is not present.

Parameters:

key - the key that needs to be removed

Returns:

the value of key, or null if the key was not found.

Class java.util.EmptyStackException

Class java.util.EmptyStackException

```
java.lang.Object
    |
    +----java.lang.Throwable
            |
            +----java.lang.Exception
                    |
                    +----java.lang.RuntimeException
                            |
                            +----java.util.EmptyStackException
```

---

public class **EmptyStackException**
extends RuntimeException

Signals that the stack is empty.
See Also:
Stack

Interface Index

Interface Index

EmptyStackException()
Constructs a new EmptyStackException with no detail message.

Interface Index

Interface Index

EmptyStackException

```
public EmptyStackException()
```

Constructs a new EmptyStackException with no detail message. A detail message is a String that describes the exception.

Interface java.util.Enumeration

Interface java.util.Enumeration

public interface **Enumeration**
extends Object

The Enumeration interface specifies a set of methods that may be used to enumerate, or count through, a set of values. The enumeration is consumed by use; its values may only be counted once.

For example, to print all elements of a Vector v:

```
for (Enumeration e = v.elements() ; e.hasMoreElements() ;) {
    System.out.println(e.nextElement());
}
```

See Also:
Vector, Hashtable

hasMoreElements()
Returns true if the enumeration contains more elements; false if its empty.

nextElement()
Returns the next element of the enumeration.

hasMoreElements

```
public abstract boolean hasMoreElements()
```

Returns true if the enumeration contains more elements; false if its empty.

nextElement

```
public abstract Object nextElement()
```

Returns the next element of the enumeration. Calls to this method will enumerate successive elements.

Throws: NoSuchElementException

If no more elements exist.

Class java.util.Hashtable

Class java.util.Hashtable

```
java.lang.Object
    |
    +----java.util.Dictionary
            |
            +----java.util.Hashtable
```

---

public class **Hashtable**
extends Dictionary
implements Cloneable

Hashtable class. Maps keys to values. Any object can be used as a key and/or value.

To sucessfully store and retrieve objects from a hash table, the object used as the key must implement the hashCode() and equals() methods.

This example creates a hashtable of numbers. It uses the names of the numbers as keys:

```
Hashtable numbers = new Hashtable();
numbers.put("one", new Integer(1));
numbers.put("two", new Integer(2));
numbers.put("three", new Integer(3));
```

To retrieve a number use:

```
Integer n = (Integer)numbers.get("two");
if (n != null) {
    System.out.println("two = " + n);
}
```

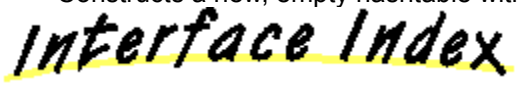
See Also:
hashCode, equals

Interface Index

Interface Index

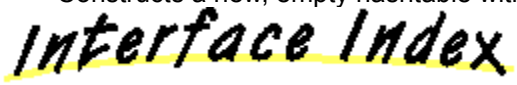
Hashtable(int, float)

Constructs a new, empty hashtable with the specified initial capacity and the specified load factor.



Hashtable(int)

Constructs a new, empty hashtable with the specified initial capacity.



Hashtable()

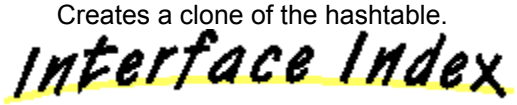
Constructs a new, empty hashtable.

clear()

Clears the hash table so that it has no more elements in it.

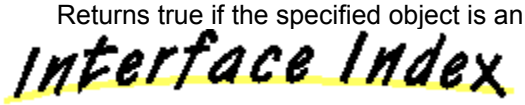
clone()

Creates a clone of the hashtable.



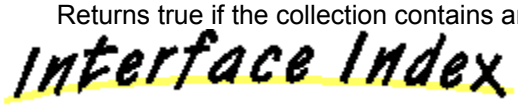
contains(Object)

Returns true if the specified object is an element of the hashtable.



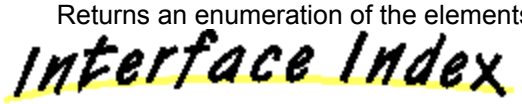
containsKey(Object)

Returns true if the collection contains an element for the key.



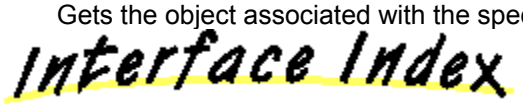
elements()

Returns an enumeration of the elements.



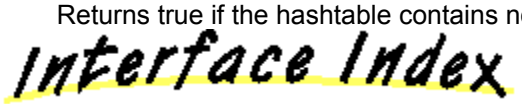
get(Object)

Gets the object associated with the specified key in the hashtable.



isEmpty()

Returns true if the hashtable contains no elements.

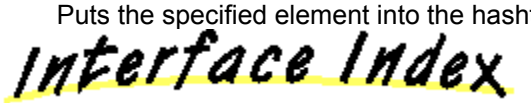


keys()

Returns an enumeration of the hashtable's keys.

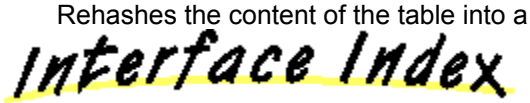
put(Object, Object)

Puts the specified element into the hashtable, using the specified key.



rehash()

Rehashes the content of the table into a bigger table.

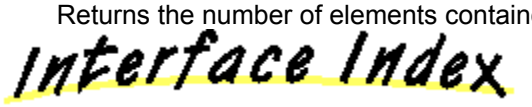


remove(Object)

Removes the element corresponding to the key.

size()

Returns the number of elements contained in the hashtable.



toString()

Converts to a rather lengthy String.

Hashtable

```
public Hashtable(int initialCapacity,
                 float loadFactor)
```

Constructs a new, empty hashtable with the specified initial capacity and the specified load factor.

Parameters:

initialCapacity - the initial number of buckets

loadFactor - a number between 0.0 and 1.0, it defines the threshold for rehashing the hashtable into a bigger one.

Throws: IllegalArgumentException

If the initial capacity is less than or equal to zero.

Throws: IllegalArgumentException

If the load factor is less than or equal to zero.

Hashtable

```
public Hashtable(int initialCapacity)
```

Constructs a new, empty hashtable with the specified initial capacity.

Parameters:

initialCapacity - the initial number of buckets

Hashtable

```
public Hashtable()
```

Constructs a new, empty hashtable. A default capacity and load factor is used. Note that the hashtable will automatically grow when it gets full.

Interface Index **size**

```
public int size()
```

Returns the number of elements contained in the hashtable.
Overrides:
size in class Dictionary

Interface Index **isEmpty**

```
public boolean isEmpty()
```

Returns true if the hashtable contains no elements.
Overrides:
isEmpty in class Dictionary

Interface Index **keys**

```
public synchronized Enumeration keys()
```

Returns an enumeration of the hashtable's keys.
Overrides:
keys in class Dictionary
See Also:
elements, Enumeration

Interface Index **elements**

```
public synchronized Enumeration elements()
```

Returns an enumeration of the elements. Use the Enumeration methods on the returned object to fetch the elements sequentially.
Overrides:
elements in class Dictionary
See Also:
keys, Enumeration

Interface Index **contains**

```
public synchronized boolean contains(Object value)
```

Returns true if the specified object is an element of the hashtable. This operation is more expensive than the containsKey() method.

Parameters:

value - the value that we are looking for

Throws: NullPointerException

If the value being searched for is equal to null.

See Also:

containsKey

Interface Index **containsKey**

```
public synchronized boolean containsKey(Object key)
```

Returns true if the collection contains an element for the key.

Parameters:

key - the key that we are looking for

See Also:

contains

Interface Index **get**

```
public synchronized Object get(Object key)
```

Gets the object associated with the specified key in the hashtable.

Parameters:

key - the specified key

Returns:

s the element for the key or null if the key is not defined in the hash table.

Overrides:

get in class Dictionary

See Also:

put

rehash

```
protected void rehash()
```

Rehashes the content of the table into a bigger table. This method is called automatically when the hashtable's size exceeds the threshold.

put

```
public synchronized Object put(Object key,
                               Object value)
```

Puts the specified element into the hashtable, using the specified key. The element may be retrieved by doing a get() with the same key. The key and the element cannot be null.

Parameters:

key - the specified key in the hashtable
value - the specified element

Returns:

the old value of the key, or null if it did not have one.

Throws: NullPointerException

If the value of the element is equal to null.

Overrides:

put in class Dictionary

See Also:

get

remove

```
public synchronized Object remove(Object key)
```

Removes the element corresponding to the key. Does nothing if the key is not present.

Parameters:

key - the key that needs to be removed

Returns:

the value of key, or null if the key was not found.

Overrides:

remove in class Dictionary

clear

```
public synchronized void clear()
```

Clears the hash table so that it has no more elements in it.

clone

```
public synchronized Object clone()
```

Creates a clone of the hashtable. A shallow copy is made, the keys and elements themselves are NOT cloned. This is a relatively expensive operation.

Overrides:

clone in class Object

toString

```
public synchronized String toString()
```

Converts to a rather lengthy String.

Overrides:

toString in class Object

Class java.util.NoSuchElementException

Class java.util.NoSuchElementException

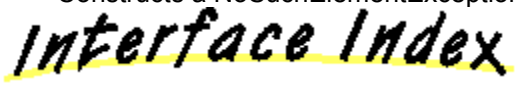
```
java.lang.Object
    |
    +----java.lang.Throwable
            |
            +----java.lang.Exception
                    |
                    +----java.lang.RuntimeException
                            |
                            +----java.util.NoSuchElementException
```

---

public class **NoSuchElementException**
extends RuntimeException

Signals that an enumeration is empty.
See Also:
Enumeration

NoSuchElementException()
Constructs a NoSuchElementException with no detail message.



NoSuchElementException(String)
Constructs a NoSuchElementException with the specified detail message.

NoSuchElementException

```
public NoSuchElementException()
```

Constructs a NoSuchElementException with no detail message. A detail message is a String that describes this particular exception.


```
public NoSuchElementException(String s)
```

Constructs a NoSuchElementException with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class java.util.Observable

Class java.util.Observable

```
java.lang.Object
    |
    +----java.util.Observable
```

---

public class **Observable**
extends Object

This class should be subclassed by observable object, or "data" in the Model-View paradigm. An Observable object may have any number of Observers. Whenever the Observable instance changes, it notifies all of its observers. Notification is done by calling the update() method on all observers.

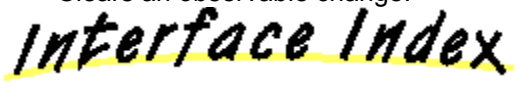
Observable()

addObserver(Observer)

Adds an observer to the observer list.

clearChanged()

Clears an observable change.



countObservers()

Counts the number of observers.

deleteObserver(Observer)

Deletes an observer from the observer list.

Interface Index **deleteObservers**()
Deletes observers from the observer list.
Interface Index **hasChanged**()
Returns a true boolean if an observable change has occurred.
Interface Index **notifyObservers**()
Notifies all observers if an observable change occurs.
Interface Index **notifyObservers**(Object)
Notifies all observers of the specified observable change which occurred.
Interface Index **setChanged**()
Sets a flag to note an observable change.

Interface Index

Interface Index **Observable**

```
public Observable()
```

Interface Index

Interface Index **addObserver**

```
public synchronized void addObserver(Observer o)
```

Adds an observer to the observer list.
Parameters:
o - the observer to be added

Interface Index **deleteObserver**

```
public synchronized void deleteObserver(Observer o)
```

Deletes an observer from the observer list.
Parameters:

o - the observer to be deleted

notifyObservers

```
public void notifyObservers()
```

Notifies all observers if an observable change occurs.

notifyObservers

```
public synchronized void notifyObservers(Object arg)
```

Notifies all observers of the specified observable change which occurred.

Parameters:

arg - what is being notified

deleteObservers

```
public synchronized void deleteObservers()
```

Deletes observers from the observer list.

setChanged

```
protected synchronized void setChanged()
```

Sets a flag to note an observable change.

clearChanged

```
protected synchronized void clearChanged()
```

Clears an observable change.

hasChanged

```
public synchronized boolean hasChanged()
```

Returns a true boolean if an observable change has occurred.

countObservers

```
public synchronized int countObservers()
```

Counts the number of observers.

Interface java.util.Observer

Interface java.util.Observer

public interface **Observer**
extends Object

When implemented, this interface allows all classes to be observable by instances of class Observer.

update(Observable, Object)
Called when observers in the observable list need to be updated.

update

```
public abstract void update(Observable o,
                            Object arg)
```

Called when observers in the observable list need to be updated.

Parameters:

o - the list of observers
arg - the argument being notified

Class java.util.Properties

Class java.util.Properties

```
java.lang.Object
    |
    +----java.util.Dictionary
            |
            +----java.util.Hashtable
                    |
                    +----java.util.Properties
```

---

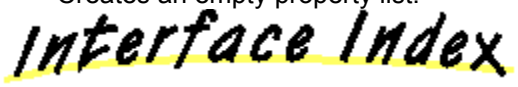
public class **Properties**
extends Hashtable

Persistent properties class. This class is basically a hashtable that can be saved/loaded from a stream. If a property is not found, a property list containing defaults is searched. This allows arbitrary nesting.

defaults

Properties()

Creates an empty property list.

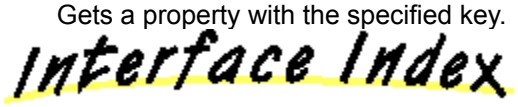


Properties(Properties)

Creates an empty property list with specified defaults.

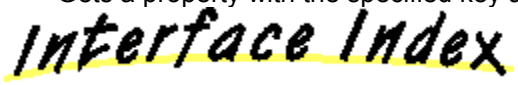
getProperty(String)

Gets a property with the specified key.



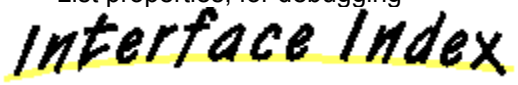
getProperty(String, String)

Gets a property with the specified key and default.



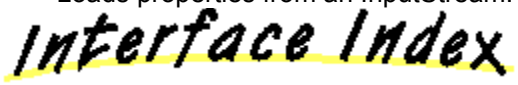
list(PrintStream)

List properties, for debugging



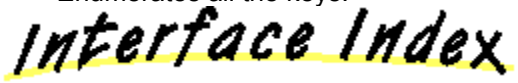
load(InputStream)

Loads properties from an InputStream.



propertyNames()

Enumerates all the keys.



save(OutputStream, String)

Save properties to an OutputStream.

defaults

```
protected Properties defaults
```

Properties

```
public Properties()
```

Creates an empty property list.

Properties

```
public Properties(Properties defaults)
```

Creates an empty property list with specified defaults.

Parameters:

defaults - the defaults

Interface Index**load**

```
public synchronized void load(InputStream in) throws IOException
```

Loads properties from an InputStream.
Parameters:
in - the input stream
Throws: IOException
Error when reading from input stream.

Interface Index**save**

```
public synchronized void save(OutputStream out,
                              String header)
```

Save properties to an OutputStream. Use the header as a comment at the top of the file.

Interface Index**getProperty**

```
public String getProperty(String key)
```

Gets a property with the specified key. If the key is not found in this property list, tries the defaults. This method returns null if the property is not found.
Parameters:
key - the hashtable key

Interface Index**getProperty**

```
public String getProperty(String key,
                          String defaultValue)
```

Gets a property with the specified key and default. If the key is not found in this property list, tries the defaults. This method returns defaultValue if the property is not found.

propertyNames

```
public Enumeration propertyNames()
```

Enumerates all the keys.

list

```
public void list(PrintStream out)
```

List properties, for debugging

Class java.util.Random

Class java.util.Random

```
java.lang.Object
   |
   +----java.util.Random
```

---

public class **Random**
extends Object

A Random class generates a stream of pseudo-random numbers.

To create a new random number generator, use one of the following methods:

```
new Random()
new Random(long seed)
```

The form `new Random()` initializes the generator to a value based on the current time. The form `new Random(long seed)` seeds the random number generator with a specific initial value; use this if an application requires a repeatable stream of pseudo-random numbers.

The random number generator uses a 48-bit seed, which is modified using a linear congruential formula. See Donald Knuth, *The Art of Computer Programming, Volume 2*, Section 3.2.1. The generator's seed can be reset with the following method:

```
setSeed(long seed)
```

To create a pseudo-random number, use one of the following functions:

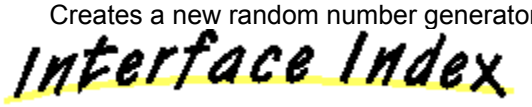
```
nextInt()
nextLong()
nextFloat()
nextDouble()
nextGaussian()
```

See Also:
random

Interface Index

Random()

Creates a new random number generator.

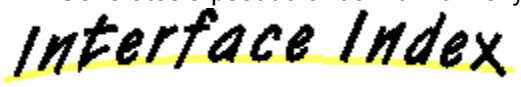


Random(long)

Creates a new random number generator using a single `long` seed.

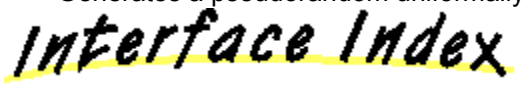
nextDouble()

Generates a pseudorandom uniformally distributed `double` value between 0.0 and 1.0.



nextFloat()

Generates a pseudorandom uniformally distributed `float` value between 0.0 and 1.0.



nextGaussian()

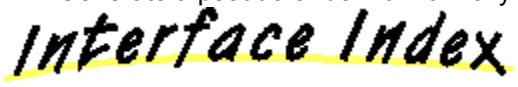
Generates a pseudorandom Gaussian distributed `double` value with mean 0.0 and standard deviation 1.0.

nextInt()

Generates a pseudorandom uniformally distributed `int` value.

nextLong()

Generate a pseudorandom uniformally distributed `long` value.



setSeed(long)

Sets the seed of the random number generator using a single `long` seed.

Random

```
public Random()
```

Creates a new random number generator. Its seed will be initialized to a value based on the current time.

Random

```
public Random(long seed)
```

Creates a new random number generator using a single `long` seed.

Parameters:

seed - the initial seed

See Also:

setSeed

Interface Index

Interface Index **setSeed**

```
public synchronized void setSeed(long seed)
```

Sets the seed of the random number generator using a single `long` seed.

Parameters:

seed - the initial seed

Interface Index **nextInt**

```
public int nextInt()
```

Generates a pseudorandom uniformally distributed `int` value.

Returns:

an integer value.

Interface Index **nextLong**

```
public long nextLong()
```

Generate a pseudorandom uniformally distributed `long` value.

Returns:

A long integer value

nextFloat

```
public float nextFloat()
```

Generates a pseudorandom uniformally distributed `float` value between 0.0 and 1.0.

Returns:

a `float` between 0.0 and 1.0 .

nextDouble

```
public double nextDouble()
```

Generates a pseudorandom uniformally distributed `double` value between 0.0 and 1.0.

Returns:

a `float` between 0.0 and 1.0 .

nextGaussian

```
public synchronized double nextGaussian()
```

Generates a pseudorandom Gaussian distributed `double` value with mean 0.0 and standard deviation 1.0.

Returns:

a Gaussian distributed `double`.

Class java.util.Stack

Class java.util.Stack

```
java.lang.Object
    |
    +----java.util.Vector
             |
             +----java.util.Stack
```

---

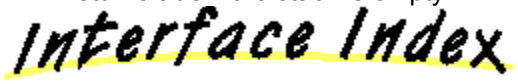
public class **Stack**
extends Vector

A Last-In-First-Out(LIFO) stack of objects.

Stack()

empty()

Returns true if the stack is empty.



peek()

Peeks at the top of the stack.

pop()

Pops an item off the stack.

push(Object)

Pushes an item onto the stack.

search(Object)

Sees if an object is on the stack.

Stack

```
public Stack()
```


push

```
public Object push(Object item)
```

Pushes an item onto the stack.

Parameters:

item - the item to be pushed on.

pop

```
public Object pop()
```

Pops an item off the stack.

Throws: EmptyStackException

If the stack is empty.

peek

```
public Object peek()
```

Peeks at the top of the stack.

Throws: EmptyStackException

If the stack is empty.

empty

```
public boolean empty()
```

Returns true if the stack is empty.

search

```
public int search(Object o)
```

Sees if an object is on the stack.

Parameters:

o - the desired object

Returns:

the distance from the top, or -1 if it is not found.

Class java.util.StringTokenizer

Class java.util.StringTokenizer

```
java.lang.Object
   |
   +----java.util.StringTokenizer
```

---

public class **StringTokenizer**
extends Object
implements Enumeration

StringTokenizer is a class that controls simple linear tokenization of a String. The set of delimiters, which defaults to common whitespace characters, may be specified at creation time or on a per-token basis.

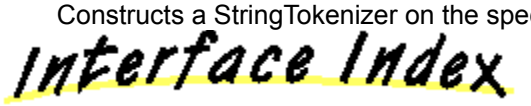
Example usage:

```
String s = "this is a test";
StringTokenizer st = new StringTokenizer(s);
while (st.hasMoreTokens()) {
        println(st.nextToken());
}
```

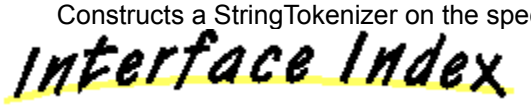
Prints the following on the console:

```
this
is
a
test
```

StringTokenizer(String, String, boolean)
Constructs a StringTokenizer on the specified String, using the specified delimiter set.



StringTokenizer(String, String)
Constructs a StringTokenizer on the specified String, using the specified delimiter set.



StringTokenizer(String)

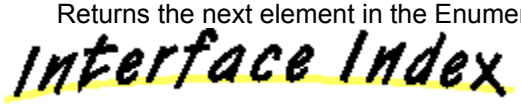
Constructs a StringTokenizer on the specified String, using the default delimiter set (which is " \t\n\r").

countTokens()
Returns the next number of tokens in the String using the current deliminter set.

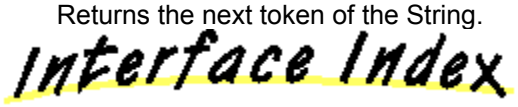
hasMoreElements()
Returns true if the Enumeration has more elements.

hasMoreTokens()
Returns true if more tokens exist.

nextElement()
Returns the next element in the Enumeration.



nextToken()
Returns the next token of the String.



nextToken(String)
Returns the next token, after switching to the new delimiter set.

StringTokenizer

```
public StringTokenizer(String str,
                       String delim,
                       boolean returnTokens)
```

Constructs a StringTokenizer on the specified String, using the specified delimiter set.
Parameters:
str - the input String
delim - the delimiter String
returnTokens - returns delimiters as tokens or skip them

StringTokenizer

```
public StringTokenizer(String str,
```

String delim)

Constructs a StringTokenizer on the specified String, using the specified delimiter set.

Parameters:

str - the input String

delim - the delimiter String

StringTokenizer

```
public StringTokenizer(String str)
```

Constructs a StringTokenizer on the specified String, using the default delimiter set (which is " \t\n\r").

Parameters:

str - the String

hasMoreTokens

```
public boolean hasMoreTokens()
```

Returns true if more tokens exist.

nextToken

```
public String nextToken()
```

Returns the next token of the String.

Throws: NoSuchElementException

If there are no more tokens in the String.

nextToken

```
public String nextToken(String delim)
```

Returns the next token, after switching to the new delimiter set. The new delimiter set remains the

default after this call.
Parameters:
delim - the new delimiters

hasMoreElements

```
public boolean hasMoreElements()
```

Returns true if the Enumeration has more elements.

nextElement

```
public Object nextElement()
```

Returns the next element in the Enumeration.
Throws: NoSuchElementException
If there are no more elements in the enumeration.

countTokens

```
public int countTokens()
```

Returns the next number of tokens in the String using the current deliminter set. This is the number of times nextToken() can return before it will generate an exception. Use of this routine to count the number of tokens is faster than repeatedly calling nextToken() because the substrings are not constructed and returned for each token.

Class java.util.Vector

Class java.util.Vector

```
java.lang.Object
    |
    +----java.util.Vector
```

---

public class **Vector**
extends Object
implements Cloneable

Vector class (a growable array).

Each vector tries to optimize storage management by maintaining a capacity and a capacityIncrement. The capacity is always at least as large as the vector size; it is usually larger because as elements are added to the vector, the vector's storage increases in chunks the size of capacityIncrement. Setting the capacity to what you want before inserting a large number of objects will reduce the amount of incremental reallocation. You can safely ignore the capacity and the vector will still work correctly.

capacityIncrement
The size of the increment.

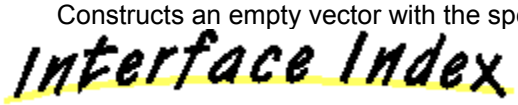
elementCount
The number of elements in the buffer.

elementData
The buffer where elements are stored.

Vector(int, int)
Constructs an empty vector with the specified storage capacity and the specified capacityIncrement.

Vector(int)

Constructs an empty vector with the specified storage capacity.

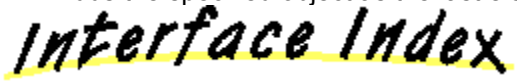


Vector()

Constructs an empty vector.

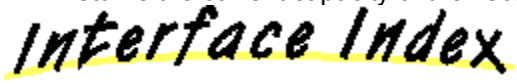
addElement(Object)

Adds the specified object as the last element of the vector.



capacity()

Returns the current capacity of the vector.

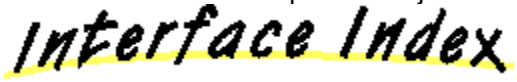


clone()

Clones this vector.

contains(Object)

Returns true if the specified object is a value of the collection.

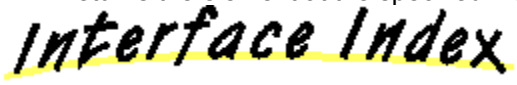


copyInto(Object[])

Copies the elements of this vector into the specified array.

elementAt(int)

Returns the element at the specified index.



elements()

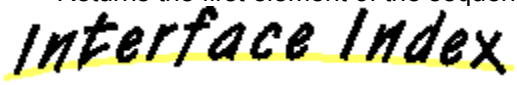
Returns an enumeration of the elements.

ensureCapacity(int)

Ensures that the vector has at least the specified capacity.

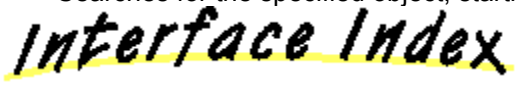
firstElement()

Returns the first element of the sequence.



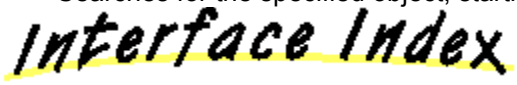
indexOf(Object)

Searches for the specified object, starting from the first position and returns an index to it.



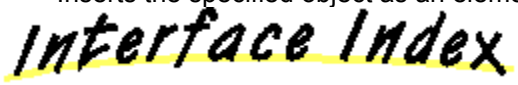
indexOf(Object, int)

Searches for the specified object, starting at the specified position and returns an index to it.



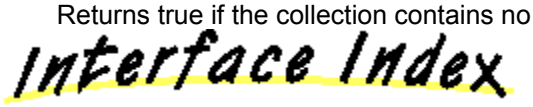
insertElementAt(Object, int)

Inserts the specified object as an element at the specified index.



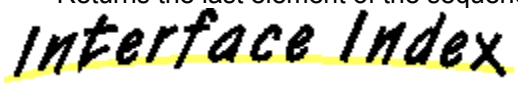
isEmpty()

Returns true if the collection contains no values.



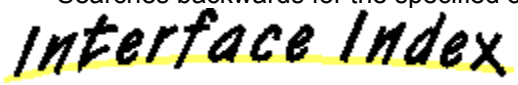
lastElement()

Returns the last element of the sequence.



lastIndexOf(Object)

Searches backwards for the specified object, starting from the last position and returns an index to it.



lastIndexOf(Object, int)

Searches backwards for the specified object, starting from the specified position and returns an index to it.

removeAllElements()

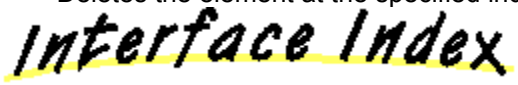
Removes all elements of the vector.

removeElement(Object)

Removes the element from the vector.

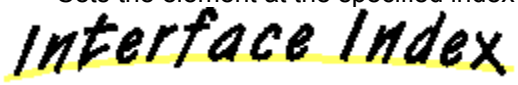
removeElementAt(int)

Deletes the element at the specified index.



setElementAt(Object, int)

Sets the element at the specified index to be the specified object.



setSize(int)

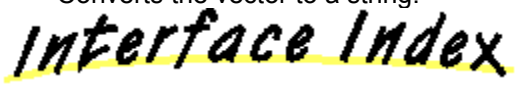
Sets the size of the vector.

size()

Returns the number of elements in the vector.

toString()

Converts the vector to a string.



trimToSize()

Trims the vector's capacity down to size.

elementData

```
protected Object elementData[]
```

The buffer where elements are stored.

elementCount

```
protected int elementCount
```

The number of elements in the buffer.

capacityIncrement

```
protected int capacityIncrement
```

The size of the increment. If it is 0 the size of the the buffer is doubled everytime it needs to grow.

Vector

```
public Vector(int initialCapacity,
              int capacityIncrement)
```

Constructs an empty vector with the specified storage capacity and the specified capacityIncrement.

Parameters:

initialCapacity - the initial storage capacity of the vector
capacityIncrement - how much to increase the element's size by.

Vector

```
public Vector(int initialCapacity)
```

Constructs an empty vector with the specified storage capacity.

Parameters:

initialCapacity - the initial storage capacity of the vector

Vector

```
public Vector()
```

Constructs an empty vector.

copyInto

```
public final synchronized void copyInto(Object anArray[])
```

Copies the elements of this vector into the specified array.

Parameters:

anArray - the array where elements get copied into

trimToSize

```
public final synchronized void trimToSize()
```

Trims the vector's capacity down to size. Use this operation to minimize the storage of a vector. Subsequent insertions will cause reallocation.

ensureCapacity

```
public final synchronized void ensureCapacity(int minCapacity)
```

Ensures that the vector has at least the specified capacity.

Parameters:

minCapacity - the desired minimum capacity

setSize

```
public final synchronized void setSize(int newSize)
```

Sets the size of the vector. If the size shrinks, the extra elements (at the end of the vector) are lost; if the size increases, the new elements are set to null.

Parameters:

newSize - the new size of the vector

capacity

```
public final int capacity()
```

Returns the current capacity of the vector.

size

```
public final int size()
```

Returns the number of elements in the vector. Note that this is not the same as the vector's capacity.

isEmpty

```
public final boolean isEmpty()
```

Returns true if the collection contains no values.

elements

```
public final synchronized Enumeration elements()
```

Returns an enumeration of the elements. Use the Enumeration methods on the returned object to fetch the elements sequentially.

contains

```
public final boolean contains(Object elem)
```

Returns true if the specified object is a value of the collection.

Parameters:

elem - the desired element

indexOf

```
public final int indexOf(Object elem)
```

Searches for the specified object, starting from the first position and returns an index to it.

Parameters:

elem - the desired element

Returns:

the index of the element, or -1 if it was not found.

Interface Index indexOf

```
public final synchronized int indexOf(Object elem,
                                       int index)
```

Searches for the specified object, starting at the specified position and returns an index to it.

Parameters:

elem - the desired element

index - the index where to start searching

Returns:

the index of the element, or -1 if it was not found.

Interface Index lastIndexOf

```
public final int lastIndexOf(Object elem)
```

Searches backwards for the specified object, starting from the last position and returns an index to it.

Parameters:

elem - the desired element

Returns:

the index of the element, or -1 if it was not found.

Interface Index lastIndexOf

```
public final synchronized int lastIndexOf(Object elem,
                                           int index)
```

Searches backwards for the specified object, starting from the specified position and returns an index to it.

Parameters:

elem - the desired element

index - the index where to start searching

Returns:

the index of the element, or -1 if it was not found.

elementAt

```
public final synchronized Object elementAt(int index)
```

Returns the element at the specified index.

Parameters:

index - the index of the desired element

Throws: ArrayIndexOutOfBoundsException

If an invalid index was given.

firstElement

```
public final synchronized Object firstElement()
```

Returns the first element of the sequence.

Throws: NoSuchElementException

If the sequence is empty.

lastElement

```
public final synchronized Object lastElement()
```

Returns the last element of the sequence.

Throws: NoSuchElementException

If the sequence is empty.

setElementAt

```
public final synchronized void setElementAt(Object obj,
                                             int index)
```

Sets the element at the specified index to be the specified object. The previous element at that position is discarded.

Parameters:

obj - what the element is to be set to

index - the specified index

Throws: ArrayIndexOutOfBoundsException
If the index was invalid.

removeElementAt

```
public final synchronized void removeElementAt(int index)
```

Deletes the element at the specified index. Elements with an index greater than the current index are moved down.
Parameters:
index - the element to remove
Throws: ArrayIndexOutOfBoundsException
If the index was invalid.

insertElementAt

```
public final synchronized void insertElementAt(Object obj,
                                                int index)
```

Inserts the specified object as an element at the specified index. Elements with an index greater or equal to the current index are shifted up.
Parameters:
obj - the element to insert
index - where to insert the new element
Throws: ArrayIndexOutOfBoundsException
If the index was invalid.

addElement

```
public final synchronized void addElement(Object obj)
```

Adds the specified object as the last element of the vector.
Parameters:
obj - the element to be added

removeElement

```
public final synchronized boolean removeElement(Object obj)
```

Removes the element from the vector. If the object occurs more than once, only the first is removed. If the object is not an element, returns false.

Parameters:

obj - the element to be removed

Returns:

true if the element was actually removed; false otherwise.

removeAllElements

```
public final synchronized void removeAllElements()
```

Removes all elements of the vector. The vector becomes empty.

clone

```
public synchronized Object clone()
```

Clones this vector. The elements are **not** cloned.

Overrides:

clone in class Object

toString

```
public final synchronized String toString()
```

Converts the vector to a string. Useful for debugging.

Overrides:

toString in class Object

Interface sun.tools.debug.DebuggerCallback

Interface sun.tools.debug.DebuggerCallback

public interface **DebuggerCallback**
extends Object

The DebuggerCallback interface is used to communicate asynchronous information from the debugger to its client. This may be the actual client object, or a delegate of its choosing.

Interface Index

Interface Index

breakpointEvent(RemoteThread)
A breakpoint has been hit in the specified thread.

Interface Index

exceptionEvent(RemoteThread, String)
An exception has occurred.

Interface Index

printToConsole(String)
Print text to the debugger's console window.

Interface Index

quitEvent()
The client interpreter has exited, either by returning from its main thread, or by calling System.exit().

Interface Index

threadDeathEvent(RemoteThread)
A thread has died.

Interface Index

Interface Index

printToConsole

```
public abstract void printToConsole(String text) throws Exception
```

Print text to the debugger's console window.

breakpointEvent

```
public abstract void breakpointEvent(RemoteThread t) throws Exception
```

A breakpoint has been hit in the specified thread.

exceptionEvent

```
public abstract void exceptionEvent(RemoteThread t,
                                    String errorText) throws Exception
```

An exception has occurred.

threadDeathEvent

```
public abstract void threadDeathEvent(RemoteThread t) throws Exception
```

A thread has died.

quitEvent

```
public abstract void quitEvent() throws Exception
```

The client interpreter has exited, either by returning from its main thread, or by calling System.exit().

Class sun.tools.debug.RemoteArray

Class sun.tools.debug.RemoteArray

```
java.lang.Object
    |
    +----sun.tools.debug.RemoteValue
             |
             +----sun.tools.debug.RemoteObject
                      |
                      +----sun.tools.debug.RemoteArray
```

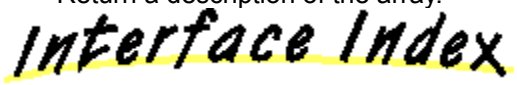
---

public class **RemoteArray**
extends RemoteObject

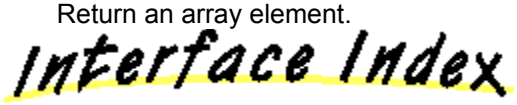
The RemoteArray class allows remote debugging of arrays.
See Also:
RemoteValue, RemoteDebugger

arrayTypeName(int)
Return the element type as a string.

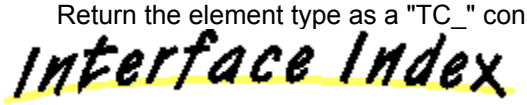
description()
Return a description of the array.



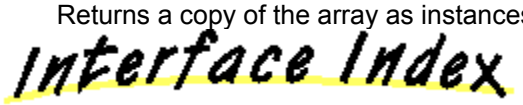
getElement(int)
Return an array element.



getElementType()
Return the element type as a "TC_" constant, such as "TC_CHAR".

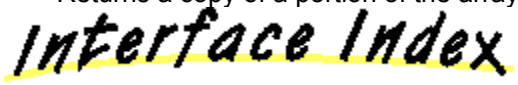


getElements()
Returns a copy of the array as instances of RemoteValue.



getElements(int, int)

Returns a copy of a portion of the array as instances of RemoteValue.

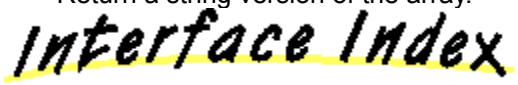


getSize()

Return the number of elements in the array.

toString()

Return a string version of the array.



typeName()

Return this RemoteValue's type ("array").

getSize

```
public final int getSize()
```

Return the number of elements in the array.

typeName

```
public String typeName()
```

Return this RemoteValue's type ("array").

Overrides:

typeName in class RemoteObject

arrayTypeName

```
public String arrayTypeName(int type)
```

Return the element type as a string.

getElementType

```
public final int getElementType() throws Exception
```

Return the element type as a "TC_" constant, such as "TC_CHAR".

 getElement

```
public final RemoteValue getElement(int index) throws Exception
```

Return an array element.
Parameters:
index - the index of the element
Returns:
the element as a RemoteValue
Throws: ArrayIndexOutOfBoundsException
when the index is greater than the size of the array

 getElements

```
public final RemoteValue[] getElements() throws Exception
```

Returns a copy of the array as instances of RemoteValue.

 getElements

```
public final RemoteValue[] getElements(int beginIndex,
                                       int endIndex) throws Exception
```

Returns a copy of a portion of the array as instances of RemoteValue.
Parameters:
beginIndex - the beginning array index
endIndex - the final array index
Throws: ArrayIndexOutOfBoundsException
when the index is greater than the size of the array

 description

```
public String description()
```

Return a description of the array.
Overrides:
description in class RemoteObject

toString

```
public String toString()
```

Return a string version of the array.

Overrides:

toString in class RemoteObject

Class sun.tools.debug.RemoteBoolean

Class sun.tools.debug.RemoteBoolean

```
java.lang.Object
    |
    +----sun.tools.debug.RemoteValue
             |
             +----sun.tools.debug.RemoteBoolean
```

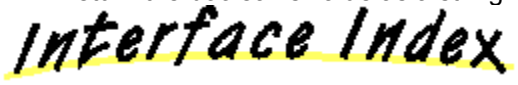
---

public class **RemoteBoolean**
extends RemoteValue

The RemoteBoolean class extends RemoteValue for booleans.
See Also:
RemoteValue, RemoteDebugger

get()
Return the boolean's value.

toString()
Return the boolean's value as a string.



typeName()
Print this RemoteValue's type ("boolean").

get

```
public boolean get()
```

Return the boolean's value.

typeName

```
public String typeName()
```

Print this RemoteValue's type ("boolean").

Overrides:

typeName in class RemoteValue

toString

```
public String toString()
```

Return the boolean's value as a string.

Overrides:

toString in class Object

Class sun.tools.debug.RemoteByte

Class sun.tools.debug.RemoteByte

```
java.lang.Object
    |
    +----sun.tools.debug.RemoteValue
             |
             +----sun.tools.debug.RemoteByte
```

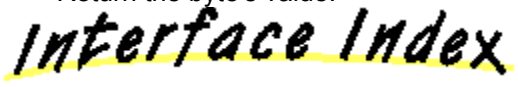
---

public class **RemoteByte**
extends RemoteValue

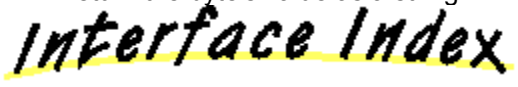
The RemoteByte class extends RemoteValue for bytes.

See Also:
RemoteValue, RemoteDebugger

get()
Return the byte's value.



toString()
Return the byte's value as a string.



typeName()
Print this RemoteValue's type ("byte").

get

```
public byte get()
```

Return the byte's value.

typeName

```
public String typeName()
```

Print this RemoteValue's type ("byte").

Overrides:

typeName in class RemoteValue

toString

```
public String toString()
```

Return the byte's value as a string.

Overrides:

toString in class Object

Class sun.tools.debug.RemoteChar

Class sun.tools.debug.RemoteChar

```
java.lang.Object
    |
    +----sun.tools.debug.RemoteValue
             |
             +----sun.tools.debug.RemoteChar
```

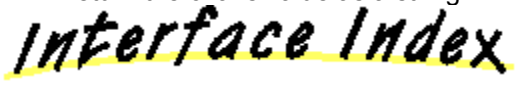
---

public class **RemoteChar**
extends RemoteValue

The RemoteChar class extends RemoteValue for chars.
See Also:
RemoteValue, RemoteDebugger

get()
Return the char's value.

toString()
Return the char's value as a string.



typeName()
Print this RemoteValue's type ("char").

get

```
public char get()
```

Return the char's value.

typeName

```
public String typeName()
```

Print this RemoteValue's type ("char").

Overrides:

typeName in class RemoteValue

toString

```
public String toString()
```

Return the char's value as a string.

Overrides:

toString in class Object

Class sun.tools.debug.RemoteClass

Class sun.tools.debug.RemoteClass

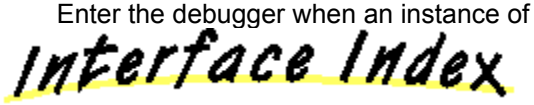
```
java.lang.Object
    |
    +----sun.tools.debug.RemoteValue
             |
             +----sun.tools.debug.RemoteObject
                      |
                      +----sun.tools.debug.RemoteClass
```

---

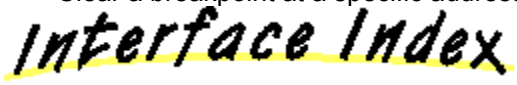
public class **RemoteClass**
extends RemoteObject

The RemoteClass class allows access to a class in a remote Java interpreter.
See Also:
RemoteDebugger

catchExceptions()
Enter the debugger when an instance of this class is thrown.



clearBreakpoint(int)
Clear a breakpoint at a specific address in a class.



clearBreakpointLine(int)
Clear a breakpoint at a specified line.

clearBreakpointMethod(RemoteField)
Clear a breakpoint at the start of a specified method.

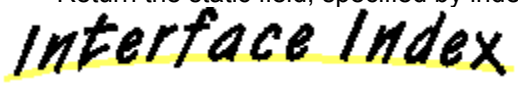
description()
Return a (somewhat verbose) description.

getClassLoader()

Return the classloader for this class.

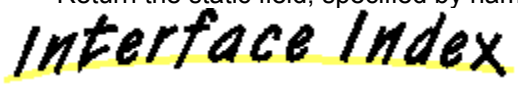
getField(int)

Return the static field, specified by index.



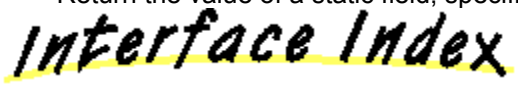
getField(String)

Return the static field, specified by name.



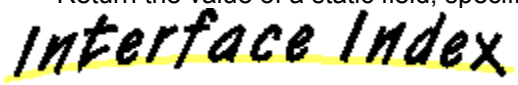
getFieldValue(int)

Return the value of a static field, specified by its index



getFieldValue(String)

Return the value of a static field, specified by name.

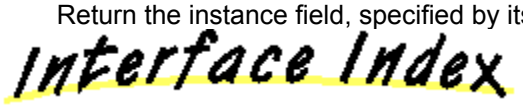


getFields()

Return all the static fields for this class.

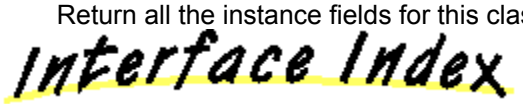
getInstanceField(int)

Return the instance field, specified by its index.



getInstanceFields()

Return all the instance fields for this class.

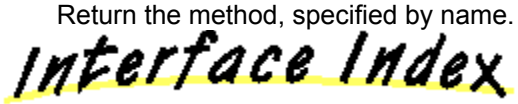


getInterfaces()

Return the interfaces for this class.

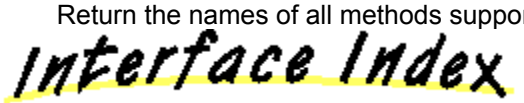
getMethod(String)

Return the method, specified by name.



getMethodNames()

Return the names of all methods supported by this class.

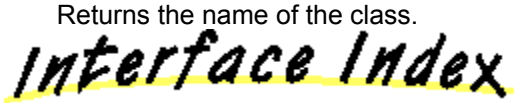


getMethods()

Return the class's methods.

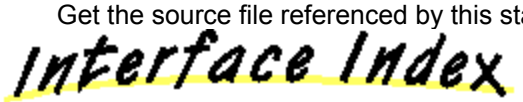
getName()

Returns the name of the class.



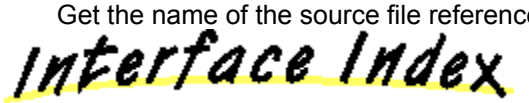
getSourceFile()

Get the source file referenced by this stackframe.



getSourceFileName()

Get the name of the source file referenced by this stackframe.



getStaticFields()

Return all the static fields for this class.

Interface Index **getSuperclass**()
Return the superclass for this class.
Interface Index **ignoreExceptions**()
Don't enter the debugger when an instance of this class is thrown.
Interface Index **isInterface**()
Is this RemoteClass an interface?
Interface Index **setBreakpointLine**(int)
Set a breakpoint at a specified source line number in a class.
Interface Index **setBreakpointMethod**(RemoteField)
Set a breakpoint at the first line of a class method.
Interface Index **toString**()
Return a (somewhat verbose) description.
Interface Index **typeName**()
Returns the name of the class as its type.

Interface Index

Interface Index **getName**

```
public String getName() throws Exception
```

Returns the name of the class.

Interface Index **typeName**

```
public String typeName() throws Exception
```

Returns the name of the class as its type.
Overrides:
typeName in class RemoteObject

Interface Index **isInterface**

```
public boolean isInterface() throws Exception
```

Is this RemoteClass an interface?

Interface Index **getSuperclass**

```
public RemoteClass getSuperclass() throws Exception
```

Return the superclass for this class.

Interface Index **getClassLoader**

```
public RemoteObject getClassLoader() throws Exception
```

Return the classloader for this class.

Interface Index **getInterfaces**

```
public RemoteClass[] getInterfaces() throws Exception
```

Return the interfaces for this class.

Interface Index **getSourceFileName**

```
public String getSourceFileName()
```

Get the name of the source file referenced by this stackframe.

Interface Index **getSourceFile**

```
public InputStream getSourceFile() throws Exception
```

Get the source file referenced by this stackframe.

Interface Index **getFields**

```
public RemoteField[] getFields() throws Exception
```

Return all the static fields for this class.

Overrides:
getFields in class RemoteObject

getStaticFields

```
public RemoteField[] getStaticFields() throws Exception
```

Return all the static fields for this class.

getInstanceFields

```
public RemoteField[] getInstanceFields() throws Exception
```

Return all the instance fields for this class. Note: because this is a RemoteClass method, only the name and type methods will be valid, not the data.

getField

```
public RemoteField getField(int n) throws Exception
```

Return the static field, specified by index.
Throws: ArrayIndexOutOfBoundsException
when the index is greater than the number of instance variables
Overrides:
getField in class RemoteObject

getField

```
public RemoteField getField(String name) throws Exception
```

Return the static field, specified by name.
Overrides:
getField in class RemoteObject

getInstanceField

```
public RemoteField getInstanceField(int n) throws Exception
```

Return the instance field, specified by its index. Note: because this is a RemoteClass method, only the name and type information is valid, not the data.

Throws: ArrayIndexOutOfBoundsException
when the index is greater than the number of instance variables

getFieldValue

```
public RemoteValue getFieldValue(int n) throws Exception
```

Return the value of a static field, specified by its index

Overrides:
getFieldValue in class RemoteObject

getFieldValue

```
public RemoteValue getFieldValue(String name) throws Exception
```

Return the value of a static field, specified by name.

Overrides:
getFieldValue in class RemoteObject

getMethod

```
public RemoteField getMethod(String name) throws Exception
```

Return the method, specified by name.

getMethods

```
public RemoteField[] getMethods() throws Exception
```

Return the class's methods.

getMethodNames

```
public String[] getMethodNames() throws Exception
```

Return the names of all methods supported by this class.

setBreakpointLine

```
public String setBreakpointLine(int lineno) throws Exception
```

Set a breakpoint at a specified source line number in a class.

Parameters:

lineno - the line number where the breakpoint is set

Returns:

an empty string if successful, otherwise a description of the error.

setBreakpointMethod

```
public String setBreakpointMethod(RemoteField method) throws Exception
```

Set a breakpoint at the first line of a class method.

Parameters:

method - the method where the breakpoint is set

Returns:

an empty string if successful, otherwise a description of the error.

clearBreakpoint

```
public String clearBreakpoint(int pc) throws Exception
```

Clear a breakpoint at a specific address in a class.

Parameters:

pc - the address of the breakpoint to be cleared

Returns:

an empty string if successful, otherwise a description of the error.

clearBreakpointLine

```
public String clearBreakpointLine(int lineno) throws Exception
```

Clear a breakpoint at a specified line.

Parameters:

lineno - the line number of the breakpoint to be cleared

Returns:

an empty string if successful, otherwise a description of the error.

clearBreakpointMethod

```
public String clearBreakpointMethod(RemoteField method) throws Exception
```

Clear a breakpoint at the start of a specified method.

Parameters:

method - the method of the breakpoint to be cleared

Returns:

an empty string if successful, otherwise a description of the error.

catchExceptions

```
public void catchExceptions() throws Exception
```

Enter the debugger when an instance of this class is thrown.

Throws: ClassCastException

when this class isn't an exception class

ignoreExceptions

```
public void ignoreExceptions() throws Exception
```

Don't enter the debugger when an instance of this class is thrown.

Throws: ClassCastException

when this class isn't an exception class

description

```
public String description()
```

Return a (somewhat verbose) description.

Overrides:

description in class RemoteObject

toString

```
public String toString()
```

Return a (somewhat verbose) description.

Overrides:

toString in class RemoteObject

Class sun.tools.debug.RemoteDebugger

Class sun.tools.debug.RemoteDebugger

```
java.lang.Object
    |
    +----sun.tools.debug.RemoteDebugger
```

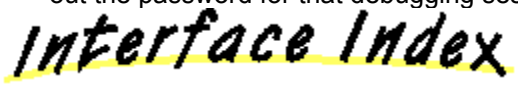
---

public class **RemoteDebugger**
extends Object

The RemoteDebugger class defines a client interface to the Java debugging classes. It is used to instantiate a connection with the Java interpreter being debugged.

RemoteDebugger(String, String, DebuggerCallback, boolean)
Create a remote debugger, connecting it with a running Java interpreter.

To connect to a running interpreter, it must be started with the "-debug" option, whereupon it will print out the password for that debugging session.

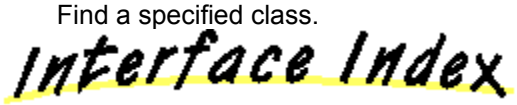


RemoteDebugger(String, DebuggerCallback, boolean)
Create a remote debugger, and connect it to a new client interpreter.

close()
Close the connection to the remote debugging agent.

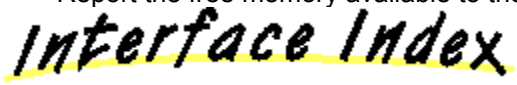
findClass(String)

Find a specified class.



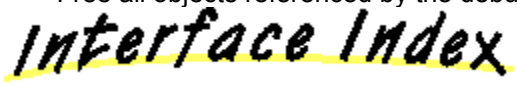
freeMemory()

Report the free memory available to the Java interpreter being debugged.



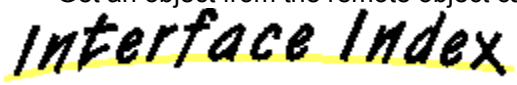
gc(RemoteObject[])

Free all objects referenced by the debugger.



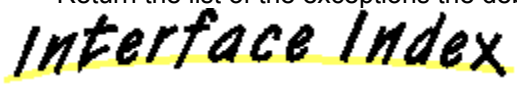
get(Integer)

Get an object from the remote object cache.



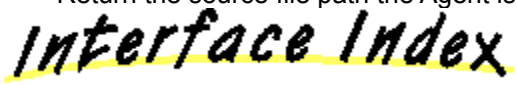
getExceptionCatchList()

Return the list of the exceptions the debugger will stop on.



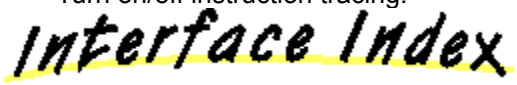
getSourcePath()

Return the source file path the Agent is currently using.



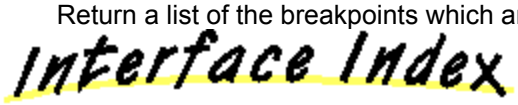
itrace(boolean)

Turn on/off instruction tracing.



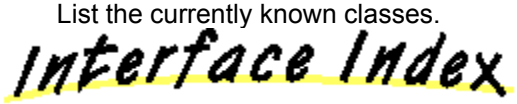
listBreakpoints()

Return a list of the breakpoints which are currently set.



listClasses()

List the currently known classes.

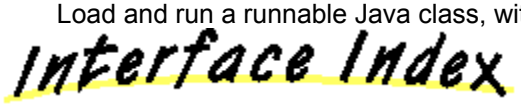


listThreadGroups(RemoteThreadGroup)

List threadgroups

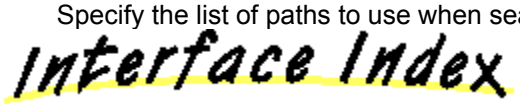
run(int, String[])

Load and run a runnable Java class, with any optional parameters.



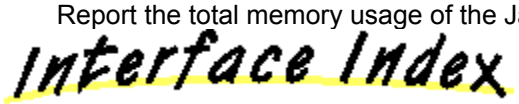
setSourcePath(String)

Specify the list of paths to use when searching for a source file.



totalMemory()

Report the total memory usage of the Java interpreter being debugged.



trace(boolean)

Turn on/off method call tracing.

RemoteDebugger

```
public RemoteDebugger(String host,
                      String password,
                      DebuggerCallback client,
                      boolean verbose) throws Exception
```

Create a remote debugger, connecting it with a running Java interpreter.

To connect to a running interpreter, it must be started with the "-debug" option, whereupon it will print out the password for that debugging session.

Parameters:

host - the name of the system where a debuggable Java instance is running (default is localhost).

password - the password reported by the debuggable Java instance. This should be null when starting a client interpreter.

client - the object to which notification messages are sent (it must support the DebuggerCallback interface)

verbose - turn on internal debugger message text

RemoteDebugger

```
public RemoteDebugger(String javaArgs,
                      DebuggerCallback client,
                      boolean verbose) throws Exception
```

Create a remote debugger, and connect it to a new client interpreter.

Parameters:

javaArgs - optional java command-line parameters, such as -classpath

client - the object to which notification messages are sent (it must support the DebuggerCallback interface)

verbose - turn on internal debugger message text

close

```
public void close()
```

Close the connection to the remote debugging agent.

get

```
public RemoteObject get(Integer id)
```

Get an object from the remote object cache.

Parameters:

id - the remote object's id

Returns:

the specified RemoteObject, or null if not cached.

listClasses

```
public RemoteClass[] listClasses() throws Exception
```

List the currently known classes.

findClass

```
public RemoteClass findClass(String name) throws Exception
```

Find a specified class. If the class isn't already known by the remote debugger, the lookup request will be passed to the Java interpreter being debugged. NOTE: Substrings, such as "String" for "java.lang.String" will return with the first match, and will not be successfully found if the request is passed to the remote interpreter.

Parameters:

name - the name (or a substring of the name)of the class

Returns:

the specified (Remote)Class, or null if not found.

listThreadGroups

```
public RemoteThreadGroup[] listThreadGroups(RemoteThreadGroup tg) throws
Exception
```

List threadgroups

Parameters:

tg - the threadgroup which hold the groups to be listed, or null for all threadgroups

Interface Index

gc

```
public void gc(RemoteObject save_list[]) throws Exception
```

Free all objects referenced by the debugger. The remote debugger maintains a copy of each object it has examined, so that references won't become invalidated by the garbage collector of the Java interpreter being debugged. The gc() method frees all all of these references, except those specified to save.

Parameters:

save_list - the list of objects to save.

Interface Index

trace

```
public void trace(boolean traceOn) throws Exception
```

Turn on/off method call tracing. When turned on, each method call is reported to the stdout of the Java interpreter being debugged. This output is not captured in any way by the remote debugger.

Parameters:

traceOn - turn tracing on or off

Interface Index

itrace

```
public void itrace(boolean traceOn) throws Exception
```

Turn on/off instruction tracing. When turned on, each Java instruction is reported to the stdout of the Java interpreter being debugged. This output is not captured in any way by the remote debugger.

Parameters:

traceOn - turn tracing on or off

Interface Index

totalMemory

```
public int totalMemory() throws Exception
```

Report the total memory usage of the Java interpreter being debugged.

Interface Index

freeMemory

```
public int freeMemory() throws Exception
```

Report the free memory available to the Java interpreter being debugged.

Interface Index

run

```
public RemoteThreadGroup run(int argc,
                             String argv[]) throws Exception
```

Load and run a runnable Java class, with any optional parameters. The class is started inside a new threadgroup in the Java interpreter being debugged. NOTE: Although it is possible to run multiple runnable classes from the same Java interpreter, there is no guarantee that all applets will work cleanly with each other. For example, two applets may want exclusive access to the same shared resource,such as a specific port.

Parameters:

argc - the number of parameters

argv - the array of parameters: the class to be run is first, followed by any optional parameters used by that class.

Returns:

the new ThreadGroup the class is running in, or null on error

Interface Index

listBreakpoints

```
public String[] listBreakpoints() throws Exception
```

Return a list of the breakpoints which are currently set.

Returns:

an array of Strings of the form "class_name:line_number".

Interface Index

getExceptionCatchList

```
public String[] getExceptionCatchList() throws Exception
```

Return the list of the exceptions the debugger will stop on.

Returns:

an array of exception class names, which may be zero-length.

Interface Index

getSourcePath

```
public String getSourcePath() throws Exception
```

Return the source file path the Agent is currently using.

Returns:

a string consisting of a list of colon-delineated paths.


```
public void setSourcePath(String pathList) throws Exception
```

Specify the list of paths to use when searching for a source file.

Parameters:

pathList - a string consisting of a list of colon-delineated paths.

Class sun.tools.debug.RemoteDouble

Class sun.tools.debug.RemoteDouble

```
java.lang.Object
    |
    +----sun.tools.debug.RemoteValue
             |
             +----sun.tools.debug.RemoteDouble
```

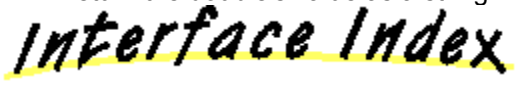
---

public class **RemoteDouble**
extends RemoteValue

The RemoteDouble class extends RemoteValue for doubles.
See Also:
RemoteValue, RemoteDebugger

get()
Return the double's value.

toString()
Return the double's value as a string.



typeName()
Print this RemoteValue's type ("double").

get

```
public double get()
```

Return the double's value.

typeName

```
public String typeName()
```

Print this RemoteValue's type ("double").

Overrides:

typeName in class RemoteValue

toString

```
public String toString()
```

Return the double's value as a string.

Overrides:

toString in class Object

Class sun.tools.debug.RemoteField

Class sun.tools.debug.RemoteField

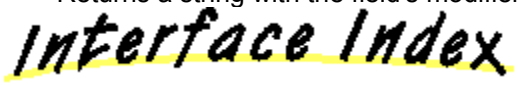
```
java.lang.Object
    |
    +----sun.tools.debug.Field
             |
             +----sun.tools.debug.RemoteField
```

---

public class **RemoteField**
extends Field
implements AgentConstants

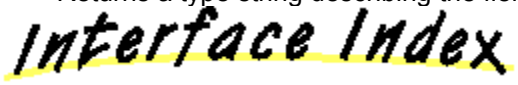
A RemoteField allows access to a variable or method of an object or class in a remote Java interpreter.
See Also:
RemoteStackVariable

getModifiers()
Returns a string with the field's modifiers, such as "public", "static", "final", etc.

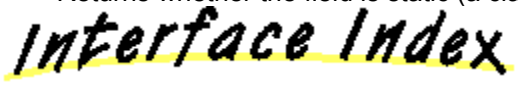


getName()
Returns the name of the field.

getType()
Returns a type string describing the field.



isStatic()
Returns whether the field is static (a class variable or method).



toString()
Returns a String that represents the value of this Object.

Interface Index **getName**

```
public String getName()
```

Returns the name of the field.

Interface Index **getType**

```
public String getType()
```

Returns a type string describing the field.

Interface Index **getModifiers**

```
public String getModifiers()
```

Returns a string with the field's modifiers, such as "public", "static", "final", etc. If the field has no modifiers, an empty String is returned.

Interface Index **isStatic**

```
public boolean isStatic()
```

Returns whether the field is static (a class variable or method).

Interface Index **toString**

```
public String toString()
```

Returns a String that represents the value of this Object.
Overrides:
toString in class Object

Class sun.tools.debug.RemoteFloat

Class sun.tools.debug.RemoteFloat

```
java.lang.Object
    |
    +----sun.tools.debug.RemoteValue
             |
             +----sun.tools.debug.RemoteFloat
```

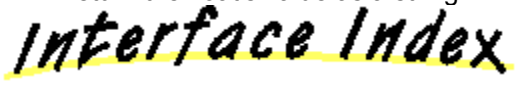
---

public class **RemoteFloat**
extends RemoteValue

The RemoteFloat class extends RemoteValue for floats.
See Also:
RemoteValue, RemoteDebugger

get()
Return the float's value.

toString()
Return the float's value as a string.



typeName()
Print this RemoteValue's type ("float").

get

```
public float get()
```

Return the float's value.

typeName

```
public String typeName()
```

Print this RemoteValue's type ("float").

Overrides:

typeName in class RemoteValue

toString

```
public String toString()
```

Return the float's value as a string.

Overrides:

toString in class Object

Class sun.tools.debug.RemoteInt

Class sun.tools.debug.RemoteInt

```
java.lang.Object
    |
    +----sun.tools.debug.RemoteValue
             |
             +----sun.tools.debug.RemoteInt
```

---

public class **RemoteInt**
extends RemoteValue

The RemoteInt class extends RemoteValue for ints.
See Also:
RemoteValue, RemoteDebugger

RemoteInt(int)

get()

Return the int's value.

toString()

Return the int's value as a string.

typeName()

Print this RemoteValue's type ("int").

Interface Index **RemoteInt**

```
public RemoteInt(int i)
```

Interface Index

Interface Index **get**

```
public int get()
```

Return the int's value.

Interface Index **typeName**

```
public String typeName()
```

Print this RemoteValue's type ("int").
Overrides:
typeName in class RemoteValue

Interface Index **toString**

```
public String toString()
```

Return the int's value as a string.
Overrides:
toString in class Object

Class sun.tools.debug.RemoteLong

Class sun.tools.debug.RemoteLong

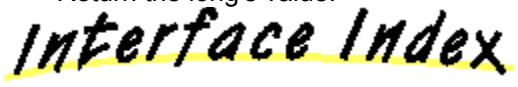
```
java.lang.Object
    |
    +----sun.tools.debug.RemoteValue
             |
             +----sun.tools.debug.RemoteLong
```

---

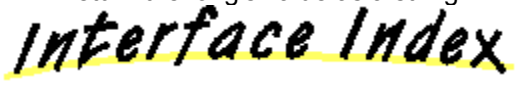
public class **RemoteLong**
extends RemoteValue

The RemoteLong class extends RemoteValue for longs.
See Also:
RemoteValue, RemoteDebugger

get()
Return the long's value.



toString()
Return the long's value as a string.



typeName()
Print this RemoteValue's type ("long").

get

```
public long get()
```

Return the long's value.

typeName

```
public String typeName()
```

Print this RemoteValue's type ("long").

Overrides:

typeName in class RemoteValue

toString

```
public String toString()
```

Return the long's value as a string.

Overrides:

toString in class Object

Class sun.tools.debug.RemoteObject

Class sun.tools.debug.RemoteObject

```
java.lang.Object
    |
    +----sun.tools.debug.RemoteValue
             |
             +----sun.tools.debug.RemoteObject
```

---

public class **RemoteObject**
extends RemoteValue

The RemoteObject class allows access to an object in a remote Java interpreter. Remote objects are not created by the local debugger, but are returned by the remote debugging agent when queried for the values of instance or static variables of known objects (such as classes), or from local (stack) variables. Each remote object has a reference cached by the remote Java interpreter, so that the object will not be garbage-collected during examination. The RemoteDebugger's gc() operation frees references to objects that are no longer being examined.

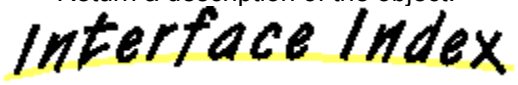
See Also:

RemoteDebugger, RemoteClass, RemoteString, RemoteThread, RemoteThreadGroup

---

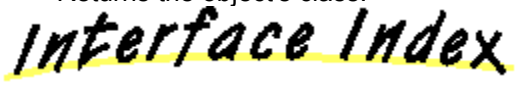
description()

Return a description of the object.



getClazz()

Returns the object's class.



getField(int)

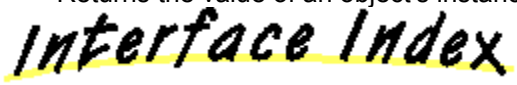
Return an instance variable, specified by slot number.

getField(String)

Return an instance variable, specified by name.

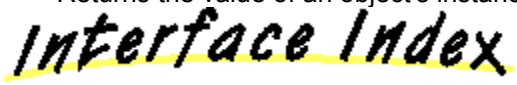
getFieldValue(int)

Returns the value of an object's instance variable.



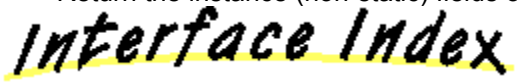
getFieldValue(String)

Returns the value of an object's instance variable.



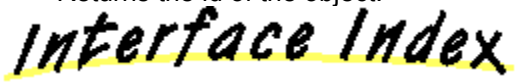
getFields()

Return the instance (non-static) fields of an object.



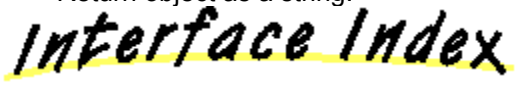
getId()

Returns the id of the object.



toString()

Return object as a string.



typeName()

Returns the RemoteValue's type name ("Object").

typeName

```
public String typeName() throws Exception
```

Returns the RemoteValue's type name ("Object").

Overrides:

typeName in class RemoteValue

getId

```
public final int getId()
```

Returns the id of the object.

getClazz

```
public final RemoteClass getClazz()
```

Returns the object's class.

getFieldValue

```
public RemoteValue getFieldValue(int n) throws Exception
```

Returns the value of an object's instance variable.

Parameters:

n - the slot number of the variable to be returned.

getFieldValue

```
public RemoteValue getFieldValue(String name) throws Exception
```

Returns the value of an object's instance variable.

Parameters:

name - the name of the instance variable

Returns:

the variable as a RemoteValue, or null if name not found.

getFields

```
public RemoteField[] getFields() throws Exception
```

Return the instance (non-static) fields of an object.

getField

```
public RemoteField getField(int n) throws Exception
```

Return an instance variable, specified by slot number.

Parameters:

n - the slot number of the variable to be returned.

getField

```
public RemoteField getField(String name) throws Exception
```

Return an instance variable, specified by name.

Parameters:
name - the name of the instance variable
Returns:
the variable as a RemoteField, or null if name not found.

description

```
public String description()
```

Return a description of the object.
Overrides:
description in class RemoteValue

toString

```
public String toString()
```

Return object as a string.
Overrides:
toString in class Object

Class sun.tools.debug.RemoteShort

Class sun.tools.debug.RemoteShort

```
java.lang.Object
    |
    +----sun.tools.debug.RemoteValue
             |
             +----sun.tools.debug.RemoteShort
```

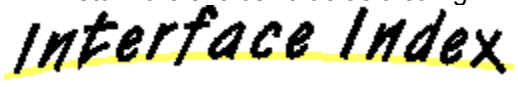
---

public class **RemoteShort**
extends RemoteValue

The RemoteShort class extends RemoteValue for shorts.
See Also:
RemoteValue, RemoteDebugger

get()
Return the short's value.

toString()
Return the short's value as a string.



typeName()
Print this RemoteValue's type ("short").

get

```
public short get()
```

Return the short's value.

typeName

```
public String typeName()
```

Print this RemoteValue's type ("short").

Overrides:

typeName in class RemoteValue

toString

```
public String toString()
```

Return the short's value as a string.

Overrides:

toString in class Object

Class sun.tools.debug.RemoteStackFrame

Class sun.tools.debug.RemoteStackFrame

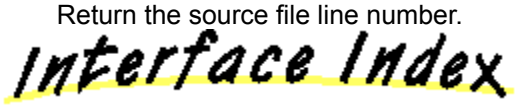
```
java.lang.Object
    |
    +----sun.tools.debug.StackFrame
            |
            +----sun.tools.debug.RemoteStackFrame
```

---

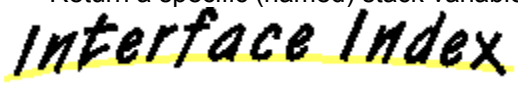
public class **RemoteStackFrame**
extends StackFrame

The RemoteStackFrame class provides access to a stackframe of a suspended thread.
See Also:
RemoteDebugger, RemoteThread

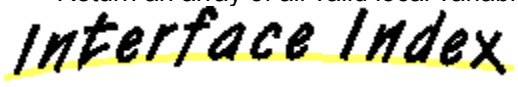
getLineNumber()
Return the source file line number.



getLocalVariable(String)
Return a specific (named) stack variable.



getLocalVariables()
Return an array of all valid local variables and method arguments for this stack frame.



getMethodName()
Get the method name referenced by this stackframe.

getPC()
Get the program counter referenced by this stackframe.

Interface Index **getRemoteClass**()

Get the class this stackframe references.

Interface Index

Interface Index **getLocalVariable**

```
public RemoteStackVariable getLocalVariable(String name) throws Exception
```

Return a specific (named) stack variable. A slot number of -1 indicates that the variable is not currently in scope.

Interface Index **getLocalVariables**

```
public RemoteStackVariable[] getLocalVariables() throws Exception
```

Return an array of all valid local variables and method arguments for this stack frame.

Interface Index **getLineNumber**

```
public int getLineNumber()
```

Return the source file line number.

Interface Index **getMethodName**

```
public String getMethodName()
```

Get the method name referenced by this stackframe.

Interface Index **getPC**

```
public int getPC()
```

Get the program counter referenced by this stackframe.

Interface Index **getRemoteClass**

```
public RemoteClass getRemoteClass()
```

Get the class this stackframe references.

Class sun.tools.debug.RemoteStackVariable

Class sun.tools.debug.RemoteStackVariable

```
java.lang.Object
    |
    +----sun.tools.debug.LocalVariable
            |
            +----sun.tools.debug.RemoteStackVariable
```

---

public class **RemoteStackVariable**
extends LocalVariable

A RemoteStackVariable represents a method argument or local variable. It is similar to a RemoteField, but is much more transient in nature.

See Also:

RemoteField

getName()
Return the name of a stack variable or argument.

getValue()
Return the value of a stack variable or argument.

inScope()
Return whether variable is in scope.

getName

```
public String getName()
```

Return the name of a stack variable or argument.


```
public RemoteValue getValue()
```

Return the value of a stack variable or argument.


```
public boolean inScope()
```

Return whether variable is in scope.

Class sun.tools.debug.RemoteString

Class sun.tools.debug.RemoteString

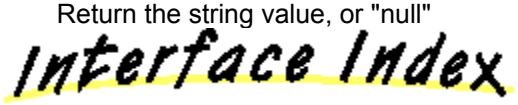
```
java.lang.Object
    |
    +----sun.tools.debug.RemoteValue
             |
             +----sun.tools.debug.RemoteObject
                      |
                      +----sun.tools.debug.RemoteString
```

---

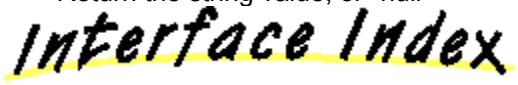
public class **RemoteString**
extends RemoteObject

The RemoteString class allows access to a string in a remote Java interpreter.
See Also:
RemoteDebugger

description()
Return the string value, or "null"



toString()
Return the string value, or "null"



typeName()
Print this RemoteValue's type ("String").

typeName

```
public String typeName()
```

Print this RemoteValue's type ("String").
Overrides:
typeName in class RemoteObject

description

```
public String description()
```

Return the string value, or "null"
Overrides:
description in class RemoteObject

toString

```
public String toString()
```

Return the string value, or "null"
Overrides:
toString in class RemoteObject

Class sun.tools.debug.RemoteThread

Class sun.tools.debug.RemoteThread

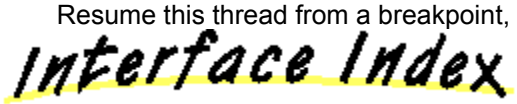
```
java.lang.Object
    |
    +----sun.tools.debug.RemoteValue
             |
             +----sun.tools.debug.RemoteObject
                      |
                      +----sun.tools.debug.RemoteThread
```

---

public class **RemoteThread**
extends RemoteObject

The RemoteThread class allows access to a thread in a remote Java interpreter.
See Also:
RemoteDebugger, RemoteThreadGroup

cont()
Resume this thread from a breakpoint, unless it previously suspended.



down(int)
Change the current stackframe to be one or more frames lower (as in, toward the current program counter).

dumpStack()
Dump the stack.

getCurrentFrame()
Get the current stack frame.

getCurrentFrameIndex()
Return the current stackframe index

getName()

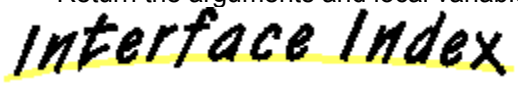
Return the name of the thread.

getStackVariable(String)

Return a stack variable from the current stackframe.

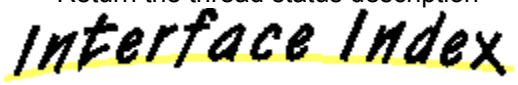
getStackVariables()

Return the arguments and local variable from the current stackframe.



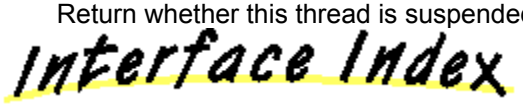
getStatus()

Return the thread status description



isSuspended()

Return whether this thread is suspended.



next()

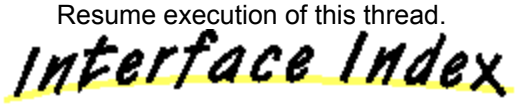
Continue execution of this thread to the next line, but don't step into a method call.

resetCurrentFrameIndex()

Reset the current stackframe

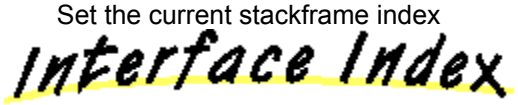
resume()

Resume execution of this thread.



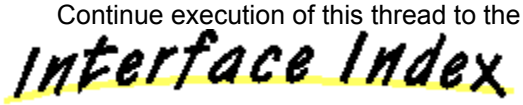
setCurrentFrameIndex(int)

Set the current stackframe index



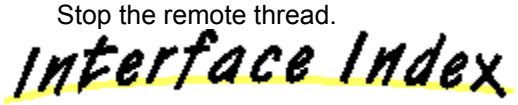
step(boolean)

Continue execution of this thread to the next instruction or line.



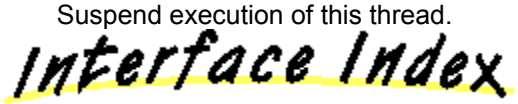
stop()

Stop the remote thread.



suspend()

Suspend execution of this thread.



up(int)

Change the current stackframe to be one or more frames higher (as in, away from the current program counter).

getName

```
public String getName() throws Exception
```

Return the name of the thread.

getCurrentFrameIndex

```
public int getCurrentFrameIndex()
```

Return the current stackframe index

setCurrentFrameIndex

```
public void setCurrentFrameIndex(int iFrame)
```

Set the current stackframe index

resetCurrentFrameIndex

```
public void resetCurrentFrameIndex()
```

Reset the current stackframe

up

```
public void up(int nFrames) throws Exception
```

Change the current stackframe to be one or more frames higher (as in, away from the current program counter).

Parameters:

nFrames - the number of stackframes

Throws: IllegalAccessError

when the thread isn't suspended or waiting at a breakpoint

Throws: ArrayIndexOutOfBoundsException

when the requested frame is beyond the stack boundary

down

```
public void down(int nFrames) throws Exception
```

Change the current stackframe to be one or more frames lower (as in, toward the current program counter).

Parameters:

nFrames - the number of stackframes

Throws: IllegalAccessError

when the thread isn't suspended or waiting at a breakpoint

Throws: ArrayIndexOutOfBoundsException

when the requested frame is beyond the stack boundary

getStatus

```
public String getStatus() throws Exception
```

Return the thread status description

dumpStack

```
public RemoteStackFrame[] dumpStack() throws Exception
```

Dump the stack.

getCurrentFrame

```
public RemoteStackFrame getCurrentFrame() throws Exception
```

Get the current stack frame.

Throws: IllegalAccessError

when the thread isn't suspended or waiting at a breakpoint

suspend

```
public void suspend() throws Exception
```

Suspend execution of this thread.

resume

```
public void resume() throws Exception
```

Resume execution of this thread.

step

```
public void step(boolean skipLine) throws Exception
```

Continue execution of this thread to the next instruction or line.

Parameters:

skipLine - true to execute to next source line, false to next instruction.

Throws: IllegalAccessError

when the thread isn't suspended or waiting at a breakpoint

next

```
public void next() throws Exception
```

Continue execution of this thread to the next line, but don't step into a method call. If no line information is available, next() is equivalent to step().

Throws: IllegalAccessError

when the thread isn't suspended or waiting at a breakpoint

isSuspended

```
public boolean isSuspended()
```

Return whether this thread is suspended.

cont

```
public void cont() throws Exception
```

Resume this thread from a breakpoint, unless it previously suspended.

stop

```
public void stop() throws Exception
```

Stop the remote thread.

getStackVariable

```
public RemoteStackVariable getStackVariable(String name) throws Exception
```

Return a stack variable from the current stackframe.

Returns:

the variable as a RemoteValue, or null if not found.

getStackVariables

```
public RemoteStackVariable[] getStackVariables() throws Exception
```

Return the arguments and local variable from the current stackframe.

Returns:

an array of RemoteValues.

Class sun.tools.debug.RemoteThreadGroup

Class sun.tools.debug.RemoteThreadGroup

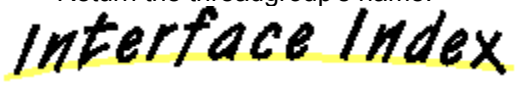
```
java.lang.Object
    |
    +----sun.tools.debug.RemoteValue
             |
             +----sun.tools.debug.RemoteObject
                      |
                      +----sun.tools.debug.RemoteThreadGroup
```

---

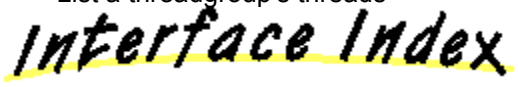
public class **RemoteThreadGroup**
extends RemoteObject

The RemoteThreadGroup class allows access to a threadgroup in a remote Java interpreter.
See Also:
RemoteDebugger, RemoteThreadGroup

getName()
Return the threadgroup's name.



listThreads(boolean)
List a threadgroup's threads



stop()
Stop the remote threadgroup.

getName

```
public String getName() throws Exception
```

Return the threadgroup's name.

stop

```
public void stop() throws Exception
```

Stop the remote threadgroup.

listThreads

```
public RemoteThread[] listThreads(boolean recurse) throws Exception
```

List a threadgroup's threads

Class sun.tools.debug.RemoteValue

Class sun.tools.debug.RemoteValue

```
java.lang.Object
    |
    +----sun.tools.debug.RemoteValue
```

---

public class **RemoteValue**
extends Object
implements AgentConstants

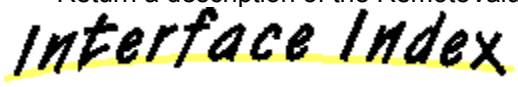
The RemoteValue class allows access to a copy of a value in the remote Java interpreter. This value may be a primitive type, such as a boolean or float, or an object, class, array, etc. Remote values are not created by the local debugger, but are returned by the remote debugging agent when queried for the values of instance or static variables of known objects, or from local (stack) variables.

See Also:

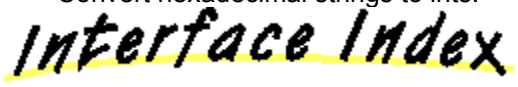
RemoteDebugger, RemoteArray, RemoteBoolean, RemoteByte, RemoteChar , RemoteClass, RemoteDouble, RemoteFloat, RemoteInt, RemoteLong , RemoteObject, RemoteShort, RemoteString, RemoteThread, RemoteThreadGroup

---

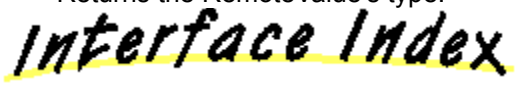
description()
Return a description of the RemoteValue.



fromHex(String)
Convert hexadecimal strings to ints.



getType()
Returns the RemoteValue's type.



isObject()
Returns whether the RemoteValue is an Object (as opposed to a primitive type, such as int).

toHex(int)
Convert an int to a hexadecimal string.

Interface Index **typeName**()

Returns the RemoteValue's type as a string.

Interface Index

Interface Index **getType**

```
public final int getType()
```

Returns the RemoteValue's type.

Interface Index **isObject**

```
public final boolean isObject()
```

Returns whether the RemoteValue is an Object (as opposed to a primitive type, such as int).

Interface Index **typeName**

```
public abstract String typeName() throws Exception
```

Returns the RemoteValue's type as a string.

Interface Index **description**

```
public String description()
```

Return a description of the RemoteValue.

Interface Index **toHex**

```
public static String toHex(int n)
```

Convert an int to a hexadecimal string.

Interface Index **fromHex**

```
public static int fromHex(String hexStr)
```

Convert hexadecimal strings to ints.

Class sun.tools.debug.StackFrame

Class sun.tools.debug.StackFrame

```
java.lang.Object
    |
    +----sun.tools.debug.StackFrame
```

---

public class **StackFrame**
extends Object

The StackFrame class represents a stack frame of a suspended thread.
See Also:
RemoteDebugger, RemoteStackFrame, RemoteThread

StackFrame()

toString()
Returns a String that represents the value of this Object.

StackFrame

```
public StackFrame()
```

Interface Index **toString**

```
public String toString()
```

Returns a String that represents the value of this Object.

Overrides:

toString in class Object

Class Hierarchy

Class Hierarchy

- class java.lang.Object
  - interface java.applet.AppletContext
  - interface java.applet.AppletStub
  - interface java.applet.AudioClip
  - class java.util.BitSet (implements java.lang.Cloneable)
  - class java.lang.Boolean
  - class java.awt.BorderLayout (implements java.awt.LayoutManager)
  - interface java.awt.peer.ButtonPeer (extends java.awt.peer.ComponentPeer)
  - interface java.awt.peer.CanvasPeer (extends java.awt.peer.ComponentPeer)
  - class java.awt.CardLayout (implements java.awt.LayoutManager)
  - class java.lang.Character
  - class java.awt.CheckboxGroup
  - interface java.awt.peer.CheckboxMenuItemPeer (extends java.awt.peer. MenuItemPeer)
  - interface java.awt.peer.CheckboxPeer (extends java.awt.peer.ComponentPeer)
  - interface java.awt.peer.ChoicePeer (extends java.awt.peer.ComponentPeer)
  - class java.lang.Class
  - class java.lang.ClassLoader
  - interface java.lang.Cloneable
  - class java.awt.Color
  - class java.awt.image.ColorModel
    - class java.awt.image.DirectColorModel
    - class java.awt.image.IndexColorModel
  - class java.lang.Compiler
  - class java.awt.Component (implements java.awt.image.ImageObserver)
    - class java.awt.Button
    - class java.awt.Canvas
    - class java.awt.Checkbox
    - class java.awt.Choice
    - class java.awt.Container
      - class java.awt.Panel
        - class java.applet.Applet
      - class java.awt.Window
        - class java.awt.Dialog
          - class java.awt.FileDialog
        - class java.awt.Frame (implements java.awt. MenuContainer)
    - class java.awt.Label
    - class java.awt.List
    - class java.awt.Scrollbar
    - class java.awt.TextComponent
      - class java.awt.TextArea
      - class java.awt.TextField
  - interface java.awt.peer.ComponentPeer
  - interface sun.tools.java.Constants (extends sun.tools.java.RuntimeConstants)
  - interface java.awt.peer.ContainerPeer (extends java.awt.peer.ComponentPeer)
  - class java.net.ContentHandler
  - interface java.net.ContentHandlerFactory

- interface java.io.DataInput
- interface java.io.DataOutput
- class java.net.DatagramPacket
- class java.net.DatagramSocket
- class java.util.Date
- interface sun.tools.debug.DebuggerCallback
- interface java.awt.peer.DialogPeer (extends java.awt.peer.WindowPeer)
- class java.util.Dictionary
  - class java.util.Hashtable (implements java.lang. Cloneable)
    - class java.util.Properties
- class java.awt.Dimension
- interface java.util.Enumeration
- class java.awt.Event
- class sun.tools.debug.Field
  - class sun.tools.debug.RemoteField (implements sun.tools.debug.AgentConstants)
- class java.io.File
- class java.io.FileDescriptor
- interface java.awt.peer.FileDialogPeer (extends java.awt.peer.DialogPeer)
- interface java.io.FilenameFilter
- class java.awt.image.FilteredImageSource (implements java.awt.image. ImageProducer)
- class java.awt.FlowLayout (implements java.awt.LayoutManager)
- class java.awt.Font
- class java.awt.FontMetrics
- interface java.awt.peer.FramePeer (extends java.awt.peer.WindowPeer)
- class java.awt.Graphics
- class java.awt.GridBagConstraints (implements java.lang.Cloneable)
- class java.awt.GridBagLayout (implements java.awt.LayoutManager)
- class java.awt.GridLayout (implements java.awt.LayoutManager)
- class java.awt.Image
- interface java.awt.image.ImageConsumer
- class java.awt.image.ImageFilter (implements java.awt.image.ImageConsumer , java.lang.Cloneable)
  - class java.awt.image.CropImageFilter
  - class java.awt.image.RGBImageFilter
- interface java.awt.image.ImageObserver
- interface java.awt.image.ImageProducer
- class java.net.InetAddress
- class java.io.InputStream
  - class java.io.ByteArrayInputStream
  - class java.io.FileInputStream
  - class java.io.FilterInputStream
    - class java.io.BufferedInputStream
    - class java.io.DataInputStream (implements java.io.DataInput)
    - class java.io.LineNumberInputStream
    - class java.io.PushbackInputStream
  - class java.io.PipedInputStream
  - class java.io.SequenceInputStream
  - class java.io.StringBufferInputStream
- class java.awt.Insets (implements java.lang. Cloneable)
- interface java.awt.peer.LabelPeer (extends java.awt.peer.ComponentPeer)
- interface java.awt.LayoutManager
- interface java.awt.peer.ListPeer (extends java.awt.peer.ComponentPeer)
- class sun.tools.debug.LocalVariable
  - class sun.tools.debug.RemoteStackVariable
- class java.lang.Math
- class java.awt.MediaTracker

- class java.awt.image.MemoryImageSource (implements java.awt.image.ImageProducer)
- interface java.awt.peer.MenuBarPeer (extends java.awt.peer.MenuComponentPeer)
- class java.awt.MenuComponent
  - class java.awt.MenuBar (implements java.awt. MenuContainer)
  - class java.awt.MenuItem
    - class java.awt.CheckboxMenuItem
    - class java.awt.Menu (implements java.awt.MenuContainer)
- interface java.awt.peer.MenuComponentPeer
- interface java.awt.MenuContainer
- interface java.awt.peer.MenuItemPeer (extends java.awt.peer.MenuComponentPeer)
- interface java.awt.peer.MenuPeer (extends java.awt.peer.MenuItemPeer)
- class java.lang.Number
  - class java.lang.Double
  - class java.lang.Float
  - class java.lang.Integer
  - class java.lang.Long
- class java.util.Observable
- interface java.util.Observer
- class java.io.OutputStream
  - class java.io.ByteArrayOutputStream
  - class java.io.FileOutputStream
  - class java.io.FilterOutputStream
    - class java.io.BufferedOutputStream
    - class java.io.DataOutputStream (implements java.io.DataOutput)
    - class java.io.PrintStream
  - class java.io.PipedOutputStream
- interface java.awt.peer.PanelPeer (extends java.awt.peer.ContainerPeer)
- class java.awt.image.PixelGrabber (implements java.awt.image.ImageConsumer)
- class java.awt.Point
- class java.awt.Polygon
- class java.lang.Process
- class java.util.Random
- class java.io.RandomAccessFile (implements java.io.DataOutput , java.io.DataInput)
- class java.awt.Rectangle
- class sun.tools.debug.RemoteDebugger
- class sun.tools.debug.RemoteValue (implements sun.tools.debug.AgentConstants)
  - class sun.tools.debug.RemoteBoolean
  - class sun.tools.debug.RemoteByte
  - class sun.tools.debug.RemoteChar
  - class sun.tools.debug.RemoteDouble
  - class sun.tools.debug.RemoteFloat
  - class sun.tools.debug.RemoteInt
  - class sun.tools.debug.RemoteLong
  - class sun.tools.debug.RemoteObject
    - class sun.tools.debug.RemoteArray
    - class sun.tools.debug.RemoteClass
    - class sun.tools.debug.RemoteString
    - class sun.tools.debug.RemoteThread
    - class sun.tools.debug.RemoteThreadGroup
  - class sun.tools.debug.RemoteShort
- interface java.lang.Runnable
- class java.lang.Runtime
- interface sun.tools.java.RuntimeConstants
- interface java.awt.peer.ScrollbarPeer (extends java.awt.peer.ComponentPeer)
- class java.lang.SecurityManager
- class java.net.ServerSocket

- class java.net.Socket
- class java.net.SocketImpl
- interface java.net.SocketImplFactory
- class sun.tools.debug.StackFrame
  - class sun.tools.debug.RemoteStackFrame
- class java.io.StreamTokenizer
- class java.lang.String
- class java.lang.StringBuffer
- class java.util.StringTokenizer (implements java.util.Enumeration)
- class java.lang.System
- interface java.awt.peer.TextAreaPeer (extends java.awt.peer.TextComponentPeer)
- interface java.awt.peer.TextComponentPeer (extends java.awt.peer.ComponentPeer)
- interface java.awt.peer.TextFieldPeer (extends java.awt.peer.TextComponentPeer)
- class java.lang.Thread (implements java.lang.Runnable)
- class java.lang.ThreadGroup
- class java.lang.Throwable
  - class java.lang.Error
    - class java.awt.AWTError
    - class java.lang.LinkageError
      - class java.lang.ClassCircularityError
      - class java.lang.ClassFormatError
      - class java.lang.IncompatibleClassChangeError
        - class java.lang.AbstractMethodError
        - class java.lang.IllegalAccessError
        - class java.lang.InstantiationError
        - class java.lang.NoSuchFieldError
        - class java.lang.NoSuchMethodError
      - class java.lang.NoClassDefFoundError
      - class java.lang.UnsatisfiedLinkError
      - class java.lang.VerifyError
    - class java.lang.ThreadDeath
    - class java.lang.VirtualMachineError
      - class java.lang.InternalError
      - class java.lang.OutOfMemoryError
      - class java.lang.StackOverflowError
      - class java.lang.UnknownError
  - class java.lang.Exception
    - class java.awt.AWTException
    - class java.lang.ClassNotFoundException
    - class java.lang.CloneNotSupportedException
    - class java.io.IOException
      - class java.io.EOFException
      - class java.io.FileNotFoundException
      - class java.io.InterruptedIOException
      - class java.net.MalformedURLException
      - class java.net.ProtocolException
      - class java.net.SocketException
      - class java.io.UTFDataFormatException
      - class java.net.UnknownHostException
      - class java.net.UnknownServiceException
    - class java.lang.IllegalAccessException
    - class java.lang.InstantiationException
    - class java.lang.InterruptedException
    - class java.lang.NoSuchMethodException
    - class java.lang.RuntimeException
      - class java.lang.ArithmeticException

        - class java.lang.ArrayStoreException
        - class java.lang.ClassCastException
        - class java.util.EmptyStackException
        - class java.lang.IllegalArgumentException
            - class java.lang.IllegalThreadStateException
            - class java.lang.NumberFormatException
        - class java.lang.IllegalMonitorStateException
        - class java.lang.IndexOutOfBoundsException
            - class java.lang.ArrayIndexOutOfBoundsException
            - class java.lang.StringIndexOutOfBoundsException
        - class java.lang.NegativeArraySizeException
        - class java.util.NoSuchElementException
        - class java.lang.NullPointerException
        - class java.lang.SecurityException
- class java.awt.Toolkit
- class java.net.URL
- class java.net.URLConnection
- class java.net.URLEncoder
- class java.net.URLStreamHandler
- interface java.net.URLStreamHandlerFactory
- class java.util.Vector (implements java.lang.Cloneable)
    - class java.util.Stack
- interface java.awt.peer.WindowPeer (extends java.awt.peer.ContainerPeer)

file:/D|/APIGOLD/ALLNAM~1.HTM#StackFrame()

file:/D|/APIGOLD/ALLNAM~1.HTM#toString()

file:/D|/APIGOLD/JAVAA~22.HTM#drawImage(java.awt.Image, int, int, int, int, java.awt.image.ImageObserver)

file:/D|/APIGOLD/JAVAA~33.HM#HEIGHT

URL

file:/D|/APIGOLD/JAVAL~53.HTM#capacit()