

Bluette is. ..

You can compile and run java source codes easily in the environment.

With 'Form Designer' you can build Applets and Applications easily.

With 'Class Viewer' you can browse information about your class in addition to whole class hierarchy.

You can change options for editor and other environment as you want.

In version 0.7(or higher) you can define the default source code for java components.

You can set your own short-keys for your convenience.

Arguments for Applet or Application can be passed easily through argument passing windows.

With 'Semantic Prototyper' you can design and build your program easily defining and describing the behaviour, fields in Bluette ver 0.9(or higher).

Bluette provides on-line help system you can use in every where in its environment and also provides various useful edit functions.

Bluette provides simple 'Syntax Coloring' functionality.

Automatic indentation, smart Tab(Backspace) function is included in editor for your convenience.

Compile and Run

You can compile java source codes selecting 'Compile' menu item or pressing Ctrl-F9.

You can run your program selecting 'Run' menu item or pressing 'F9'.

Sometimes Bluetecan not recognize source codes' identification, in that case you can manually choose program that runs your program selecting 'Java' or 'Appletviewer' in 'Compile' menu.

While your program is running you can stop your program by selecting 'Reset' menu item or pressing Ctrl-F2.


Using Components


Component palettes in main window help you build your prototypings for java application and applet.

After opening new form and choosing a component you want if you click your mouse down on the form, the default source codes go into the editor.

Using Properties

When you newly create a form or component, or you change the selected component in a form you can see kinds and values of the selected component or form in the [Object Inspector](#).

To input values in properties with  button, you have to push that button to see the choice you can choose. If you push this


 button, you can see another window for editing those properties and you only have to modify or set values as you want to.

Otherwise you can directly feel those properties with valid values which are not out of its range.

Setting Event handler

Each component has its own or common Event.

Select a component to set event handler and double click the event field you want to handle, then you can see the default event handling codes and the new event handler is registered in [Object Inspector](#).

You can also use  button to reuse existing event handlers.

Openning New File


If you select 'File' menu and 'New' menu item, you can see two child menu to select new file and new form. If you choose 'New File' menu item, you can see another child menus for choosing the kind of a new file. You can select the kind of new file and then editor will open a new file you chose.

Selecting Form

If you select 'File' menu and 'New' menu item, you can see two child menu to select new file and new form. If you choose 'New Form' menu item, you can see another child menus for choosing the kind of a new form. You can select the kind of new form and then a new [form designer](#) will be created and editor will open a new file you chose with default codes.


Working with a Form

With [Form designer](#) you can easily build your application's interface quickly.

You can swap form and editor pressing F12 or pushing a  button in [Speed Bar](#).

To rename a form you have to change the 'Name' field in Object Inspector, but notice that if you change the name of form, the name of class and file will be also changed with a new name.

Building Menus

To build menu(menu bar) put a  (MenuBar component) in the form designer from component palette and double click it to show menu designer.

All the changes and setting can be done in [menu designer](#) and you can also launch it in 'Items' property in [properties page](#) in Object Inspector.

Keyboard Events (JDK 1.0.2)

KEY_ACTION

You pressed an Action Key such as function key or control key like PgUp, PgDn.
You can use 'key' and 'modifiers' variables to inspect what kind of key has been pressed.

KEY_ACTION_RELEASE

You released an Action Key such as function key or control key like PgUp, PgDn.
You can use 'key' and 'modifiers' variables to inspect what kind of key has been pressed.

KEY_PRESS

You pressed non Action Key including ASCII control characters.
You can use 'key' and 'modifiers' variables to inspect what kind of key has been pressed.

KEY_RELEASE

You released non Action Key including ASCII control characters.
You can use 'key' and 'modifiers' variables to inspect what kind of key has been pressed.

The Source of Java Events (JDK 1.0.2)

the Keyboard

the Mouse

window creation, destruction, and movement

scroll bar activities

list box item selection and deselection

change of input focus

component and menu actions

Window Creation, Destruction, and Movement (JDK 1.0.2)

WINDOW_DEICONIFY

Window is being blown back up from a icon.

WINDOW_ICONIFY

Window is being shrunk down to an icon.

WINDOW_DESTROY

Window is being destroyed.

WINDOW_EXPOSE

Window is being shown.

WINDOW_MOVED

Window is being moved.

ScrollBar Events (JDK 1.0.2)

SCROLL_ABSOLUTE

User is dragging thumb of a scrollbar.
You can use a 'value' variable for the position of thumb.

SCROLL_LINE_DOWN

User is slightly scrolling down using arrows which are on both ends of scrollbars.
You can use a 'value' variable for the position of thumb.

SCROLL_LINE_UP

User is slightly scrolling up using arrows which are on both ends of scrollbars.
You can use a 'value' variable for the position of thumb.

SCROLL_PAGE_DOWN

User is scrolling down pressing background of scrollbars.
You can use a 'value' variable for the position of thumb.

SCROLL_PAGE_UP

User is scrolling up pressing background of scrollbars.
You can use a 'value' variable for the position of thumb.

List box Item Selection and Deselection (JDK 1.0.2)

LIST_SELECT

User selected list box item.

LIST_DESELECT

User deselected list box item when 'MultiSelect' property of a list box is true.

Change of Input Focus (JDK 1.0.2)

GOT_FOCUS

Object which is passed through 'target' variable got focus.
Keystrokes will be directed here now.

LOST_FOCUS

Object which is passed through 'target' variable lost focus.
Keystrokes will be directed elsewhere now.

Components and Menu Action (JDK 1.0.2)

ACTION_EVENT

User clicked a button or menu item.

(Or user got components do their default actions such as check in checkbox, return key in textfield, etc..)

Mouse Event (JDK 1.0.2)

MOUSE_DOWN

User pressed a mouse button.

Variable 'x' and 'y' have the mouse position, 'modifiers' has information about the status of shift, control, alt keys. The 'count' variable tells you the number of clicks user did.

MOUSE_UP

User released a mouse button.

Variable 'x' and 'y' have the mouse position, 'modifiers' has information about the status of shift, control, alt keys.

MOUSE_DRAG

User moves a mouse pressing a mouse button.

Variable 'x' and 'y' have the mouse position, 'modifiers' has information about the status of shift, control, alt keys.

MOUSE_MOVE

User moves a mouse.

Variable 'x' and 'y' have the mouse position, 'modifiers' has information about the status of shift, control, alt keys.

MOUSE_ENTER

User moved a mouse and the mouse entered a windowed component.

Variable 'x' and 'y' have the mouse position.

MOUSE_EXIT

User moved a mouse and the mouse left the client area of a windowed component.

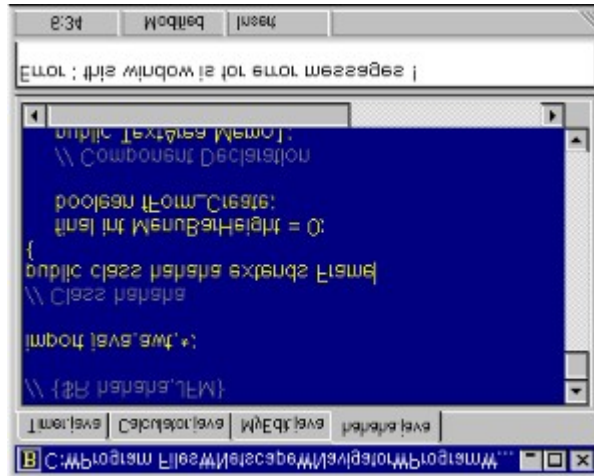
Variable 'x' and 'y' have the mouse position.

When you write your program...

Bluette is designed just to implement the principle of RAD tool, it does not take care of your mistakes basically. But now Bluette became a free software and left its private use range I had to consider to handle errors. Most of fatal errors are being handled but there are still lots of unhandled errors which can be called as 'bugs'. When I have time to deal with those holes of safety I will handle them, but do not tell me "right now".

Please do not exploit(?) a baby Bluette.

Code Editor



You can open and edit multiple documents and use simple 'Syntax coloring' facility of BlueJ.

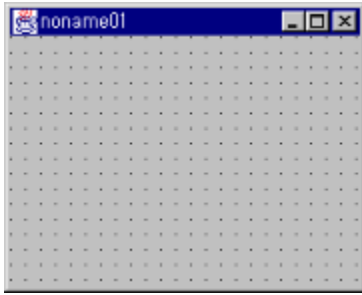
You can see the line and column numbers in the status bar and can set a file as read-only.

Automatic indentation and [keyboard templet](#) can help you write programs easily.

You can see error messages in the 'message window' below the editor when there is any error during compilation and running your programs.

You can set options in 'Option' menu and use other useful function pressing right mouse button on the editor window.

Form (Form Designer)



The form is the basic programming environment for building your applications or applets.

You can select any component from component palette and put it onto the form to build your program interface.

Grids on the form help you to align component and you can use 'alignment palette' to align components precisely.

Standard Component Page



Standard component page holds Window common controls such as following :

[MenuBar](#)

[Label](#)

[TextField](#)

[TextArea](#)

[Button](#)

[CheckBox](#)

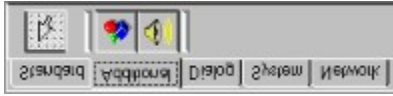
[List](#)

[Choice](#)

[Panel](#)

[ScrollBar](#)

Additional Component Page



Additional component page holds components which play a important but optional role in Java programming such as following :

[Image](#)
[AudioClip](#)

Dialog Component Page

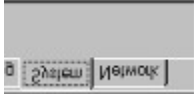


Dialog component page holds a couple of dialog box component that are provided by operating system such as following :

[OpenDialog](#)

[SaveDialog](#)

Other Component Page



System component page and Network component page are not implemented yet.

Not only that I did have enough time to study the specifications and properties of each component and to make it a class, but also I am not that expert programmer to complete those huge work at once. Maybe you can use those components after one of my colleagues finish his thesis. For the time being you can use [Keyboard template](#) for restoring your codes for those components and retrieve or edit those predefined codes when you need them.

Adding or Removing Components

I'm so sorry... Bluette is not designed as Delphi which is composed of developing environment and library written in same language and not able to include Java class file freely we had to give users another way. After much debate we would like to redesign Bluette to support those capability of adding and removing components.

Keyboard Shortcuts

Bluette provides following keyboard shortcuts :

Ctrl-C : Copy a selected text from editor to clipboard.

Ctrl-F : Find text.

Ctrl-L : Goto the line number you want.

Ctrl-O : Open file.

Ctrl-P : Print current file.

Ctrl-R : Replace text while searching .

Ctrl-S : Save current file.

Ctrl-T : Open [keyborad Templet](#) window.

Ctrl-V : Paste a text from clipboard to editor.

Ctrl-X : Cut a selected text from editor to clipboard.

Ctrl-Z : Undo the latest edit command.

Ctrl-F2 : Stop running of applet or application.

Ctrl-F4 : Close current file and form if editor file has a form.

Ctrl-F9 : Compile source file.

Ctrl-Home : Move cursor the first column in a line in a editor.

Ctrl-End : Move cursor the last column in a line in a editor.

Ctrl-Del : Delete a word from the current cusor position forward.

Ctrl-Backspace : Delete a word from the current cursor position backward.

F1 : Show help.

F9 : Run applet or application.

F11 : Show object Inspector window.

F12 : Open editor window if editor window is closed, or show corresponding form to current file in the editor window if you are in the editor window, or show a correponding page in a editor window if you are working in the form designer.

Keyboard Template

In addition to shortcuts which are provided by Bluette, you can also define your own shortcuts using 'Ctrl' and 'Alt' keys.

Those defined shortcuts can be saved into a file you can use various set of shortcuts having made a couple of files including each set of shortcuts you defined previously

Menu Designer



Menu designer is a window , which lets you design main menu that is called 'MenuBar' in Java. You can build tree-structured your own menu system using visual interface..

To make new menu item you have to move to the parent menu item of it first and then push 'New MenuItem' button.

You can make any menu item as a separator pushing 'Separator' button , can change the checked state of menu item pushing 'Checked' button, and can desable a menu item or menu pushing 'Enabled' button.

You can see the event handler of the selected menuitem on the status bar below the dialog box.

File Menu

You can use 'File' menu when you create new form or new file, or open, save and print a file.

'File' menu contains following menu items :

New

'New' menu item has following sub-menu items :

New File

'New File' menu item has two sub-menu item representing the kind of files you can select.

If you select 'Java file' menu item new Java file will be created and if you select 'HTML file' menu item you can see new blank HTML file in the editor window.

New Form

'New Form' menu item has three sub-menu item representing the kind of forms you can select.

'for Applet' menu item creates a form for an applet,

'for Application' menu item creates a form for an application.

'for Another Frame' is for a form which is for secondary frame or building general type of a user form, so that does not have a 'main()' function.

Each time you create a new form default source codes for those forms are inserted into the editor window with extension '.java'.

Open

Java file or HTML file which are previously edited and saved you can use 'Open' menu item or simply press the shortcut 'Ctrl-O' for that command.

Reopen

Once file is loaded into the editor, the name of that file will be added to sub-menus of 'Reopen' menu when you close that file only if it is not new file.

To open that file again you only have to open 'Reopen' menu item and select file that you want to reopen.

The most recently used file is on the top of the 'Reopen' menu item like stack structure.

If you select any file in 'Reopen' menu that file will be removed from 'Reopen' menu again.

Save

To save files after modification either select 'Save' menu item or press shortcut 'Ctrl-S' for that command.

Save as

Select 'Save As' menu item if you want to save your file with different name after modification.

Filters in Open dialogbox and Save dialogbox can be edited in 'Environment' menu item in 'Option' menu.

Editing file filter enables you handle other files easily in Open and Save dialogbox.

Close

To close files select 'Close' menu item or press shortcut 'Ctrl-F4' for that command.

If the file being closed is not new file the name of that file will be added to 'Reopen' menu and you can reopen it later you need it.

Close All

To close all the files currently being open select 'Close All' menu item.

Names of all the closed file are added to 'Reopen' menu.

Until you create new file or open any file into the editor, the editor window will be hidden.

Print

Prints file in the current page and you can also press shortcut 'Ctrl-P' for this command.

Print Setup

'Print Setup' menu item open a dialogbox to set or reset your printer for your own purpose.

Exit

terminates Bluette.

If there is any file which is not saved Bluette will ask you whether you save that file or not. If you choose 'Cancel' the termination will be canceled.

Edit menu

Edit menu is for text processing like copy and paste or undo of text via the clipboard.

'Edit' menu contains following menu items :

Undo

You can undo your last edit command such as 'Cut', 'Delete', 'Paste' by selecting 'Undo' menu item or pressing shortcut 'Ctrl-Z'.

Cut

If you select 'Cut' menu item after selecting text in the editor, that text will be deleted and placed on the clipboard.

Copy

If you select 'Copy' menu item after selecting text in the editor, that text will be placed on the clipboard.

Paste

If you select 'Paste' menu item after copying or cutting text, that text will be inserted from the current cursor position.

Delete

If you select 'Delete' menu item after selecting text, that text will be removed from editor and not copied to clipboard.

Search Menu

You can use 'Search' menu when use need to find or replace some text or need to go to the specific line.

'Search' menu contains following menu items :

Find

Finds text that you want to search.

Replace

Find text and replace it with another text you gave.

Find Next

Finds text from the current cursor position with text that you recently searched.

Goto Line Number

Moves cursor to the specific line in the editor.

View Menu

'View' menu contains following menu items :

Editor Window

Shows the editor window. Usually this menu item is disabled since almost you will see the editor window.

Object Inspector

Shows 'Object Inspector' window. Usually this menu item is also disabled since you will see the 'Object Inspector' window.

Class Viewer

Shows [Class Viewer](#) with the information about your Java source code and class with whole class hierarchy after browsing current file.

Status Bar

Shows or hides the 'Status Bar' below the editor window.

Speed Bar

Shows or hides the 'Speed Bar' panel in the main window.

Run Menu

You can compile and run your Java applications and applets or HTML file which contains Java applet, or can stop running them, and even can pass arguments.

‘Run’ menu contains following menu items :

Compile

Compiles Java source code.

Run

Runs programs depending on the kind of source.

HTML file : Blueette launches an Appletviewer to see the Applet in it.

Java file : if source is an applet Blueette launches Java runtime or if source is an application Blueette launches an Appletviewer.

Blueette decides which program should be run judging from source program’s identity. But sometimes Blueette can not recognize whether it is an applet or an application, then you have to run proper application manually between ‘Java’ and ‘Appletviewer’.

Reset

Stops your program’s running.

Arguments

If your program needs arguments you can open windows for passing argument to them.

Applets and applications are different in passing arguments, so you have to open different window to [pass arguments](#). Each window has help a button for your reference.

Java

Runs Java runtime. Even though your source is Java application if Blueette can not understand it, you can select this command.

Javah

Runs ‘Javah.exe’ of JDK to generate C header file and C source file.

Javadoc

Runs ‘Javadoc.exe’ of JDK to generate API documents in HTML file type including declaration parts and comments from Java source file.

Appletviewer

Runs Appletviewer. Even though your source is Java applet if Blueette can not understand it, you can select this command.

Jdb (Java Debugger)

Runs Jdb which is a command line debugger of JDK.

Option Menu

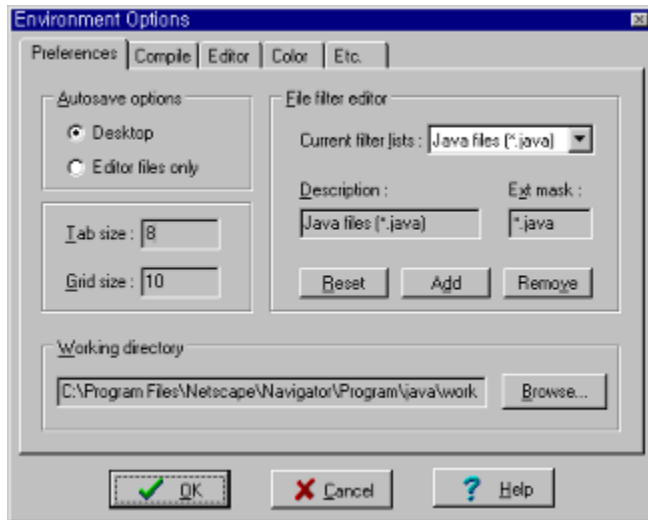
Sets options for the entire Bluette programming environment.

'Option' menu contains following menu items :

Environment

This is for overall properties and options for Bluette.

You can see the new window for setting your own environment options.



In this [Environment options window](#) you can set directory path, editor options, compiler path and class path, etc..

Templet

Bluette provides the keyboard templet function which can help you write your program more easiliy.

Details are in [Keyboard templet window](#) section in Help file.

Tools menu

Bluette will include a couple of useful tools such as image editor, simple evaluator and semantic prototyper as soon as possible.

Current 'Tools' menu contains following menu items :

[Semantic Prototyper](#)
[Evaluator](#)

Help Menu

You can use Bluette on-line help system whenever you need help on Java language as well we Bluette.

‘Help’ menu contains following two menu items :

Help Topic

Opens Help content window.

Bluette is...

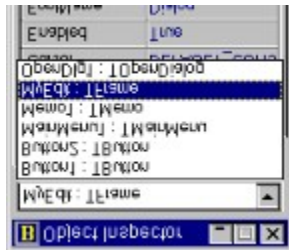
Opens a small dialogbox which introduces Bluette.

Object Inspector

Object Inspector is...

The Object Inspector is the gateway between your application's visual appearance and the code that makes your application run.

The Object Inspector enables you to
set design-time properties for components you have placed on a form (or for the form itself), and
create and help you navigate through event handlers.



The Object selector at the top of the Object Inspector is a drop-down list containing all the components on the active form and it also displays their object type. This lets quickly select different components on the current form.

You can resize the columns of the Object Inspector by dragging the separator line to a new position.

Object Inspector has two pages

[Properties Page](#)

[Events Page](#)

Speed Bar

Speed Bar is...

The SpeedBar provides shortcuts for menu commands. The graphic below shows the Bluette SpeedBar.



If you let your mouse stay over each buttons you can see help message about those buttons.

Semantic Prototyper

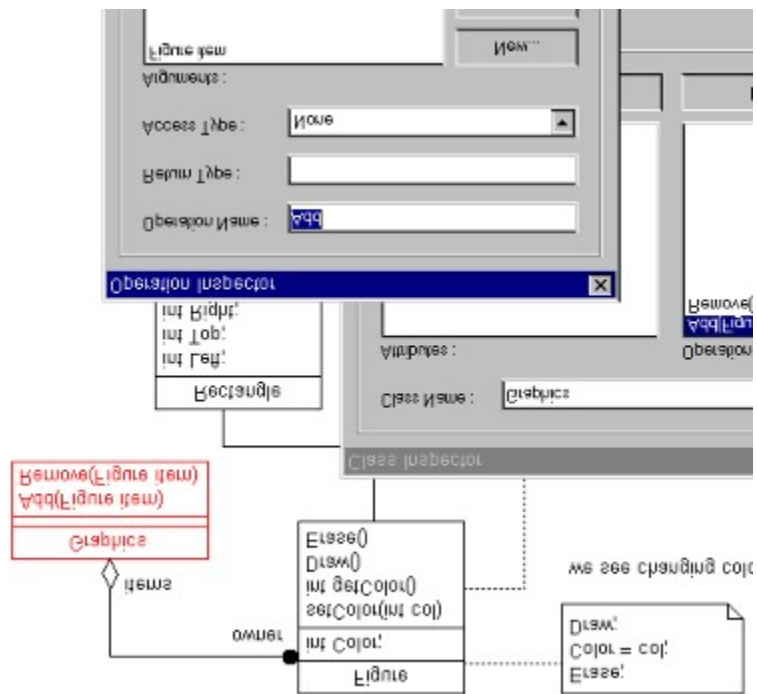
Semantic Prototyper is ...

Semantic Prototyper is a tool implemented by one of my junior, which is not perfect but so useful tool. It is almost over to consider the compatibility to Bluette and utility itself.

Object Description Language and Rumbaugh's Object Model Notation for graphical notation are adapted and its design mechanism is shown below.

Being used with Bluette you can get more advantage in building, generating sources and design your application through the easy prototyping.

For details refer to <http://www.donga.ac.kr/~leemk>



Properties Page

The Properties page of the Object Inspector enables you to set design-time properties for components on your form, and for the form itself. You can set run-time properties by writing source code inside event handlers.

The Properties page displays only the properties of the component that is selected on the form.

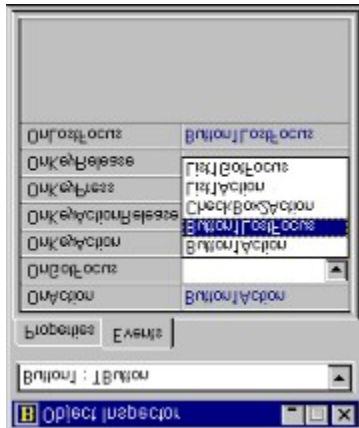
By setting properties at design time you are defining the initial state of a component.



Events Page

The Events page of the Object Inspector enables you to connect forms and components to program events. When you double-click an event from the Events page, Blueette creates an event handler and switches focus to the Code Editor. In the Code Editor, you write the code inside event-handlers that specifies how a component or form responds to the a particular event.

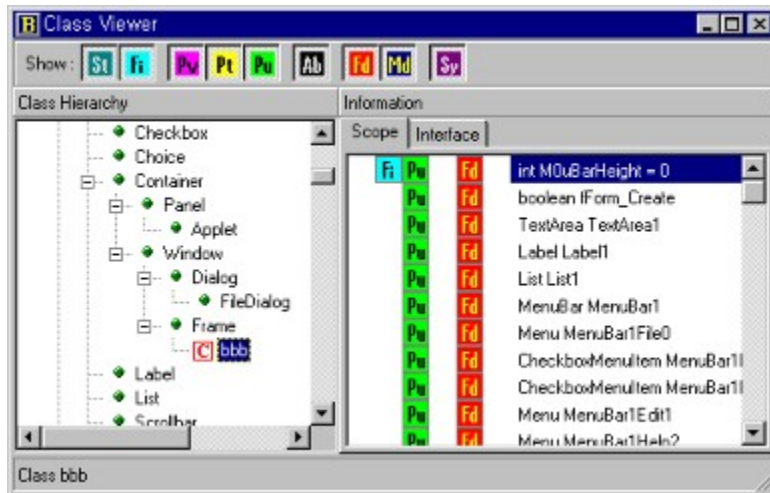
The Events page displays only the events of the component that is selected in the form.



Class Viewer

Class viewer reads Java source file and show you the information about class you wrote.

You can see Classes, Interfaces, type or value of fields and methods, inheritance information, and access control information with following interface..



Scope page shows you information of inside classes and Interface page shows you information for class relation and Interface that it adapted.

Environment Options Window

Environment options window has following pages for your convenience :

Preferences page

Preferences page contains fields for 'autosave option', 'tab size', 'grid size of form', 'working directory' and 'file filter'.

Compile page

Compile page contains fields for 'compile option', 'class path' and compiler name'.

Editor page

Editor page contains fields for general 'editor option' and 'font'.

Color page

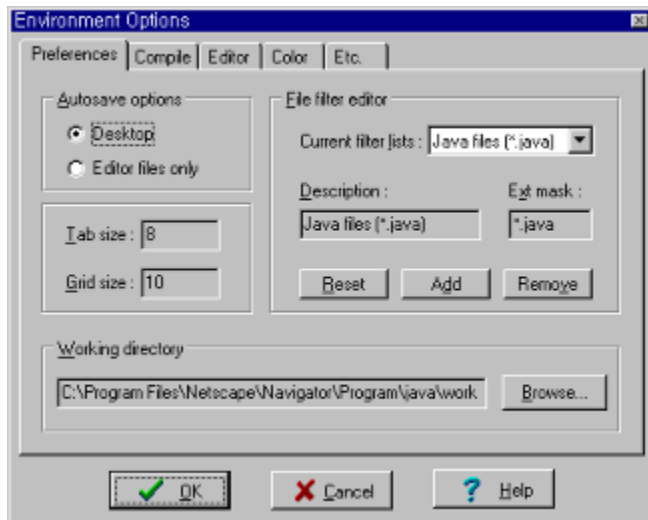
Color page contains fields for 'color of lexical element' and 'foreground and backgroun of Code Editor'.

Etc. page

Etc. page contains fields for 'default width and height of an applet'.

Preferences Page

In Preferences page you can change following options :



Autosave options : If you select “Editor files only” option Bluette only saves your modified files when it exit. If you select “Desktop” option Bluette save desktop arguments in addition to the function of “Editor files only” option.

Tab size : Bluettere replaces a tab character with a couple of space. So tab size determines how many spaces should be inserted instead of tab character.

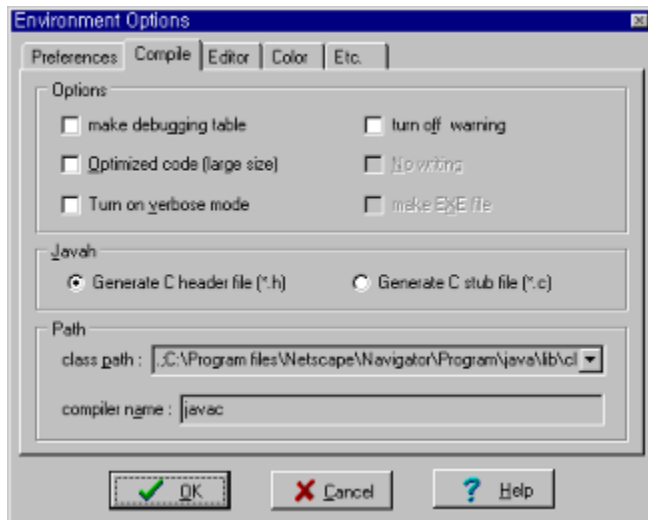
Grid size : The grid on the form help you place components precisely.

Working directory : Working directory is the default directory where the file is saved or open when you are working with files.

File filter editor : You can edit file filter for using open and save dialog box to open(save) files easily.

Compile Page

In Compile page you can change following options :



Compiler option : sets compiler option.

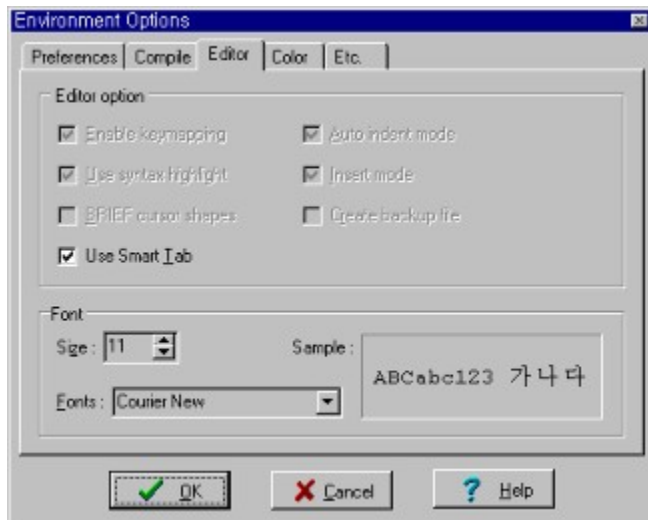
Javah options : option for running 'Javah'.

Class path : Compiler and run-time need classpath variable to be set before compile and running your program.

Compiler name : The name of compiler such as 'javac' or 'jc'.
Bluette only support 'javac' of JDK now and other compilers will be supported later.

Editor Page

In Editor page you can change following options :



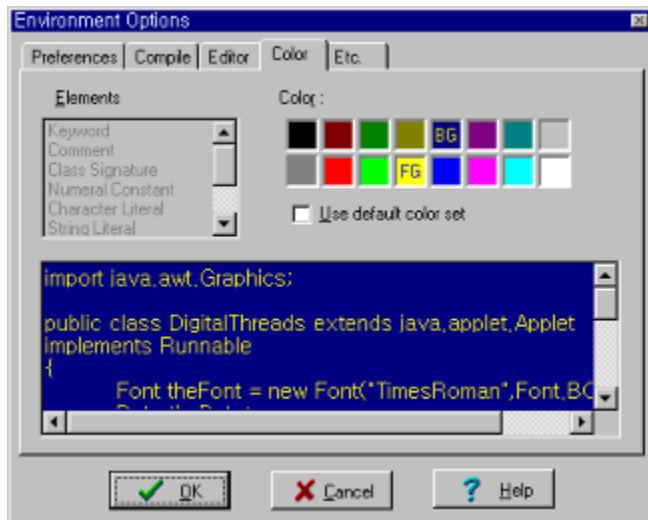
Editor options : general editor options.

Font : set font size, name of Code Editor.

The look of font could be different from the one in sample and Code Editor's.

Color Page

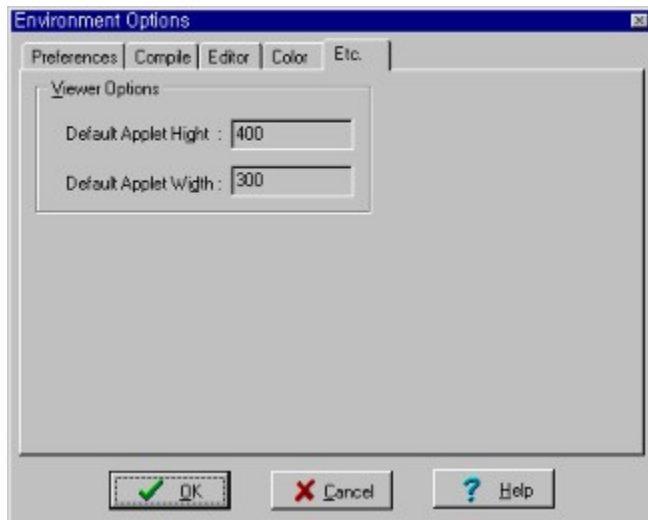
In Color page you can change following options :



Color : Sets the color of each element.

Etc. Page

In Etc. page you can change following options :



Default Applet Height : If you don't specify the height of you form for an Applet, this value will be used for default applet height.

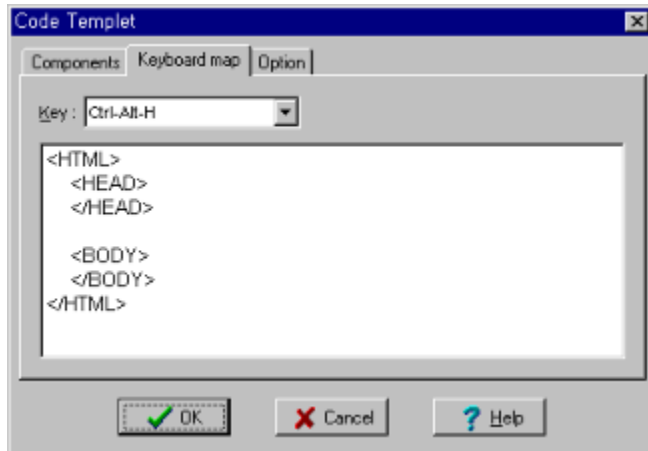
Default Applet Width : If you don't specify the width of you form for an Applet, this value will be used for default applet width.

Keyboard Templet Window

You can save frequently used text in this Keyboard templet and use them when you need simply pressing a couple of key combination.

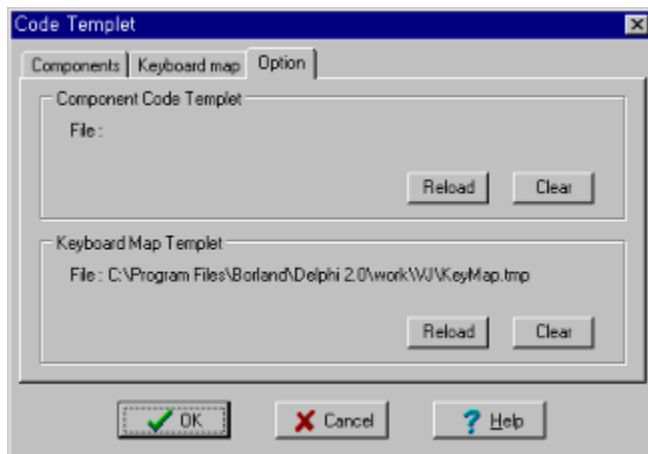
Shortcuts can be defined as the combination of 'Ctrl', 'Alt' and a character from 'A' to 'Z'.

For example, you can define shortcut 'Ctrl-Alt-H' as below :



You can save your set of shortcuts to text files and alternate those templet files.

You can set the name of files in 'Option' page.



If you push the "Reload" button you can see an open dialog box for reset the templet file.

If you push the "Clear" button you can clear all the setting of shortcuts.

'Component Code Templet' is not supported in this version.

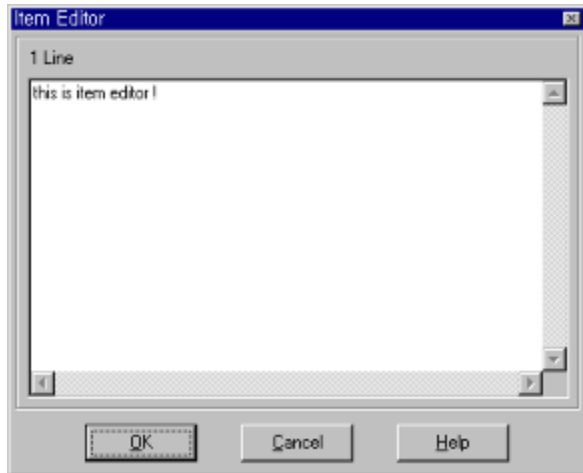
Simple Evaluator

Sorry, this function is not supported in this verion.

Item Editor

This window is for editing items of TextArea, List and Choice component.

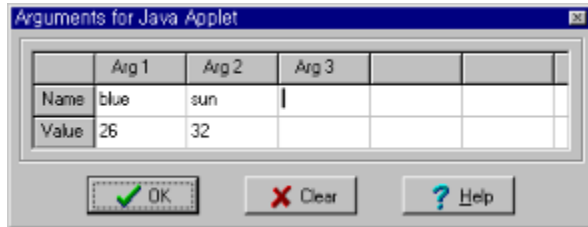
The content of item editor will be the initial value of each component.



Arguments Passing Windows

Bluette provides Arguments Passing Window to pass arguments to applets and applications easily.

Arguments Passing Window for an Applet has two fields of Name and Value field, and you can input proper values for those arguments with no limit.



The dialog box titled "Arguments for Java Applet" contains a table with columns for argument names and values. The first two columns are labeled "Arg 1" and "Arg 2". The "Name" row shows "blue" for Arg 1 and "sun" for Arg 2. The "Value" row shows "26" for Arg 1 and "32" for Arg 2. There are three buttons at the bottom: "OK" (with a green checkmark), "Clear" (with a red X), and "Help" (with a blue question mark).

	Arg 1	Arg 2	Arg 3		
Name	blue	sun			
Value	26	32			

Arguments Passing Window for an Application has only a single line edit box for command line argument passing

And those arguments passed once will be saved and you can reuse them any time until you exit Bluette.



The dialog box titled "Arguments for Java application" features a single text input field labeled "Enter arguments list:". The field contains the text "arg1 arg2 arg3". There are two buttons at the bottom: "OK" (with a green checkmark) and "Help" (with a blue question mark).

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)

Class java.lang.Object

Declaration

```
public class Object
```

Description

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

Constructors

[Object\(\)](#)

Methods

[clone\(\)](#)

[equals\(Object\)](#)

[finalize\(\)](#)

[getClass\(\)](#)

[hashCode\(\)](#)

[notify\(\)](#)

[notifyAll\(\)](#)

[toString\(\)](#)

[wait\(long\)](#)

[wait\(long, int\)](#)

[wait\(\)](#)

Object

Applies to

`class java.lang.Object`

Declaration

```
public Object()
```

getClass

Applies to

class java.lang.Object

Declaration

```
public final Class getClass()
```

Description

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

Applies to

[class java.lang.Object](#)

Declaration

```
public int hashCode()
```

Description

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

Applies to

class java.lang.Object

Declaration

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

Description

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

Applies to

class java.lang.Object

Declaration

protected Object clone() throws CloneNotSupportedException

Description

Creates a clone of the object. A new instance is allocated and a bitwise clone of the current object is placed 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

Applies to

class java.lang.Object

Declaration

```
public String toString()
```

Description

Returns a String that represents the value of this Object. It is recommended that all subclasses override this method.

notify

Applies to

class java.lang.Object

Declaration

```
public final void notify()
```

Description

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

Applies to

class java.lang.Object

Declaration

```
public final void notifyAll()
```

Description

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

Applies to

class java.lang.Object

Declaration

```
public final void wait(long timeout) throws InterruptedException  
public final void wait(long timeout, int nanos) throws InterruptedException  
public final void wait() throws InterruptedException
```

Description

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.

wait

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.

finalize

Applies to

class java.lang.Object

Declaration

protected void finalize() throws Throwable

Description

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.

Interface `java.applet.AppletContext`

Declaration

```
public interface AppletContext  
    extends Object
```

Description

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.

Methods

[getApplet\(String\)](#)

[getApplets\(\)](#)

[getAudioClip\(URL\)](#)

[getImage\(URL\)](#)

[showDocument\(URL\)](#)

[showDocument\(URL, String\)](#)

[showStatus\(String\)](#)

getAudioClip

Applies to

Interface `java.applet.AppletContext`

Declaration

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

Description

Gets an audio clip.

getImage

Applies to

Interface `java.applet.AppletContext`

Declaration

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

Description

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

Applies to

Interface `java.applet.AppletContext`

Declaration

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

Description

Gets an applet by name.

Returns:

null if the applet does not exist.

getApplets

Applies to

Interface `java.applet.AppletContext`

Declaration

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

Description

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

showDocument

Applies to

Interface java.applet.AppletContext

Declaration

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

Description

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

showStatus

Applies to

Interface `java.applet.AppletContext`

Declaration

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

Description

Show a status string.

Interface `java.applet.AppletStub`

Declaration

`public interface AppletStub extends Object`

Description

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

Methods

`appletResize(int, int)`

`getAppletContext()`

`getCodeBase()`

`getDocumentBase()`

`getParameter(String)`

`isActive()`

isActive

Declaration

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

Description

Returns true if the applet is active.

getDocumentBase

Declaration

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

Description

Gets the document URL.

getCodeBase

Declaration

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

Description

Gets the base URL.

getParameter

Declaration

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

Description

Gets a parameter of the applet.

getAppletContext

Declaration

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

Description

Gets a handler to the applet's context.

appletResize

Declaration

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

Description

Called when the applet wants to be resized.

Interface `java.applet.AudioClip`

Declaration

```
public interface AudioClip  
    extends Object
```

Description

A very high level abstraction of audio.

Methods

[loop\(\)](#)

[play\(\)](#)

[stop\(\)](#)

play

Applies to

Interface java.applet.AudioClip

Declaration

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

Description

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

loop

Applies to

Declaration

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

Description

Starts playing the clip in a loop.

stop

Applies to

Declaration

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

Description

Stops playing the clip.

Class `java.util.BitSet`

`java.lang.Object`

|

+----`java.util.BitSet`

Declaration

`public final class BitSet`

`extends Object implements Cloneable`

Description

A set of bits. The set automatically grows as more bits are needed.

Constructors

[`BitSet\(\)`](#)

[`BitSet\(int\)`](#)

Methods

[`and\(BitSet\)`](#)

[`clear\(int\)`](#)

[`clone\(\)`](#)

[`equals\(Object\)`](#)

[`get\(int\)`](#)

[`hashCode\(\)`](#)

[`or\(BitSet\)`](#)

[`set\(int\)`](#)

[`size\(\)`](#)

[`toString\(\)`](#)

[`xor\(BitSet\)`](#)

BitSet

Applies to

Class `java.util.BitSet`

Declaration

```
public BitSet()  
public BitSet(int nbits)
```

Description

```
public BitSet()
```

Creates an empty set.

```
public BitSet(int nbits)
```

Creates an empty set with the specified size.

Parameters:

nbits - the size of the set

set

Applies to

Class `java.util.BitSet`

Declaration

```
public void set(int bit)
```

Description

Sets a bit.

Parameters:

bit - the bit to be set

clear

Applies to

Class `java.util.BitSet`

Declaration

```
public void clear(int bit)
```

Description

Clears a bit.

Parameters:

bit - the bit to be cleared

get

Applies to

Class `java.util.BitSet`

Declaration

```
public boolean get(int bit)
```

Description

Gets a bit.

Parameters:

bit - the bit to be gotten

and

Applies to

Class `java.util.BitSet`

Declaration

```
public void and(BitSet set)
```

Description

Logically ANDs this bit set with the specified set of bits.

Parameters:

set - the bit set to be ANDed with

or

Applies to

Class `java.util.BitSet`

Declaration

```
public void or(BitSet set)
```

Description

Logically ORs this bit set with the specified set of bits.

Parameters:

set - the bit set to be ORed with

xor

Applies to

Class `java.util.BitSet`

Declaration

```
public void xor(BitSet set)
```

Description

Logically XORs this bit set with the specified set of bits.

Parameters:

set - the bit set to be XORed with

hashCode

Applies to

Class `java.util.BitSet`

Declaration

```
public int hashCode()
```

Description

Gets the hashcode.

Overrides:

hashCode in class `Object`

size

Applies to

Class `java.util.BitSet`

Declaration

```
public int size()
```

Description

Calculates and returns the set's size

equals

Applies to

Class `java.util.BitSet`

Declaration

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

Description

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`

clone

Applies to

Class `java.util.BitSet`

Declaration

```
public Object clone()
```

Description

Clones the BitSet.

Overrides:

clone in class `Object`

toString

Applies to

Class `java.util.BitSet`

Declaration

```
public String toString()
```

Description

Converts the BitSet to a String.

Overrides:

toString in class Object

Class java.lang.Boolean

java.lang.Object

|

+----java.lang.Boolean

Declaration

```
public final class Boolean
    extends Object
```

Description

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.

Variables

FALSE

MAX_VALUE

MIN_VALUE

TRUE

Constructors

Boolean(boolean)

Boolean(String)

Constructors

booleanValue()

equals(Object)

getBoolean(String)

hashCode()

toString()

valueOf(String)

TRUE

Applies to

Class `java.lang.Boolean`

Declaration

```
public final static Boolean TRUE
```

Description

Assigns this Boolean to be true.

FALSE

Applies to

Class `java.lang.Boolean`

Declaration

```
public final static Boolean FALSE
```

Description

Assigns this Boolean to be false.

MIN_VALUE

Applies to

Class `java.lang.Boolean`

Declaration

```
public final static char MIN_VALUE
```

Description

The minimum value a Character can have. The lowest minimum value an Integer can have is .

MAX_VALUE

Applies to

Class `java.lang.Boolean`

Declaration

```
public final static char MAX_VALUE
```

Description

The maximum value a Character can have. The greatest maximum value an Integer can have is .

Boolean

Applies to

Class `java.lang.Boolean`

Declaration

```
public Boolean(boolean value)  
public Boolean(String s)
```

Description

public Boolean(boolean value)

Constructs a Boolean object initialized to the specified boolean value.

Parameters:

value - the value of the 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

booleanValue

Applies to

Class `java.lang.Boolean`

Declaration

```
public boolean booleanValue()
```

Description

Returns the value of this Boolean object as a boolean.

valueOf

Applies to

Class `java.lang.Boolean`

Declaration

```
public static Boolean valueOf(String s)
```

Description

Returns the boolean value represented by the specified String.

Parameters:

s - the String to be parsed

toString

Applies to

Class `java.lang.Boolean`

Declaration

```
public String toString()
```

Description

Returns a new String object representing this Boolean's value.

Overrides:

toString in class Object

hashCode

Applies to

Class `java.lang.Boolean`

Declaration

```
public int hashCode()
```

Description

Returns a hashcode for this Boolean.

Overrides:

hashCode in class `Object`

equals

Applies to

Class `java.lang.Boolean`

Declaration

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

Description

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

Applies to

Class `java.lang.Boolean`

Declaration

```
public static boolean getBoolean(String name)
```

Description

Gets a Boolean from the properties.

Parameters:

name - the property name.

Class java.awt.BorderLayout

java.lang.Object

|

+----java.awt.BorderLayout

Declaration

```
public class BorderLayout
    extends Object
    implements LayoutManager
```

Description

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

.

Constructors

[BorderLayout\(\)](#)

[BorderLayout\(int, int\)](#)

Methods

[addLayoutComponent\(String, Component\)](#)

[layoutContainer\(Container\)](#)

[minimumLayoutSize\(Container\)](#)

[preferredLayoutSize\(Container\)](#)

[removeLayoutComponent\(Component\)](#)

[toString\(\)](#)

BorderLayout

Applies to

Class `java.awt.BorderLayout`

Declaration

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

Description

```
public BorderLayout()
```

Constructs a new BorderLayout.

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

Constructs a BorderLayout with the specified gaps.

Parameters:

hgap - the horizontal gap

vgap - the vertical gap

addLayoutComponent

Applies to

Class java.awt.BorderLayout

Declaration

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

Description

Adds the specified named component to the layout.

Parameters:

name - the String name

comp - the component to be added

removeLayoutComponent

Applies to

Class `java.awt.BorderLayout`

Declaration

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

Description

Removes the specified component from the layout.

Parameters:

comp - the component to be removed

minimumLayoutSize

Applies to

Class `java.awt.BorderLayout`

Declaration

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

Description

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

Applies to

Class `java.awt.BorderLayout`

Declaration

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

Description

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

Applies to

Class `java.awt.BorderLayout`

Declaration

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

Description

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

Applies to

Class `java.awt.BorderLayout`

Declaration

```
public String toString()
```

Description

Returns the String representation of this BorderLayout's values.

Overrides:

toString in class Object

Interface `java.awt.peer.ButtonPeer`

Declaration

```
public interface ButtonPeer
    extends Object
    extends ComponentPeer
```

Methods

[setLabel\(String\)](#)

setLabel

Applies to

Interface `java.awt.peer.ButtonPeer`

Declaration

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

Interface `java.awt.peer.CanvasPeer`

Declaration

```
public interface CanvasPeer
    extends Object
    extends ComponentPeer
```

Class java.awt.CardLayout

java.lang.Object

|

+----java.awt.CardLayout

Declaration

```
public class CardLayout
    extends Object
    implements LayoutManager
```

Description

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.

Constructors

[CardLayout\(\)](#)

[CardLayout\(int, int\)](#)

Methods

[addLayoutComponent\(String, Component\)](#)

[first\(Container\)](#)

[last\(Container\)](#)

[layoutContainer\(Container\)](#)

[minimumLayoutSize\(Container\)](#)

[next\(Container\)](#)

[preferredLayoutSize\(Container\)](#)

[previous\(Container\)](#)

[removeLayoutComponent\(Component\)](#)

[show\(Container, String\)](#)

[toString\(\)](#)

CardLayout

Applies to

Class `java.awt.CardLayout`

Declaration

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

Description

```
public CardLayout()
```

Creates a new card layout.

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

Creates a card layout with the specified gaps.

Parameters:

hgap - the horizontal gap

vgap - the vertical gap

addLayoutComponent

Applies to

Class `java.awt.CardLayout`

Declaration

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

Description

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

Applies to

Class `java.awt.CardLayout`

Declaration

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

Description

Removes the specified component from the layout.

Parameters:

comp - the component to be removed

preferredLayoutSize

Applies to

Class `java.awt.CardLayout`

Declaration

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

Description

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

Applies to

Class `java.awt.CardLayout`

Declaration

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

Description

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

Applies to

Class `java.awt.CardLayout`

Declaration

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

Description

Performs a layout in the specified panel.

Parameters:

parent - the name of the parent container

first

Applies to

Class `java.awt.CardLayout`

Declaration

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

Description

Flip to the first card.

Parameters:

parent - the name of the parent container

next

Applies to

Class `java.awt.CardLayout`

Declaration

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

Description

Flips to the next card of the specified container.

Parameters:

parent - the name of the container

[previous](#)

Applies to

Class `java.awt.CardLayout`

Declaration

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

Description

Flips to the previous card of the specified container.

Parameters:

parent - the name of the parent container

last

Applies to

Class `java.awt.CardLayout`

Declaration

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

Description

Flips to the last card of the specified container.

Parameters:

parent - the name of the parent container

show

Applies to

Class `java.awt.CardLayout`

Declaration

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

Description

Flips to the specified component name in the specified container.

Parameters:

parent - the name of the parent container

name - the component name

toString

Applies to

Class `java.awt.CardLayout`

Declaration

```
public String toString()
```

Description

Returns the String representation of this CardLayout's values.

Overrides:

toString in class Object

Class java.lang.Character

java.lang.Object

|

+----java.lang.Character

Declaration

```
public final class Character
    extends Object
```

Description

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.

Variables

MAX_RADIX

MIN_RADIX

Constructors

Character(char)

Methods

charValue()

digit(char, int)

equals(Object)

forDigit(int, int)

hashCode()

isDigit(char)

isLowerCase(char)

isSpace(char)

isUpperCase(char)

toLowerCase(char)

toString()

toUpperCase(char)

MIN_RADIX

Applies to

Class `java.lang.Character`

Declaration

```
public final static int MIN_RADIX
```

Description

The minimum radix available for conversion to and from Strings. The lowest minimum value that a radix can be is 2.

See Also:

`toString`

MAX_RADIX

Applies to

Class `java.lang.Character`

Declaration

```
public final static int MAX_RADIX
```

Description

The maximum radix available for conversion to and from Strings. The largest maximum value that a radix can have is 36.

See Also:

`toString`

Character

Applies to

Class `java.lang.Character`

Declaration

```
public Character(char value)
```

Description

Constructs a Character object with the specified value.

Parameters:

value - value of this Character object

isLowerCase

Applies to

Class `java.lang.Character`

Declaration

```
public static boolean isLowerCase(char ch)
```

Description

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.

isUpperCase

Applies to

Class `java.lang.Character`

Declaration

```
public static boolean isUpperCase(char ch)
```

Description

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.

isDigit

Applies to

Class java.lang.Character

Declaration

```
public static boolean isDigit(char ch)
```

Description

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.

isSpace

Applies to

Class `java.lang.Character`

Declaration

```
public static boolean isSpace(char ch)
```

Description

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.

toLowerCase

Applies to

Class `java.lang.Character`

Declaration

```
public static char toLowerCase(char ch)
```

Description

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

toUpperCase

Applies to

Class java.lang.Character

Declaration

```
public static char toUpperCase(char ch)
```

Description

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

digit

Applies to

Class `java.lang.Character`

Declaration

```
public static int digit(char ch,  
                        int radix)
```

Description

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

forDigit

Applies to

Class `java.lang.Character`

Declaration

```
public static char forDigit(int digit,  
                           int radix)
```

Description

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

charValue

Applies to

Class `java.lang.Character`

Declaration

```
public char charValue()
```

Description

Returns the value of this Character object.

hashCode

Applies to

Class java.lang.Character

Declaration

```
public int hashCode()
```

Description

Returns a hashcode for this Character.

Overrides:

hashCode in class Object

equals

Applies to

Class java.lang.Character

Declaration

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

Description

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

Applies to

Class java.lang.Character

Declaration

```
public String toString()
```

Description

Returns a String object representing this character's value.

Overrides:

toString in class Object

Class java.awt.CheckboxGroup

java.lang.Object

|

+----java.awt.CheckboxGroup

Declaration

```
public class CheckboxGroup
    extends Object
```

Description

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.

Constructors

[CheckboxGroup\(\)](#)

Methods

[getCurrent\(\)](#)

[setCurrent\(Checkbox\)](#)

[toString\(\)](#)

CheckboxGroup

Applies to

Class `java.awt.CheckboxGroup`

Declaration

```
public CheckboxGroup()
```

Description

Creates a new `CheckboxGroup`.

getCurrent

Applies to

Class `java.awt.CheckboxGroup`

Declaration

```
public Checkbox getCurrent()
```

Description

Gets the current choice.

setCurrent

Applies to

Class `java.awt.CheckboxGroup`

Declaration

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

Description

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

Applies to

Class `java.awt.CheckboxGroup`

Declaration

```
public String toString()
```

Description

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

Overrides:

toString in class Object

Interface `java.awt.peer.CheckboxMenuItemPeer`

Declaration

```
public interface CheckboxMenuItemPeer
    extends Object
    extends MenuItemPeer
```

Methods

`setState(boolean)`

setState

Applies to

java.awt.peer.CheckboxMenuItemPeer

Declaration

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

Interface `java.awt.peer.CheckboxPeer`

public interface CheckboxPeer

extends Object

extends ComponentPeer

Methods

[setCheckboxGroup\(CheckboxGroup\)](#)

[setLabel\(String\)](#)

[setState\(boolean\)](#)

setState

Applies to

Interface `java.awt.peer.CheckboxPeer`

Declaration

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

setCheckboxGroup

Applies to

Interface `java.awt.peer.CheckboxPeer`

Declaration

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

setLabel

Applies to

Interface `java.awt.peer.CheckboxPeer`

Declaration

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

Interface `java.awt.peer.ChoicePeer`

public interface ChoicePeer

extends Object

extends ComponentPeer

Methods

[addItem\(String, int\)](#)

[select\(int\)](#)

addItem

Applies to

Interface java.awt.peer.ChoicePeer

Declaration

```
public abstract void addItem(String item,  
                             int index)
```


select

Applies to

Interface `java.awt.peer.ChoicePeer`

Declaration

```
public abstract void select(int index)
```

Class java.lang.Class

java.lang.Object

|

+----java.lang.Class

Declaration

```
public final class Class
    extends Object
```

Description

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\(String\)](#)

[getClassLoader\(\)](#)

[getInterfaces\(\)](#)

[getName\(\)](#)

[getSuperclass\(\)](#)

[isInterface\(\)](#)

[newInstance\(\)](#)

[toString\(\)](#)

forName

Applies to

Class java.lang.Class

Declaration

```
public static Class forName(String className) throws ClassNotFoundException
```

Description

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.

newInstance

Applies to

Class java.lang.Class

Declaration

```
public Object newInstance() throws InstantiationException, IllegalAccessException
```

Description

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

Applies to

Class java.lang.Class

Declaration

```
public String getName()
```

Description

Returns the name of this Class.

getSuperclass

Applies to

Class java.lang.Class

Declaration

```
public Class getSuperclass()
```

Description

Returns the superclass of this Class.

getInterfaces

Applies to

Class java.lang.Class

Declaration

```
public Class[] getInterfaces()
```

Description

Returns the interfaces of this Class. An array of length 0 is returned if this Class implements no interfaces.

getClassLoader

Applies to

Class java.lang.Class

Declaration

```
public ClassLoader getClassLoader()
```

Description

Returns the Class loader of this Class. Returns null if this Class does not have a Class loader.

See Also:

ClassLoader

isInterface

Applies to

Class `java.lang.Class`

Declaration

```
public boolean isInterface()
```

Description

Returns a boolean indicating whether or not this Class is an interface.

toString

Applies to

Class java.lang.Class

Declaration

```
public String toString()
```

Description

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.ClassLoader

java.lang.Object

|

+----java.lang.ClassLoader

Declaration

```
public class ClassLoader
    extends Object
```

Description

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;
}
}
```

Constructors

ClassLoader()

Methods

defineClass(byte[], int, int)

findSystemClass(String)

loadClass(String, boolean)

resolveClass(Class)

ClassLoader

Applies to

Class java.lang.ClassLoader

Declaration

```
protected ClassLoader()
```

Description

Constructs a new Class loader and initializes it.

loadClass

Applies to

Class java.lang.ClassLoader

Declaration

```
protected abstract Class loadClass(String name,  
                                   boolean resolve) throws ClassNotFoundException
```

Description

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

defineClass

Applies to

Class java.lang.ClassLoader

Declaration

```
protected final Class defineClass(byte data[],  
                                int offset,  
                                int length)
```

Description

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

resolveClass

Applies to

Class `java.lang.ClassLoader`

Declaration

```
protected final void resolveClass(Class c)
```

Description

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

Applies to

Class `java.lang.ClassLoader`

Declaration

protected final Class findSystemClass(String name) throws ClassNotFoundException

Description

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

Interface `java.lang.Cloneable`

```
public interface Cloneable  
extends Object
```

An interface indicating that this object may be copied or cloned.

Class java.awt.Color

java.lang.Object

|

+----java.awt.Color

Declaration

```
public final class Color
```

```
    extends Object
```

Description

A class to encapsulate RGB Colors.

Variables

black

blue

cyan

darkGray

gray

green

lightGray

magenta

orange

pink

red

white

yellow

Constructors

Color(int, int, int)

Color(int)

Color(float, float, float)

Methods

HSBtoRGB(float, float, float)

RGBtoHSB(int, int, int, float[])

brighter()

darker()

equals(Object)

getBlue()

getColor(String)

getColor(String, Color)

getColor(String, int)

getGreen()

getHSBColor(float, float, float)

getRGB()

getRed()

hashCode()

toString()

Color Variables

Applies to

Class `java.awt.Color`

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

Applies to

Class `java.awt.Color`

Declaration

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

Description

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`

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

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

Applies to

Class `java.awt.Color`

Declaration

```
public int getRed()
```

Description

Gets the red component.

See Also:

getRGB

getGreen

Applies to

Class `java.awt.Color`

Declaration

```
public int getGreen()
```

Description

Gets the green component.

See Also:

`getRGB`

getBlue

Applies to

Class `java.awt.Color`

Declaration

```
public int getBlue()
```

Description

Gets the blue component.

See Also:

`getRGB`

getRGB

Applies to

Class `java.awt.Color`

Declaration

```
public int getRGB()
```

Description

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

Applies to

Class `java.awt.Color`

Declaration

```
public Color brighter()
```

Description

Returns a brighter version of this color.

darker

Applies to

Class `java.awt.Color`

Declaration

```
public Color darker()
```

Description

Returns a darker version of this color.

hashCode

Applies to

Class `java.awt.Color`

Declaration

```
public int hashCode()
```

Description

Computes the hash code.

Overrides:

hashCode in class `Object`

equals

Applies to

Class `java.awt.Color`

Declaration

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

Description

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

Applies to

Class java.awt.Color

Declaration

```
public String toString()
```

Description

Returns the String representation of this Color's values.

Overrides:

toString in class Object

getColor

Applies to

Class `java.awt.Color`

Declaration

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

Description

public static Color getColor(String nm)

Gets the specified Color property.

Parameters:

nm - the name of the color property

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.

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

Applies to

Class `java.awt.Color`

Declaration

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

Description

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

Applies to

Class `java.awt.Color`

Declaration

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

Description

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

Applies to

Class `java.awt.Color`

Declaration

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

Description

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.image.ColorModel

java.lang.Object

|

+----java.awt.image.ColorModel

Declaration

```
public class ColorModel
    extends Object
```

Description

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

Variables

pixel_bits

Constructors

ColorModel(int)

Methods

getAlpha(int)
getBlue(int)
getGreen(int)
getPixelSize()
getRGB(int)
getRGBdefault()
getRed(int)

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 component 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 component 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 component 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 component for the specified pixel.

Returns:

The alpha transparency value ranging from 0 to 255

getRGB

```
public int getRGB(int pixel)
```

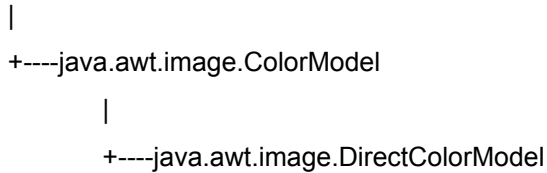
Returns the color of the pixel in the default RGB color model.

See Also:

getRGBdefault

Class java.awt.image.DirectColorModel

java.lang.Object



Declaration

```
public class DirectColorModel
    extends ColorModel
```

Description

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 reasons, but you cannot override or modify the behaviour of those methods.

See Also:

ColorModel

Constructors

```
DirectColorModel(int, int, int, int)
DirectColorModel(int, int, int, int, int)
```

Methods

```
getAlpha(int)
getAlphaMask()
```

getBlue(int)
getBlueMask()
getGreen(int)
getGreenMask()
getRGB(int)
getRed(int)
getRedMask()

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.

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 alpha, 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 component 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 component 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 component 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:

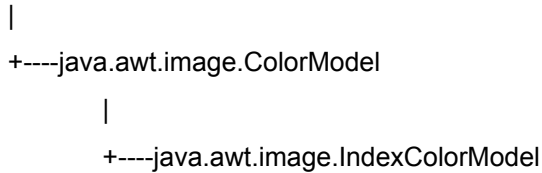
getRGB in class `ColorModel`

See Also:

getRGBdefault

Class java.awt.image.IndexColorModel

java.lang.Object



Declaration

```
public class IndexColorModel
    extends ColorModel
```

Description

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 reasons, but you cannot override or modify the behaviour of those methods.

See Also:

ColorModel

Constructors

```
IndexColorModel(int, int, byte[], byte[], byte[])
IndexColorModel (int, int, byte[], byte[], byte[], int)
IndexColorModel (int, int, byte[], byte[], byte[], byte[])
IndexColorModel(int, int, byte[], int, boolean)
IndexColorModel(int, int, byte[], int, boolean, int)
```

Methods

getAlpha(int)
getAlphas(byte[])
getBlue(int)
getBlues(byte[])
getGreen(int)
getGreens(byte[])
getMapSize()
getRGB(int)
getRed(int)
getReds(byte[])
getTransparentPixel()

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.

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.

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.

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.

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.

getRed

```
public final int getRed(int pixel)
```

Returns the red color component 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 component 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 component 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.lang.Compiler

java.lang.Object

|

+----java.lang.Compiler

Declaration

```
public final class Compiler
    extends Object
```

Methods

```
command(Object)
compileClass(Class)
compileClasses(String)
disable()
enable()
```

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.awt.Component

java.lang.Object

|

+----java.awt.Component

Declaration

```
public class Component
    extends Object
    implements ImageObserver
```

Description

A generic Abstract Window Toolkit component.

Methods

- [action\(Event, Object\)](#)
- [addNotify\(\)](#)
- [bounds\(\)](#)
- [checkImage\(Image, ImageObserver\)](#)
- [checkImage\(Image, int, int, ImageObserver\)](#)
- [createImage\(ImageProducer\)](#)
- [createImage\(int, int\)](#)
- [deliverEvent\(Event\)](#)
- [disable\(\)](#)
- [enable\(\)](#)
- [enable\(boolean\)](#)
- [getBackground\(\)](#)
- [getColorModel\(\)](#)
- [getFont\(\)](#)
- [getFontMetrics\(Font\)](#)
- [getForeground\(\)](#)
- [getGraphics\(\)](#)

getParent()
getPeer()
getToolkit()
gotFocus(Event, Object)
handleEvent(Event)
hide()
imageUpdate(Image, int, int, int, int, int)
inside(int, int)
invalidate()
isEnabled()
isShowing()
isValid()
isVisible()
keyDown(Event, int)
keyUp(Event, int)
layout()
list()
list(PrintStream)
list(PrintStream, int)
locate(int, int)
location()
lostFocus(Event, Object)
minimumSize()
mouseDown(Event, int, int)
mouseDrag(Event, int, int)
mouseEnter(Event, int, int)
mouseExit(Event, int, int)
mouseMove(Event, int, int)
mouseUp(Event, int, int)
move(int, int)
nextFocus()
paint(Graphics)
paintAll(Graphics)
paramString()
postEvent(Event)
preferredSize()

prepareImage(Image, ImageObserver)
prepareImage(Image, int, int, ImageObserver)
print(Graphics)
printAll(Graphics)
removeNotify()
repaint()
repaint(long)
repaint(int, int, int, int)
repaint(long, int, int, int, int)
requestFocus()
reshape(int, int, int, int)
resize(int, int)
resize(Dimension)
setBackground(Color)
setFont(Font)
setForeground(Color)
show()
show(boolean)
size()
toString()
update(Graphics)
validate()

getParent

Applies to

Class `java.awt.Component`

Declaration

```
public Container getParent()
```

Description

Gets the parent of the component.

getPeer

Applies to

Class `java.awt.Component`

Declaration

```
public ComponentPeer getPeer()
```

Description

Gets the peer of the component.

getToolkit

Applies to

Class `java.awt.Component`

Declaration

```
public Toolkit getToolkit()
```

Description

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

Applies to

Class `java.awt.Component`

Declaration

```
public boolean isValid()
```

Description

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

See Also:

validate, invalidate

isVisible

Applies to

Class `java.awt.Component`

Declaration

```
public boolean isVisible()
```

Description

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

Applies to

Class `java.awt.Component`

Declaration

```
public boolean isShowing()
```

Description

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

Applies to

Class `java.awt.Component`

Declaration

```
public boolean isEnabled()
```

Description

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

See Also:

enable, disable

location

Applies to

Class `java.awt.Component`

Declaration

```
public Point location()
```

Description

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

See Also:

`move`

size

Applies to

Class `java.awt.Component`

Declaration

```
public Dimension size()
```

Description

Returns the current size of this component.

See Also:

resize

bounds

Applies to

Class `java.awt.Component`

Declaration

```
public Rectangle bounds()
```

Description

Returns the current bounds of this component.

See Also:

reshape

enable

Applies to

Class `java.awt.Component`

Declaration

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

Description

public synchronized void enable()

Enables a component.

See Also:

`isEnabled`, `disable`

public void enable(boolean cond)

Conditionally enables a component.

Parameters:

`cond` - if true, enables component; disables otherwise.

See Also:

`enable`, `disable`

disable

Applies to

Class `java.awt.Component`

Declaration

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

Description

Disables a component.

See Also:

`isEnabled`, `enable`

show

Applies to

Class `java.awt.Component`

Declaration

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

Description

public synchronized void show()

Shows the component.

See Also:

`isVisible`, `hide`

public void show(boolean cond)

Conditionally shows the component.

Parameters:

`cond` - if true, it shows the component; hides otherwise.

See Also:

`show`, `hide`

hide

Applies to

Class `java.awt.Component`

Declaration

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

Description

Hides the component.

See Also:

`isVisible`, `hide`

getForeground

Applies to

Class `java.awt.Component`

Declaration

```
public Color getForeground()
```

Description

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

See Also:

`setForeground`

setForeground

Applies to

Class `java.awt.Component`

Declaration

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

Description

Sets the foreground color.

Parameters:

c - the Color

See Also:

getForeground

getBackground

Applies to

Class `java.awt.Component`

Declaration

```
public Color getBackground()
```

Description

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

Applies to

Class `java.awt.Component`

Declaration

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

Description

Sets the background color.

Parameters:

c - the Color

See Also:

getBackground

getFont

Applies to

Class `java.awt.Component`

Declaration

```
public Font getFont()
```

Description

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

Applies to

Class `java.awt.Component`

Declaration

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

Description

Sets the font of the component.

Parameters:

f - the font

See Also:

getFont

getColorModel

Applies to

Class `java.awt.Component`

Declaration

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

Description

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

See Also:

ColorModel

move

Applies to

Class `java.awt.Component`

Declaration

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

Description

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

Applies to

Class `java.awt.Component`

Declaration

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

Description

```
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

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

Resizes the Component to the specified dimension.

Parameters:

d - the component dimension

See Also:

size, reshape

reshape

Applies to

Class `java.awt.Component`

Declaration

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

Description

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

Applies to

Class `java.awt.Component`

Declaration

```
public Dimension preferredSize()
```

Description

Returns the preferred size of this component.

See Also:

`minimumSize`, `LayoutManager`

minimumSize

Applies to

Class `java.awt.Component`

Declaration

```
public Dimension minimumSize()
```

Description

Returns the minimum size of this component.

See Also:

`preferredSize`, `LayoutManager`

layout

Applies to

Class `java.awt.Component`

Declaration

```
public void layout()
```

Description

Lays out the component. This is usually called when the component is validated.

See Also:

validate, LayoutManager

validate

Applies to

Class `java.awt.Component`

Declaration

```
public void validate()
```

Description

Validates a component.

See Also:

invalidate, layout, LayoutManager

invalidate

Applies to

Class `java.awt.Component`

Declaration

```
public void invalidate()
```

Description

Invalidates a component.

See Also:

validate, layout, LayoutManager

getGraphics

Applies to

Class `java.awt.Component`

Declaration

```
public Graphics getGraphics()
```

Description

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

Applies to

Class `java.awt.Component`

Declaration

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

Description

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

Applies to

Class `java.awt.Component`

Declaration

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

Description

Paints the component.

Parameters:

g - the specified Graphics window

See Also:

update

update

Applies to

Class `java.awt.Component`

Declaration

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

Description

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

Applies to

Class `java.awt.Component`

Declaration

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

Description

Paints the component and its subcomponents.

Parameters:

g - the specified Graphics window

See Also:

paint

repaint

Applies to

Class `java.awt.Component`

Declaration

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

Description

public void repaint()

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

See Also:

paint

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

```
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

```
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 within tm milliseconds.

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

Applies to

Class `java.awt.Component`

Declaration

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

Description

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

Parameters:

`g` - the specified Graphics window

See Also:

`paint`

printAll

Applies to

Class `java.awt.Component`

Declaration

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

Description

Prints the component and its subcomponents.

Parameters:

g - the specified Graphics window

See Also:

print

imageUpdate

Applies to

Class `java.awt.Component`

Declaration

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

Description

Repaints the component when the image has changed.

Returns:

true if image has changed; false otherwise.

createImage

Applies to

Class `java.awt.Component`

Declaration

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

Description

public Image createImage(ImageProducer producer)

Creates an image from the specified image producer.

Parameters:

producer - the image producer

**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

prepareImage

Applies to

Class `java.awt.Component`

Declaration

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

Description

```
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

```
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

checkImage

Applies to

Class `java.awt.Component`

Declaration

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

Description

```
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`

```
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`

inside

Applies to

Class `java.awt.Component`

Declaration

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

Description

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

Applies to

Class `java.awt.Component`

Declaration

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

Description

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

Parameters:

x - the x coordinate

y - the y coordinate

See Also:

`inside`

deliverEvent

Applies to

Class `java.awt.Component`

Declaration

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

Description

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

Parameters:

e - the event

See Also:

handleEvent, postEvent

postEvent

Applies to

Class `java.awt.Component`

Declaration

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

Description

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

Applies to

Class `java.awt.Component`

Declaration

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

Description

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

Applies to

Class `java.awt.Component`

Declaration

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

Description

Called if the mouse is down.

Parameters:

evt - the event

x - the x coordinate

y - the y coordinate

See Also:

`handleEvent`

mouseDrag

Applies to

Class `java.awt.Component`

Declaration

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

Description

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

Applies to

Class `java.awt.Component`

Declaration

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

Description

Called if the mouse is up.

Parameters:

evt - the event

x - the x coordinate

y - the y coordinate

See Also:

`handleEvent`

mouseMove

Applies to

Class `java.awt.Component`

Declaration

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

Description

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

Applies to

Class `java.awt.Component`

Declaration

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

Description

Called when the mouse enters the component.

Parameters:

evt - the event

x - the x coordinate

y - the y coordinate

See Also:

`handleEvent`

mouseExit

Applies to

Class `java.awt.Component`

Declaration

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

Description

Called when the mouse exits the component.

Parameters:

evt - the event

x - the x coordinate

y - the y coordinate

See Also:

`handleEvent`

keyDown

Applies to

Class `java.awt.Component`

Declaration

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

Description

Called if a character is pressed.

Parameters:

evt - the event

key - the key that's pressed

See Also:

`handleEvent`

keyUp

Applies to

Class `java.awt.Component`

Declaration

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

Description

Called if a character is released.

Parameters:

evt - the event

key - the key that's released

See Also:

`handleEvent`

action

Applies to

Class `java.awt.Component`

Declaration

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

Description

Called if an action occurs in the Component.

Parameters:

evt - the event

what - the action that's occurring

See Also:

`handleEvent`

addNotify

Applies to

Class `java.awt.Component`

Declaration

```
public void addNotify()
```

Description

Notifies the Component to create a peer.

See Also:

`getPeer`, `removeNotify`

removeNotify

Applies to

Class `java.awt.Component`

Declaration

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

Description

Notifies the Component to destroy the peer.

See Also:

`getPeer`, `addNotify`

gotFocus

Applies to

Class `java.awt.Component`

Declaration

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

Description

Indicates that this component has received the input focus.

See Also:

`requestFocus`, `lostFocus`

lostFocus

Applies to

Class `java.awt.Component`

Declaration

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

Description

Indicates that this component has lost the input focus.

See Also:

`requestFocus`, `gotFocus`

requestFocus

Applies to

Class `java.awt.Component`

Declaration

```
public void requestFocus()
```

Description

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

See Also:

`gotFocus`

nextFocus

Applies to

Class `java.awt.Component`

Declaration

```
public void nextFocus()
```

Description

Moves the focus to the next component.

See Also:

`requestFocus`, `gotFocus`

paramString

Applies to

Class `java.awt.Component`

Declaration

```
protected String paramString()
```

Description

Returns the parameter String of this Component.

toString

Applies to

Class `java.awt.Component`

Declaration

```
public String toString()
```

Description

Returns the String representation of this Component's values.

Overrides:

toString in class Object

list

Applies to

Class `java.awt.Component`

Declaration

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

Description

public void list()

Prints a listing to a print stream.

public void list(PrintStream out)

Prints a listing to the specified print out stream.

Parameters:

out - the Stream name

public void list(PrintStream out, int indent)

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

Parameters:

out - the Stream name

indent - the start of the list



Class `java.awt.Button`

`java.lang.Object`

|

+----`java.awt.Component`

|

+----`java.awt.Button`

Declaration

```
public class Button
    extends Component
```

Description

A class that produces a labeled button component.

Constructors

[Button\(\)](#)

[Button\(String\)](#)

Methods

[addNotify\(\)](#)

[getLabel\(\)](#)

[paramString\(\)](#)

[setLabel\(String\)](#)

Button

Applies to

Class `java.awt.Button`

Declaration

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

Description

public Button()

Constructs a Button with no label.

public Button(String label)

Constructs a Button with a string label.

Parameters:

label - the button label

addNotify

Applies to

Class java.awt.Button

Declaration

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

Description

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

getLabel

Applies to

Class `java.awt.Button`

Declaration

```
public String getLabel()
```

Description

Gets the label of the button.

See Also:

setLabel

setLabel

Applies to

Class `java.awt.Button`

Declaration

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

Description

Sets the button with the specified label.

Parameters:

label - the label to set the button with

See Also:

getLabel

paramString

Applies to

Class `java.awt.Button`

Declaration

```
protected String paramString()
```

Description

Returns the parameter String of this button.

Overrides:

paramString in class Component

Class java.awt.Canvas

java.lang.Object

|

+----java.awt.Component

|

+----java.awt.Canvas

Declaration

```
public class Canvas
    extends Component
```

Description

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

Constructors

[Canvas\(\)](#)

Methods

[addNotify\(\)](#)

[paint\(Graphics\)](#)

Canvas

Applies to

Class `java.awt.Canvas`

Declaration

```
public Canvas()
```

addNotify

Applies to

Class java.awt.Canvas

Declaration

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

Description

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

Applies to

Class java.awt.Canvas

Declaration

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

Description

Paints the canvas in the default background color.

Parameters:

g - the specified Graphics window

Overrides:

paint in class Component



Class `java.awt.Checkbox`

`java.lang.Object`

|

+----`java.awt.Component`

|

+----`java.awt.Checkbox`

Declaration

```
public class Checkbox
    extends Component
```

Description

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

Constructors

[Checkbox\(\)](#)

[Checkbox\(String\)](#)

[Checkbox\(String, CheckboxGroup, boolean\)](#)

Methods

[addNotify\(\)](#)

[getCheckboxGroup\(\)](#)

[getLabel\(\)](#)

[getState\(\)](#)

[paramString\(\)](#)

[setCheckboxGroup\(CheckboxGroup\)](#)

[setLabel\(String\)](#)

[setState\(boolean\)](#)

Checkbox

Applies to

Class `java.awt.Checkbox`

Declaration

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

Description

public Checkbox()

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

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

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

Applies to

Class java.awt.Checkbox

Declaration

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

Description

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

Applies to

Class `java.awt.Checkbox`

Declaration

```
public String getLabel()
```

Description

Gets the label of the button.

See Also:

setLabel

setLabel

Applies to

Class `java.awt.Checkbox`

Declaration

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

Description

Sets the button with the specified label.

Parameters:

label - the label of the button

See Also:

getLabel

getState

Applies to

Class `java.awt.Checkbox`

Declaration

```
public boolean getState()
```

Description

Returns the boolean state of the Checkbox.

See Also:

`setState`

setState

Applies to

Class `java.awt.Checkbox`

Declaration

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

Description

Sets the Checkbox to the specified boolean state.

Parameters:

state - the boolean state

See Also:

getState

getCheckboxGroup

Applies to

Class `java.awt.Checkbox`

Declaration

```
public CheckboxGroup getCheckboxGroup()
```

Description

Returns the checkbox group.

See Also:

`setCheckboxGroup`

setCheckboxGroup

Applies to

Class `java.awt.Checkbox`

Declaration

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

Description

Sets the CheckboxGroup to the specified group.

Parameters:

g - the new CheckboxGroup

See Also:

getCheckboxGroup

paramString

Applies to

Class `java.awt.Checkbox`

Declaration

```
protected String paramString()
```

Description

Returns the parameter String of this Checkbox.

Overrides:

paramString in class Component



Class `java.awt.Choice`

`java.lang.Object`

|

+----`java.awt.Component`

|

+----`java.awt.Choice`

Declaration

```
public class Choice
    extends Component
```

Description

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

Constructors

[Choice\(\)](#)

Methods

[addItem\(String\)](#)

[addNotify\(\)](#)

[countItems\(\)](#)

[getItem\(int\)](#)

[getSelectedIndex\(\)](#)

[getSelectedItem\(\)](#)

[paramString\(\)](#)

[select\(int\)](#)

[select\(String\)](#)

Choice

Applies to

Class `java.awt.Choice`

Declaration

```
public Choice()
```

Description

Constructs a new Choice.

addNotify

Applies to

Class java.awt.Choice

Declaration

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

Description

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

Applies to

Class `java.awt.Choice`

Declaration

```
public int countItems()
```

Description

Returns the number of items in this Choice.

See Also:

`getItem`

getItem

Applies to

Class java.awt.Choice

Declaration

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

Description

Returns the String at the specified index in the Choice.

Parameters:

index - the index at which to begin

See Also:

countItems

addItem

Applies to

Class `java.awt.Choice`

Declaration

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

Description

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

Applies to

Class `java.awt.Choice`

Declaration

```
public String getItem()
```

Description

Returns a String representation of the current choice.

See Also:

`getSelectedItem`

getSelectedIndex

Applies to

Class `java.awt.Choice`

Declaration

```
public int getSelectedIndex()
```

Description

Returns the index of the currently selected item.

See Also:

`getSelectedItem`

select

Applies to

Class `java.awt.Choice`

Declaration

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

Description

public synchronized void select(int pos)

Selects the item with the specified position.

Parameters:

pos - the choice item position

Throws: `IllegalArgumentException`

If the choice item position is invalid.

See Also:

`getSelectedItem`, `getSelectedIndex`

public void select(String str)

Selects the item with the specified String.

Parameters:

str - the specified String

See Also:

`getSelectedItem`, `getSelectedIndex`

paramString

Applies to

Class `java.awt.Choice`

Declaration

```
protected String paramString()
```

Description

Returns the parameter String of this Choice.

Overrides:

paramString in class Component

Class java.awt.Container

java.lang.Object

|

+----java.awt.Component

|

+----java.awt.Container

Declaration

```
public class Container
    extends Component
```

Description

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

Methods

[add\(Component\)](#)

[add\(Component, int\)](#)

[add\(String, Component\)](#)

[addNotify\(\)](#)

[countComponents\(\)](#)

[deliverEvent\(Event\)](#)

[getComponent\(int\)](#)

[getComponents\(\)](#)

[getLayout\(\)](#)07

[insets\(\)](#)

[layout\(\)](#)

[list\(PrintStream, int\)](#)

[locate\(int, int\)](#)

[minimumSize\(\)](#)

[paintComponents\(Graphics\)](#)

paramString()

preferredSize()

printComponents(Graphics)

remove(Component)

removeAll()

removeNotify()

setLayout(LayoutManager)

validate()

countComponents

Applies to

Class `java.awt.Container`

Declaration

```
public int countComponents()
```

Description

Returns the number of components in this panel.

See Also:

`getComponent`

getComponent

Applies to

Class `java.awt.Container`

Declaration

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

Description

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

Applies to

Class `java.awt.Container`

Declaration

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

Description

Gets all the components in this container.

Insets

Applies to

Class `java.awt.Container`

Declaration

```
public Insets insets()
```

Description

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

Applies to

Class java.awt.Container

Declaration

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

Description

public Component add(Component comp)

Adds the specified component to this container.

Parameters:

comp - the component to be added

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

```
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

Applies to

Class `java.awt.Container`

Declaration

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

Description

Removes the specified component from this container.

Parameters:

comp - the component to be removed

See Also:

add

removeAll

Applies to

Class `java.awt.Container`

Declaration

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

Description

Removes all the components from this container.

See Also:

`add`, `remove`

getLayout

Applies to

Class `java.awt.Container`

Declaration

```
public LayoutManager getLayout()
```

Description

Gets the layout manager for this container.

See Also:

layout, setLayout

setLayout

Applies to

Class java.awt.Container

Declaration

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

Description

Sets the layout manager for this container.

Parameters:

mgr - the specified layout manager

See Also:

layout, getLayout

layout

Applies to

Class `java.awt.Container`

Declaration

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

Description

Does a layout on this Container.

Overrides:

layout in class Component

See Also:

setLayout

validate

Applies to

Class `java.awt.Container`

Declaration

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

Description

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

Overrides:

validate in class Component

See Also:

validate, invalidate

preferredSize

Applies to

Class `java.awt.Container`

Declaration

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

Description

Returns the preferred size of this container.

Overrides:

preferredSize in class Component

See Also:

minimumSize

minimumSize

Applies to

Class `java.awt.Container`

Declaration

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

Description

Returns the minimum size of this container.

Overrides:

`minimumSize` in class `Component`

See Also:

`preferredSize`

paintComponents

Applies to

Class `java.awt.Container`

Declaration

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

Description

Paints the components in this container.

Parameters:

g - the specified Graphics window

See Also:

paint, paintAll

printComponents

Applies to

Class `java.awt.Container`

Declaration

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

Description

Prints the components in this container.

Parameters:

g - the specified Graphics window

See Also:

print, printAll

deliverEvent

Applies to

Class java.awt.Container

Declaration

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

Description

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

Applies to

Class `java.awt.Container`

Declaration

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

Description

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

Applies to

Class java.awt.Container

Declaration

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

Description

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

Applies to

Class `java.awt.Container`

Declaration

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

Description

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

Applies to

Class java.awt.Container

Declaration

```
protected String paramString()
```

Description

Returns the parameter String of this Container.

Overrides:

paramString in class Component

list

Applies to

Class `java.awt.Container`

Declaration

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

Description

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.Panel`

`java.lang.Object`

|

+----`java.awt.Component`

|

+----`java.awt.Container`

|

+----`java.awt.Panel`

Declaration

```
public class Panel
    extends Container
```

Description

A Panel Container class. This produces a generic container.

Constructors

[Panel\(\)](#)

Methods

[addNotify\(\)](#)

Panel

Applies to

Class `java.awt.Panel`

Declaration

```
public Panel()
```

Description

Creates a new panel. The default layout for all panels is `FlowLayout`.

addNotify

Applies to

Class `java.awt.Panel`

Declaration

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

Description

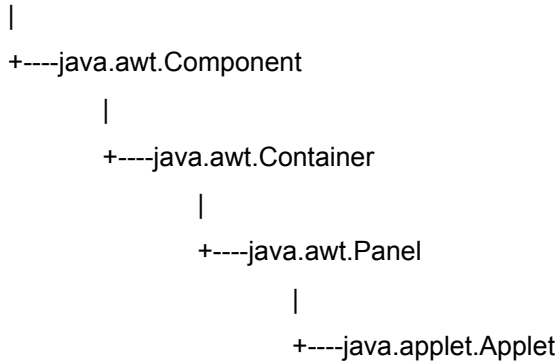
Creates the Panel's peer. The peer allows you to modify the appearance of the panel without changing its functionality.

Overrides:

addNotify in class `Container`

Class java.applet.Applet

java.lang.Object



Declaration

```
public class Applet
    extends Panel
```

Description

Base applet class.

Constructors

[Applet\(\)](#)

Methods

[destroy\(\)](#)

[getAppletContext\(\)](#)

[getAppletInfo\(\)](#)

[getAudioClip\(URL\)](#)

[getAudioClip\(URL, String\)](#)

[getCodeBase\(\)](#)

[getDocumentBase\(\)](#)

[getImage\(URL\)](#)

[getImage\(URL, String\)](#)

getParameter(String)

getParameterInfo()

init()

isActive()

play(URL)

play(URL, String)

resize(int, int)

resize(Dimension)

setStub(AppletStub)

showStatus(String)

start()

stop()

Applet

Applies to

Class java.applet.Applet

Declaration

```
public Applet()
```

setStub

Applies to

Class `java.applet.Applet`

Declaration

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

Description

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

isActive

Applies to

Class `java.applet.Applet`

Declaration

```
public boolean isActive()
```

Description

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

See Also:

start

getDocumentBase

Applies to

Class `java.applet.Applet`

Declaration

```
public URL getDocumentBase()
```

Description

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

See Also:

`getCodeBase`

getCodeBase

Applies to

Class `java.applet.Applet`

Declaration

```
public URL getCodeBase()
```

Description

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

See Also:

`getDocumentBase`

getParameter

Applies to

Class `java.applet.Applet`

Declaration

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

Description

Gets a parameter of the applet.

getAppletContext

Applies to

Class `java.applet.Applet`

Declaration

```
public AppletContext getAppletContext()
```

Description

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

Applies to

Class java.applet.Applet

Declaration

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

Description

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

Requests that the applet be resized.

Overrides:

resize in class Component

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

Requests that the applet be resized.

Overrides:

resize in class Component

showStatus

Applies to

Class `java.applet.Applet`

Declaration

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

Description

Shows a status message in the applet's context.

getImage

Applies to

Class `java.applet.Applet`

Declaration

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

Description

public Image getImage(URL url)

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

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

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

See Also:

getImage

getAudioClip

Applies to

Class `java.applet.Applet`

Declaration

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

Description

public AudioClip getAudioClip(URL url)

Gets an audio clip.

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

Gets an audio clip.

See Also:

getAudioClip

getAppletInfo

Applies to

Class `java.applet.Applet`

Declaration

```
public String getAppletInfo()
```

Description

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

getParameterInfo

Applies to

Class `java.applet.Applet`

Declaration

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

Description

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

Applies to

Class `java.applet.Applet`

Declaration

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

Description

public void play(URL url)

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

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

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

init

Applies to

Class `java.applet.Applet`

Declaration

```
public void init()
```

Description

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

Applies to

Class `java.applet.Applet`

Declaration

```
public void start()
```

Description

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

Applies to

Class `java.applet.Applet`

Declaration

```
public void stop()
```

Description

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

Applies to

Class `java.applet.Applet`

Declaration

```
public void destroy()
```

Description

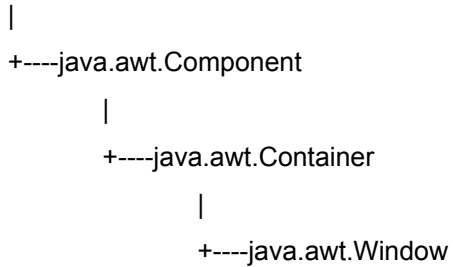
Cleans up whatever resources are being held. If the applet is active it is stopped stopped.

See Also:

init, start, stop

Class java.awt.Window

java.lang.Object



Declaration

```
public class Window
    extends Container
```

Description

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.

Constructors

[Window\(Frame\)](#)

Methods

[addNotify\(\)](#)

[dispose\(\)](#)

[getToolkit\(\)](#)

[getWarningString\(\)](#)

[pack\(\)](#)

[show\(\)](#)

[toBack\(\)](#)

[toFront\(\)](#)

Window

Applies to

Class `java.awt.Window`

Declaration

```
public Window(Frame parent)
```

Description

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

Applies to

Class java.awt.Window

Declaration

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

Description

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

Applies to

Class `java.awt.Window`

Declaration

```
public synchronized void pack()
```

Description

Packs the components of the Window.

show

Applies to

Class `java.awt.Window`

Declaration

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

Description

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

Applies to

Class `java.awt.Window`

Declaration

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

Description

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

toFront

Applies to

Class `java.awt.Window`

Declaration

```
public void toFront()
```

Description

Brings the frame to the front of the Window.

[toBack](#)

Applies to

Class `java.awt.Window`

Declaration

```
public void toBack()
```

Description

Sends the frame to the back of the Window.

getToolkit

Applies to

Class `java.awt.Window`

Declaration

```
public Toolkit getToolkit()
```

Description

Returns the toolkit of this frame.

Overrides:

getToolkit in class `Component`

See Also:

Toolkit

getWarningString

Applies to

Class `java.awt.Window`

Declaration

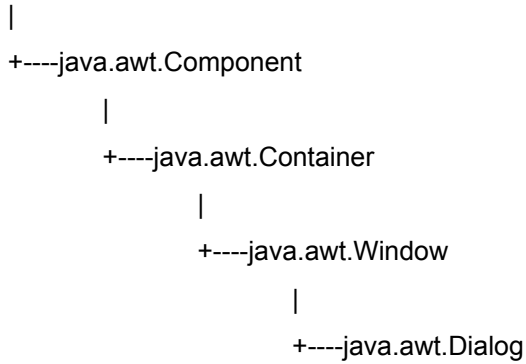
```
public final String getWarningString()
```

Description

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.awt.Dialog

java.lang.Object



Declaration

```
public class Dialog
    extends Window
```

Description

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

Constructors

[Dialog\(Frame, boolean\)](#)

[Dialog\(Frame, String, boolean\)](#)

Methods

[addNotify\(\)](#)

[getTitle\(\)](#)

[isModal\(\)](#)

[isResizable\(\)](#)

[paramString\(\)](#)

[setResizable\(boolean\)](#)

[setTitle\(String\)](#)

Dialog

Applies to

Class `java.awt.Dialog`

Declaration

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

Description

```
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

```
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

addNotify

Applies to

Class java.awt.Dialog

Declaration

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

Description

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

isModal

Applies to

Class `java.awt.Dialog`

Declaration

```
public boolean isModal()
```

Description

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

getTitle

Applies to

Class `java.awt.Dialog`

Declaration

```
public String getTitle()
```

Description

Gets the title of the Dialog.

See Also:

`setTitle`

setTitle

Applies to

Class `java.awt.Dialog`

Declaration

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

Description

Sets the title of the Dialog.

Parameters:

title - the new title being given to the Dialog

See Also:

getTitle

isResizable

Applies to

Class `java.awt.Dialog`

Declaration

```
public boolean isResizable()
```

Description

Returns true if the user can resize the frame.

setResizable

Applies to

Class java.awt.Dialog

Declaration

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

Description

Sets the resizable flag.

Parameters:

resizable - true if resizable; false otherwise

paramString

Applies to

Class java.awt.Dialog

Declaration

protected String paramString()

Description

Returns the parameter String of this Dialog.

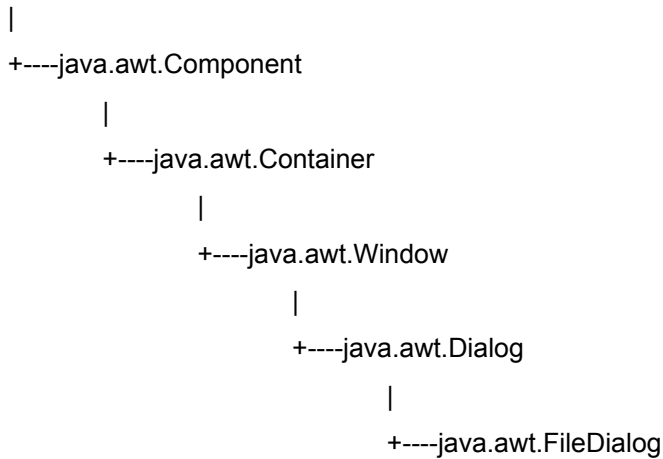
Overrides:

paramString in class Container



Class java.awt.FileDialog

java.lang.Object



Declaration

```
public class FileDialog
    extends Dialog
```

Description

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

Variables

LOAD

SAVE

Constructors

FileDialog(Frame, String)

FileDialog(Frame, String, int)

Methods

addNotify()

getDirectory()

getFile()

getFilenameFilter()

getMode()

paramString()

setDirectory(String)

setFile(String)

setFilenameFilter(FilenameFilter)

LOAD

Applies to

Class `java.awt.FileDialog`

Declaration

```
public final static int LOAD
```

Description

The file load variable.

SAVE

Applies to

Class `java.awt.FileDialog`

Declaration

```
public final static int SAVE
```

Description

The file save variable.

FileDialog

Applies to

Class java.awt.FileDialog

Declaration

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

Description

```
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

```
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

Applies to

Class java.awt.FileDialog

Declaration

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

Description

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

Applies to

Class `java.awt.FileDialog`

Declaration

```
public int getMode()
```

Description

Gets the mode of the file dialog.

getDirectory

Applies to

Class `java.awt.FileDialog`

Declaration

```
public String getDirectory()
```

Description

Gets the directory of the Dialog.

setDirectory

Applies to

Class java.awt.FileDialog

Declaration

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

Description

Set the directory of the Dialog to the specified directory.

Parameters:

dir - the specific directory

getFile

Applies to

Class `java.awt.FileDialog`

Declaration

```
public String getFile()
```

Description

Gets the file of the Dialog.

setFile

Applies to

Class `java.awt.FileDialog`

Declaration

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

Description

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

Applies to

Class `java.awt.FileDialog`

Declaration

```
public FilenameFilter getFilenameFilter()
```

Description

Gets the filter.

setFilenameFilter

Applies to

Class `java.awt.FileDialog`

Declaration

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

Description

Sets the filter for this dialog to the specified filter.

Parameters:

filter - the specified filter

paramString

Applies to

Class java.awt.FileDialog

Declaration

protected String paramString()

Description

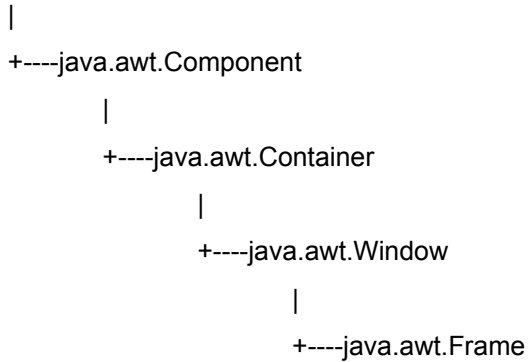
Returns the parameter String of this file dialog. Parameter String.

Overrides:

paramString in class Dialog

Class java.awt.Frame

java.lang.Object



Declaration

```
public class Frame
```

```
extends Window
```

```
implements MenuContainer
```

Description

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

Variables

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

Constructors

Frame()

Frame(String)

Methods

addNotify()

dispose()

getCursorType()

getIconImage()

getMenuBar()

getTitle()

isResizable()

paramString()

remove(MenuComponent)

setCursor(int)

setIconImage(Image)

setMenuBar(MenuBar)

setResizable(boolean)

setTitle(String)

Frame Variables

Applies to

DEFAULT_CURSOR

public final static int DEFAULT_CURSOR

CROSSHAIR_CURSOR

public final static int CROSSHAIR_CURSOR

TEXT_CURSOR

public final static int TEXT_CURSOR

WAIT_CURSOR

public final static int WAIT_CURSOR

SW_RESIZE_CURSOR

public final static int SW_RESIZE_CURSOR

SE_RESIZE_CURSOR

public final static int SE_RESIZE_CURSOR

NW_RESIZE_CURSOR

public final static int NW_RESIZE_CURSOR

NE_RESIZE_CURSOR

public final static int NE_RESIZE_CURSOR

N_RESIZE_CURSOR

public final static int N_RESIZE_CURSOR

S_RESIZE_CURSOR

public final static int S_RESIZE_CURSOR

W_RESIZE_CURSOR

```
public final static int W_RESIZE_CURSOR
```

E_RESIZE_CURSOR

```
public final static int E_RESIZE_CURSOR
```

HAND_CURSOR

```
public final static int HAND_CURSOR
```

MOVE_CURSOR

```
public final static int MOVE_CURSOR
```

Frame

Applies to

Class `java.awt.Frame`

Declaration

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

Description

public Frame()

Constructs a new Frame that is initially invisible.

See Also:

resize, show

public Frame(String title)

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

Parameters:

title - the specified title

See Also:

resize, show

addNotify

Applies to

Class java.awt.Frame

Declaration

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

Description

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

Applies to

Class `java.awt.Frame`

Declaration

```
public String getTitle()
```

Description

Gets the title of the Frame.

See Also:

`setTitle`

setTitle

Applies to

Class java.awt.Frame

Declaration

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

Description

Sets the title for this Frame to the specified title.

Parameters:

title - the specified title of this Frame

See Also:

getTitle

getImage

Applies to

Class java.awt.Frame

Declaration

```
public Image getImage()
```

Description

Returns the icon image for this Frame.

setIconImage

Applies to

Class java.awt.Frame

Declaration

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

Description

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

getMenuBar

Applies to

Class `java.awt.Frame`

Declaration

```
public MenuBar getMenuBar()
```

Description

Gets the menu bar for this Frame.

setMenuBar

Applies to

Class java.awt.Frame

Declaration

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

Description

Sets the menubar for this Frame to the specified menubar.

Parameters:

mb - the menubar being set

remove

Applies to

Class `java.awt.Frame`

Declaration

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

Description

Removes the specified menu bar from this Frame.

dispose

Applies to

Class `java.awt.Frame`

Declaration

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

Description

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

Overrides:

dispose in class Window

isResizable

Applies to

Class java.awt.Frame

Declaration

```
public boolean isResizable()
```

Description

Returns true if the user can resize the Frame.

setResizable

Applies to

Class java.awt.Frame

Declaration

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

Description

Sets the resizable flag.

Parameters:

resizable - true if resizable; false otherwise.

setCursor

Applies to

Class java.awt.Frame

Declaration

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

Description

Set the cursor image to a predefined cursor.

Parameters:

cursorType - one of the cursor constants defined above.

getCursorType

Applies to

Class java.awt.Frame

Declaration

```
public int getCursorType()
```

Description

Return the cursor type

paramString

Applies to

Class java.awt.Frame

Declaration

protected String paramString()

Description

Returns the parameter String of this Frame.

Overrides:

paramString in class Container

Class `java.awt.Label`

`java.lang.Object`

|

+----`java.awt.Component`

|

+----`java.awt.Label`

Declaration

```
public class Label
    extends Component
```

Description

A component that displays a single line of read-only text.

Variables

`CENTER`

`LEFT`

`RIGHT`

Constructors

`Label()`

`Label(String)`

`Label(String, int)`

Methods

`addNotify()`

`getAlignment()`

`getText()`

`paramString()`

`setAlignment(int)`

`setText(String)`

LEFT

Applies to

Class `java.awt.Label`

Declaration

```
public final static int LEFT
```

Description

The left alignment.

CENTER

Applies to

Class `java.awt.Label`

Declaration

```
public final static int CENTER
```

Description

The center alignment.

RIGHT

Applies to

Class `java.awt.Label`

Declaration

```
public final static int RIGHT
```

Description

The right alignment.

Label

Applies to

Class `java.awt.Label`

Declaration

```
public Label()  
public Label(String label)  
public Label(String label,  
              int alignment)
```

Description

public Label()

Constructs an empty label.

public Label(String label)

Constructs a new label with the specified String of text.

Parameters:

label - the text that makes up the 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

addNotify

Applies to

Class java.awt.Label

Declaration

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

Description

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

Applies to

Class `java.awt.Label`

Declaration

```
public int getAlignment()
```

Description

Gets the current alignment of this label.

See Also:

`setAlignment`

setAlignment

Applies to

Class `java.awt.Label`

Declaration

```
public void setAlignment(int alignment)
```

Description

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

Applies to

Class `java.awt.Label`

Declaration

```
public String getText()
```

Description

Gets the text of this label.

See Also:

setText

setText

Applies to

Class `java.awt.Label`

Declaration

```
public void setText(String label)
```

Description

Sets the text for this label to the specified text.

Parameters:

label - the text that makes up the label

See Also:

getText

paramString

Applies to

Class `java.awt.Label`

Declaration

```
protected String paramString()
```

Description

Returns the parameter String of this label.

Overrides:

paramString in class Component



Class java.awt.List

java.lang.Object

|

+----java.awt.Component

|

+----java.awt.List

Declaration

```
public class List
    extends Component
```

Description

A scrolling list of text items.

Constructors

List()

List(int, boolean)

Methods

addItem(String)

addItem(String, int)

addNotify()

allowsMultipleSelections()

clear()

countItems()

delItem(int)

delItems(int, int)

deselect(int)

getItem(int)

getRows()

getSelectedIndex()

getSelectedIndexes()
getSelectedItem()
getSelectedItems()
getVisibleIndex()
isSelected(int)
makeVisible(int)
minimumSize(int)
minimumSize()
paramString()
preferredSize(int)
preferredSize()
removeNotify()
replaceItem(String, int)
select(int)
setMultipleSelections(boolean)

List

Applies to

Class `java.awt.List`

Declaration

```
public List()  
public List(int rows,  
            boolean multipleSelections)
```

Description

public List()

Creates a new scrolling list initialized with no visible Lines or multiple selections.

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

Applies to

Class java.awt.List

Declaration

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

Description

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

Applies to

Class `java.awt.List`

Declaration

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

Description

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

Applies to

Class `java.awt.List`

Declaration

```
public int countItems()
```

Description

Returns the number of items in the list.

See Also:

`getItem`

getItem

Applies to

Class java.awt.List

Declaration

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

Description

Gets the item associated with the specified index.

Parameters:

index - the position of the item

See Also:

countItems

addItem

Applies to

Class java.awt.List

Declaration

```
public synchronized void addItem(String item)
public synchronized void addItem(String item,
                                int index)
```

Description

public synchronized void addItem(String item)

Adds the specified item to the end of scrolling list.

Parameters:

item - the item to be added

**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.

replaceAllItem

Applies to

Class java.awt.List

Declaration

```
public synchronized void replaceItem(String newValue,  
                                     int index)
```

Description

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

Applies to

Class `java.awt.List`

Declaration

```
public synchronized void clear()
```

Description

Clears the list.

See Also:

`delItem`, `delItems`

delItem

Applies to

Class java.awt.List

Declaration

```
public synchronized void delItem(int position)
```

Description

Delete an item from the list.

delItems

Applies to

Class `java.awt.List`

Declaration

```
public synchronized void delItems(int start,  
                                   int end)
```

Description

Delete multiple items from the list.

getSelectedIndex

Applies to

Class `java.awt.List`

Declaration

```
public synchronized int getSelectedIndex()
```

Description

Get the selected item on the list or -1 if no item is selected.

See Also:

select, deselect, isSelected

getSelectedIndexes

Applies to

Class `java.awt.List`

Declaration

```
public synchronized int[] getSelectedIndexes()
```

Description

Returns the selected indexes on the list.

See Also:

`select`, `deselect`, `isSelected`

getSelectedItem

Applies to

Class `java.awt.List`

Declaration

```
public synchronized String getItem()
```

Description

Returns the selected item on the list or null if no item is selected.

See Also:

select, deselect, isSelected

getSelectedItems

Applies to

Class java.awt.List

Declaration

```
public synchronized String[] getSelectedItems()
```

Description

Returns the selected items on the list.

See Also:

select, deselect, isSelected

select

Applies to

Class `java.awt.List`

Declaration

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

Description

Selects the item at the specified index.

Parameters:

index - the position of the item to select

See Also:

`getSelectedItem`, `deselect`, `isSelected`

deselect

Applies to

Class `java.awt.List`

Declaration

```
public synchronized void deselect(int index)
```

Description

Deselects the item at the specified index.

Parameters:

index - the position of the item to deselect

See Also:

select, `getSelectedItem`, `isSelected`

isSelected

Applies to

Class java.awt.List

Declaration

```
public synchronized boolean isSelected(int index)
```

Description

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

Applies to

Class `java.awt.List`

Declaration

```
public int getRows()
```

Description

Returns the number of visible lines in this list.

allowsMultipleSelections

Applies to

Class `java.awt.List`

Declaration

```
public boolean allowsMultipleSelections()
```

Description

Returns true if this list allows multiple selections.

See Also:

`setMultipleSelections`

setMultipleSelections

Applies to

Class `java.awt.List`

Declaration

```
public void setMultipleSelections(boolean v)
```

Description

Sets whether this list should allow multiple selections or not.

Parameters:

v - the boolean to allow multiple selections

See Also:

`allowsMultipleSelections`

setVisibleIndex

Applies to

Class java.awt.List

Declaration

```
public int setVisibleIndex()
```

Description

Gets the index of the item that was last made visible by the method `setVisible`.

makeVisible

Applies to

Class `java.awt.List`

Declaration

```
public void makeVisible(int index)
```

Description

Forces the item at the specified index to be visible.

Parameters:

index - the position of the item

See Also:

`getVisibleIndex`

preferredSize

Applies to

Class `java.awt.List`

Declaration

```
public Dimension preferredSize(int rows)
public Dimension preferredSize()
```

Description

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.

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

Applies to

Class `java.awt.List`

Declaration

```
public Dimension minimumSize(int rows)
public Dimension minimumSize()
```

Description

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

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

Applies to

Class `java.awt.List`

Declaration

```
protected String paramString()
```

Description

Returns the parameter String of this list.

Overrides:

paramString in class Component

Class `java.awt.Scrollbar`

`java.lang.Object`

|

+----`java.awt.Component`

|

+----`java.awt.Scrollbar`

Declaration

```
public class Scrollbar
    extends Component
```

Description

A Scrollbar component.

Variables

HORIZONTAL

VERTICAL

Constructors

Scrollbar()

Scrollbar(int)

Scrollbar(int, int, int, int, int)

Methods

addNotify()

getLineIncrement()

getMaximum()

getMinimum()

getOrientation()

getPageIncrement()

getValue()

setVisible()

paramString()

setLineIncrement(int)

setPageIncrement(int)

setValue(int)

setValues(int, int, int, int)

HORIZONTAL

Applies to

Class `java.awt.Scrollbar`

Declaration

```
public final static int HORIZONTAL
```

Description

The horizontal Scrollbar variable.

VERTICAL

Applies to

Class `java.awt.Scrollbar`

Declaration

```
public final static int VERTICAL
```

Description

The vertical Scrollbar variable.

Scrollbar

Applies to

Class `java.awt.Scrollbar`

Declaration

```
public Scrollbar()  
public Scrollbar(int orientation)  
public Scrollbar(int orientation,  
                  int value,  
                  int visible,  
                  int minimum,  
                  int maximum)
```

Description

public Scrollbar()

Constructs a new vertical 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.

public Scrollbar(int orientation,

```
    int value,  
    int visible,
```


**int minimum,
int maximum)**

Constructs a new Scrollbar with the specified orientation, value, page size, and minimum 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

addNotify

Applies to

Class java.awt.Scrollbar

Declaration

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

Description

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

getOrientation

Applies to

Class `java.awt.Scrollbar`

Declaration

```
public int getOrientation()
```

Description

Returns the orientation for this Scrollbar.

getValue

Applies to

Class `java.awt.Scrollbar`

Declaration

```
public int getValue()
```

Description

Returns the current value of this Scrollbar.

See Also:

getMinimum, getMaximum

setValue

Applies to

Class `java.awt.Scrollbar`

Declaration

```
public void setValue(int value)
```

Description

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

Applies to

Class `java.awt.Scrollbar`

Declaration

```
public int getMinimum()
```

Description

Returns the minimum value of this Scrollbar.

See Also:

`getMaximum`, `getValue`

getMaximum

Applies to

Class `java.awt.Scrollbar`

Declaration

```
public int getMaximum()
```

Description

Returns the maximum value of this Scrollbar.

See Also:

`getMinimum`, `getValue`

getVisible

Applies to

Class `java.awt.Scrollbar`

Declaration

```
public int getVisible()
```

Description

Returns the visible amount of the Scrollbar.

setLineIncrement

Applies to

Class `java.awt.Scrollbar`

Declaration

```
public void setLineIncrement(int l)
```

Description

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

Applies to

Class `java.awt.Scrollbar`

Declaration

```
public int getLineIncrement()
```

Description

Gets the line increment for this scrollbar.

setPageIncrement

Applies to

Class `java.awt.Scrollbar`

Declaration

```
public void setPageIncrement(int l)
```

Description

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

Applies to

Class `java.awt.Scrollbar`

Declaration

```
public int getPageIncrement()
```

Description

Gets the page increment for this scrollbar.

setValues

Applies to

Class java.awt.Scrollbar

Declaration

```
public void setValues(int value,  
                      int visible,  
                      int minimum,  
                      int maximum)
```

Description

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

Applies to

Class `java.awt.Scrollbar`

Declaration

```
protected String paramString()
```

Description

Returns the String parameters for this Scrollbar.

Overrides:

paramString in class Component

Class java.awt.TextComponent

java.lang.Object

|

+----java.awt.Component

|

+----java.awt.TextComponent

Declaration

```
public class TextComponent
    extends Component
```

Description

A TextComponent is a component that allows the editing of some text.

Methods

[getSelectedText\(\)](#)

[getSelectionEnd\(\)](#)

[getSelectionStart\(\)](#)

[getText\(\)](#)

[isEditable\(\)](#)

[paramString\(\)](#)

[removeNotify\(\)](#)

[select\(int, int\)](#)

[selectAll\(\)](#)

[setEditable\(boolean\)](#)

[setText\(String\)](#)

removeNotify

Applies to

Class `java.awt.TextComponent`

Declaration

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

Description

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

Applies to

Class `java.awt.TextComponent`

Declaration

```
public void setText(String t)
```

Description

Sets the text of this `TextComponent` to the specified text.

Parameters:

t - the new text to be set

See Also:

`getText`

getText

Applies to

Class `java.awt.TextComponent`

Declaration

```
public String getText()
```

Description

Returns the text contained in this `TextComponent`.

See Also:

`setText`

getSelectedText

Applies to

Class `java.awt.TextComponent`

Declaration

```
public String getSelectedText()
```

Description

Returns the selected text contained in this `TextComponent`.

See Also:

`setText`

isEditable

Applies to

Class `java.awt.TextComponent`

Declaration

```
public boolean isEditable()
```

Description

Returns the boolean indicating whether this `TextComponent` is editable or not.

See Also:

`setEditable`

setEditable

Applies to

Class java.awt.TextComponent

Declaration

```
public void setEditable(boolean t)
```

Description

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

Applies to

Class `java.awt.TextComponent`

Declaration

```
public int getSelectionStart()
```

Description

Returns the selected text's start position.

getSelectionEnd

Applies to

Class `java.awt.TextComponent`

Declaration

```
public int getSelectionEnd()
```

Description

Returns the selected text's end position.

select

Applies to

Class `java.awt.TextComponent`

Declaration

```
public void select(int selStart,  
                  int selEnd)
```

Description

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

Applies to

Class `java.awt.TextComponent`

Declaration

```
public void selectAll()
```

Description

Selects all the text in the `TextComponent`.

paramString

Applies to

Class `java.awt.TextComponent`

Declaration

```
protected String paramString()
```

Description

Returns the String of parameters for this TextComponent.

Overrides:

paramString in class Component



Class java.awt.TextArea

java.lang.Object

|

+----java.awt.Component

|

+----java.awt.TextComponent

|

+----java.awt.TextArea

Declaration

```
public class TextArea
```

```
    extends TextComponent
```

Description

A TextArea object is a multi-line area that displays text. It can be set to allow editing or read-only modes.

Constructors

[TextArea\(\)](#)

[TextArea\(int, int\)](#)

[TextArea\(String\)](#)

[TextArea\(String, int, int\)](#)

Methods

[addNotify\(\)](#)

[appendText\(String\)](#)

[getColumns\(\)](#)

[getRows\(\)](#)

[insertText\(String, int\)](#)

[minimumSize\(int, int\)](#)

[minimumSize\(\)](#)

[paramString\(\)](#)

preferredSize(int, int)

preferredSize()

replaceText(String, int, int)

TextArea

Applies to

Class `java.awt.TextArea`

Declaration

```
public TextArea()  
public TextArea(int rows,  
                int cols)  
public TextArea(String text)  
public TextArea(String text,  
                int rows,  
                int cols)
```

Description

public TextArea()

Constructs a new 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

public TextArea(String text)

Constructs a new TextArea with the specified text displayed.

Parameters:

text - the text to be displayed

```
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

Applies to

Class java.awt.TextArea

Declaration

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

Description

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

insertText

Applies to

Class `java.awt.TextArea`

Declaration

```
public void insertText(String str,  
                        int pos)
```

Description

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

appendText

Applies to

Class `java.awt.TextArea`

Declaration

```
public void appendText(String str)
```

Description

Appends the given text to the end.

Parameters:

str - the text to insert

See Also:

`insertText`

replaceText

Applies to

Class `java.awt.TextArea`

Declaration

```
public void replaceText(String str,  
                        int start,  
                        int end)
```

Description

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

getRows

Applies to

Class `java.awt.TextArea`

Declaration

```
public int getRows()
```

Description

Returns the number of rows in the `TextArea`.

getColumns

Applies to

Class `java.awt.TextArea`

Declaration

```
public int getColumns()
```

Description

Returns the number of columns in the `TextArea`.

preferredSize

Applies to

Class java.awt.TextArea

Declaration

```
public Dimension preferredSize(int rows,  
                               int cols)  
public Dimension preferredSize()
```

Description

```
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

```
public Dimension preferredSize()
```

Returns the preferred size Dimensions of the TextArea.

Overrides:

preferredSize in class Component

minimumSize

Applies to

Class java.awt.TextArea

Declaration

```
public Dimension minimumSize(int rows,  
                               int cols)  
public Dimension minimumSize()
```

Description

```
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

```
public Dimension minimumSize()
```

Returns the minimum size Dimensions of the TextArea.

Overrides:

minimumSize in class Component

paramString

Applies to

Class `java.awt.TextArea`

Declaration

```
protected String paramString()
```

Description

Returns the String of parameters for this TextArea.

Overrides:

paramString in class `TextComponent`



Class java.awt.TextField

java.lang.Object

|

+----java.awt.Component

|

+----java.awt.TextComponent

|

+----java.awt.TextField

Declaration

```
public class TextField
```

```
    extends TextComponent
```

Description

TextField is a component that allows the editing of a single line of text.

Constructors

[TextField\(\)](#)

[TextField\(int\)](#)

[TextField\(String\)](#)

[TextField\(String, int\)](#)

Methods

[addNotify\(\)](#)

[echoCharIsSet\(\)](#)

[getColumns\(\)](#)

[getEchoChar\(\)](#)

[minimumSize\(int\)](#)

[minimumSize\(\)](#)

[paramString\(\)](#)

[preferredSize\(int\)](#)

preferredSize()

setEchoCharacter(char)

TextField

Applies to

Class `java.awt.TextField`

Declaration

```
public TextField()  
public TextField(int cols)  
public TextField(String text)  
public TextField(String text,  
                  int cols)
```

Description

public TextField()

Constructs a new TextField.

public TextField(int cols)

Constructs a new TextField initialized with the specified columns.

Parameters:

cols - the number of columns

public TextField(String text)

Constructs a new TextField initialized with the specified text.

Parameters:

text - the text to be displayed

```
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

Applies to

Class java.awt.TextField

Declaration

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

Description

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

Applies to

Class `java.awt.TextField`

Declaration

```
public char getEchoChar()
```

Description

Returns the character to be used for echoing.

See Also:

`setEchoCharacter`, `echoCharIsSet`

echoCharIsSet

Applies to

Class java.awt.TextField

Declaration

```
public boolean echoCharIsSet()
```

Description

Returns true if this TextField has a character set for echoing.

See Also:

setEchoCharacter, getEchoChar

getColumns

Applies to

Class `java.awt.TextField`

Declaration

```
public int getColumns()
```

Description

Returns the number of columns in this TextField.

setEchoCharacter

Applies to

Class `java.awt.TextField`

Declaration

```
public void setEchoCharacter(char c)
```

Description

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`

preferredSize

Applies to

Class `java.awt.TextField`

Declaration

```
public Dimension preferredSize(int cols)
public Dimension preferredSize()
```

Description

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

public Dimension preferredSize()

Returns

the preferred size Dimensions needed for this TextField.

Overrides:

preferredSize in class Component

minimumSize

Applies to

Class `java.awt.TextField`

Declaration

```
public Dimension minimumSize(int cols)
public Dimension minimumSize()
```

Description

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

public Dimension minimumSize()

Returns the minimum size Dimensions needed for this TextField.

Overrides:

minimumSize in class Component

paramString

Applies to

Class `java.awt.TextField`

Declaration

```
protected String paramString()
```

Description

Returns the String of parameters for this TextField.

Overrides:

paramString in class `TextComponent`

Interface `java.awt.peer.ComponentPeer`

Declaration

```
public interface ComponentPeer
    extends Object
```

Methods

```
checkImage(Image, int, int, ImageObserver)
createImage(ImageProducer)
createImage(int, int)
disable()
dispose()
enable()
getColorModel()
getFontMetrics(Font)
getGraphics()
getToolkit()
handleEvent(Event)
hide()
minimumSize()
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()
```

show

```
public abstract void show()
```

hide

```
public abstract void hide()
```

enable

```
public abstract void enable()
```

disable

```
public abstract void disable()
```

paint

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

repaint

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

print

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

reshape

```
public abstract void reshape(int x,  
                             int y,
```

```
int width,  
int height)
```

handleEvent

```
public abstract boolean handleEvent(Event e)
```

minimumSize

```
public abstract Dimension minimumSize()
```

preferredSize

```
public abstract Dimension preferredSize()
```

getColorModel

```
public abstract ColorModel getColorModel()
```

getToolkit

```
public abstract Toolkit getToolkit()
```

getGraphics

```
public abstract Graphics getGraphics()
```

getFontMetrics

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

dispose

```
public abstract void dispose()
```

setForeground

public abstract void setForeground(Color c)

setBackground

public abstract void setBackground(Color c)

setFont

public abstract void setFont(Font f)

requestFocus

public abstract void requestFocus()

nextFocus

public abstract void nextFocus()

createImage

public abstract Image createImage(ImageProducer producer)

createImage

public abstract Image createImage(int width,
int height)

prepareImage

public abstract boolean prepareImage(Image img,
int w,
int h,
ImageObserver o)

checkImage

```
public abstract int checkImage(Image img,  
                                int w,  
                                int h,  
                                ImageObserver o)
```


Class java.lang.Runtime

java.lang.Object

|

+----java.lang.Runtime

Declaration

```
public class Runtime
```

```
    extends Object
```

Methods

```
exec(String)
```

```
exec(String, String[])
```

```
exec(String[])
```

```
exec(String[], String[])
```

```
exit(int)
```

```
freeMemory()
```

```
gc()
```

```
getLocalizedInputStream(InputStream)
```

```
getLocalizedOutputStream(OutputStream)
```

```
getRuntime()
```

```
load(String)
```

```
loadLibrary(String)
```

```
runFinalization()
```

```
totalMemory()
```

```
traceInstructions(boolean)
```

```
traceMethodCalls(boolean)
```

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 obtaining 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 obtaining 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 obtaining 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 obtaining 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 automatically 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.

Interface `java.awt.peer.ContainerPeer`

Declaration

```
public interface ContainerPeer
    extends Object
    extends ComponentPeer
```

Methods

```
insets()
```


insets

public abstract Insets insets()

Class java.net.ContentHandler

java.lang.Object

|

+----java.net.ContentHandler

Declaration

```
public class ContentHandler
    extends Object
```

Description

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()

Constructors

ContentHandler()

Methods

getContent(URLConnection)

ContentHandler

```
public ContentHandler()
```

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`

Declaration

```
public interface ContentHandlerFactory  
    extends Object
```

Description

This interface defines a factory for `ContentHandler` instances. It is used by the `URLStreamHandler` class to create `ContentHandlers` for various streams.

Methods

```
createContentHandler(String)
```

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.

Interface `java.io.DataInput`

Declaration

```
public interface DataInput
    extends Object
```

Description

`DataInput` is an interface describing streams that can read input in a machine-independent format.

See Also:

`DataInputStream`, `DataOutput`

Methods

```
readBoolean()
readByte()
readChar()
readDouble()
readFloat()
readFully(byte[])
readFully(byte[], int, int)
readInt()
readLine()
readLong()
readShort()
readUTF()
readUnsignedShort()
skipBytes(int)
```

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

Interface `java.io.DataOutput`

Declaration

```
public interface DataOutput
    extends Object
```

Description

`DataOutput` is an interface describing streams that can write output in a machine-independent format.

See Also:

`DataOutputStream`, `DataInput`

Methods

```
write(int)
write(byte[])
write(byte[], int, int)
writeBoolean(boolean)
writeByte(int)
writeBytes(String)
writeChar(int)
writeChars(String)
writeDouble(double)
writeFloat(float)
writeInt(int)
writeLong(long)
writeShort(int)
writeUTF(String)
```

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.net.DatagramPacket

java.lang.Object

|

+----java.net.DatagramPacket

Declaration

```
public final class DatagramPacket
    extends Object
```

Description

A class that represents a datagram packet containing packet data, packet length, internet addresses and port.

Constructors

```
DatagramPacket(byte[], int)
DatagramPacket(byte[], int, InetAddress, int)
```

Methods

```
getAddress()
getData()
getLength()
getPort()
```

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.

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.

getAddress

```
public InetAddress getAddress()
```


getPort

```
public int getPort()
```

getData

```
public byte[] getData()
```

getLength

```
public int getLength()
```

Class java.net.DatagramSocket

java.lang.Object

|

+----java.net.DatagramSocket

Declaration

```
public class DatagramSocket
    extends Object
```

Description

The datagram socket class implements unreliable datagrams.

Constructors

```
DatagramSocket()
DatagramSocket(int)
```

Methods

```
close()
finalize()
getLocalPort()
receive(DatagramPacket)
send(DatagramPacket)
```

DatagramSocket

```
public DatagramSocket() throws SocketException
```

Creates a datagram socket

DatagramSocket

```
public DatagramSocket(int port) throws SocketException
```

Creates a datagram socket

Parameters:

local - port to use

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

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.util.Date

java.lang.Object

|

+----java.util.Date

Declaration

```
public class Date
    extends Object
```

Description

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.

Constructors

Date()
Date(long)
Date(int, int, int)
Date(int, int, int, int, int)
Date(int, int, int, int, int, int)
Date(String)

Methods

UTC(int, int, int, int, int, int)
after(Date)
before(Date)
equals(Object)
getDate()
getDay()
getHours()
getMinutes()
getMonth()
getSeconds()
getTime()
getTimezoneOffset()
getYear()
hashCode()
parse(String)
setDate(int)
setHours(int)
setMinutes(int)
setMonth(int)
setSeconds(int)
setTime(long)
setYear(int)
toGMTString()

toLocaleString()

toString()

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().

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

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, it 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.

getYear

```
public int getYear()
```

Returns the year after 1900.

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

getTime

```
public long getTime()
```

Returns the time in milliseconds since the epoch.

setTime

```
public void setTime(long time)
```

Sets the time.

Parameters:

time - The new time value in milliseconds since the epoch.

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.

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.

Interface `sun.tools.debug.DebuggerCallback`

Declaration

```
public interface DebuggerCallback  
    extends Object
```

Description

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.

Methods

```
breakpointEvent(RemoteThread)  
exceptionEvent(RemoteThread, String)  
printToConsole(String)  
quitEvent()  
threadDeathEvent(RemoteThread)
```

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().

Interface `java.awt.peer.DialogPeer`

Declaration

```
public interface DialogPeer
    extends Object
    extends WindowPeer
```

Methods

```
setResizable(boolean)
setTitle(String)
```

setTitle

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

setResizable

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

Class java.util.Dictionary

java.lang.Object

|

+----java.util.Dictionary

Declaration

```
public class Dictionary
    extends Object
```

Description

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

Constructors

Dictionary()

Methods

elements()

get(Object)

isEmpty()

keys()

put(Object, Object)

remove(Object)

size()

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:

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:

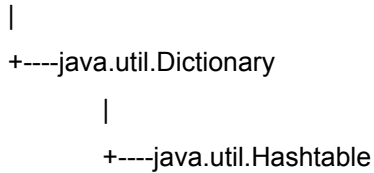
key - the key that needs to be removed

Returns:

the value of key, or null if the key was not found.

Class java.util.Hashtable

java.lang.Object



Declaration

```
public class Hashtable
    extends Dictionary
    implements Cloneable
```

Description

Hashtable class. Maps keys to values. Any object can be used as a key and/or value.

To successfully 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

Constructors

Hashtable(int, float)

Hashtable(int)

Hashtable()

Methods

clear()

clone()

contains(Object)

containsKey(Object)

elements()

get(Object)

isEmpty()

keys()

put(Object, Object)

rehash()

remove(Object)

size()

toString()

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.

size

```
public int size()
```

Returns the number of elements contained in the hashtable.
Overrides:

size in class Dictionary

isEmpty

```
public boolean isEmpty()
```

Returns true if the hashtable contains no elements.

Overrides:

isEmpty in class Dictionary

keys

```
public synchronized Enumeration keys()
```

Returns an enumeration of the hashtable's keys.

Overrides:

keys in class Dictionary

See Also:

elements, Enumeration

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

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

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

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()
```

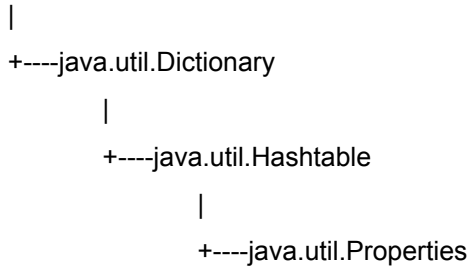
Converts to a rather lengthy String.

Overrides:

toString in class Object

Class java.util.Properties

java.lang.Object



Declaration

```
public class Properties
    extends Hashtable
```

Description

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.

Variables

defaults

Constructors

```
Properties()
Properties(Properties)
```

Methods

```
getProperty(String)
getProperty(String, String)
list(PrintStream)
load(InputStream)
propertyNames()
save(OutputStream, String)
```


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

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.

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.

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

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.awt.Dimension

java.lang.Object

|

+----java.awt.Dimension

Declaration

```
public class Dimension
```

```
    extends Object
```

Description

A class to encapsulate a width and a height Dimension.

Variables

height

width

Constructors

Dimension()

Dimension(Dimension)

Dimension(int, int)

Method

toString()

width

Applies to

Class `java.awt.Dimension`

Declaration

```
public int width
```

Description

The width dimension.

height

Applies to

Class `java.awt.Dimension`

Declaration

```
public int height
```

Description

The height dimension.

Dimension

Applies to

Class `java.awt.Dimension`

Declaration

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

Description

public Dimension()

Constructs a Dimension with a 0 width and 0 height.

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

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

Applies to

Class `java.awt.Dimension`

Declaration

```
public String toString()
```

Description

Returns the String representation of this Dimension's values.

Overrides:

toString in class Object

Interface `java.util.Enumeration`

Declaration

```
public interface Enumeration  
    extends Object
```

Description

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

Methods

```
hasMoreElements()  
nextElement()
```

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.awt.Event

java.lang.Object

|

+----java.awt.Event

Declaration

```
public class Event
    extends Object
```

Description

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

Variables

ACTION_EVENT

ALT_MASK

CTRL_MASK

DOWN

END

F1

F10

F11

F12

F2

F3

F4

F5

F6

F7

F8

F9

GOT_FOCUS
HOME
KEY_ACTION
KEY_ACTION_RELEASE
KEY_PRESS
KEY_RELEASE
LEFT
LIST_DESELECT
LIST_SELECT
LOAD_FILE
LOST_FOCUS
META_MASK
MOUSE_DOWN
MOUSE_DRAG
MOUSE_ENTER
MOUSE_EXIT
MOUSE_MOVE
MOUSE_UP
PGDN
PGUP
RIGHT
SAVE_FILE
SCROLL_ABSOLUTE
SCROLL_LINE_DOWN
SCROLL_LINE_UP
SCROLL_PAGE_DOWN
SCROLL_PAGE_UP
SHIFT_MASK
UP
WINDOW_DEICONIFY
WINDOW_DESTROY
WINDOW_EXPOSE
WINDOW_ICONIFY
WINDOW_MOVED
arg
clickCount

evt

id

key

modifiers

target

when

x

y

Constructors

Event(Object, long, int, int, int, int, int, Object)

Event(Object, long, int, int, int, int, int)

Event(Object, int, Object)

Method

controlDown()

metaDown()

paramString()

shiftDown()

toString()

translate(int, int)

Event Variables

Applies to

Class java.awt.Event

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.

HOME

public final static int HOME

The home key.

END

public final static int END

The end key.

PGUP

public final static int PGUP

The page up key.

PGDN

public final static int PGDN

The page down key.

UP

```
public final static int UP
```

The up arrow key.

DOWN

```
public final static int DOWN
```

The down arrow key.

LEFT

```
public final static int LEFT
```

The left arrow key.

RIGHT

```
public final static int RIGHT
```

The right arrow key.

F1

```
public final static int F1
```

The F1 function key.

F2

```
public final static int F2
```

The F2 function key.

F3

```
public final static int F3
```

The F3 function key.

F4

```
public final static int F4
```

The F4 function key.

F5

```
public final static int F5
```

The F5 function key.

F6

public final static int F6

The F6 function key.

F7

public final static int F7

The F7 function key.

F8

public final static int F8

The F8 function key.

F9

public final static int F9

The F9 function key.

F10

public final static int F10

The F10 function key.

F11

public final static int F11

The F11 function key.

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.

MOUSE_ENTER

```
public final static int MOUSE_ENTER
```

The mouse enter event.

MOUSE_EXIT

```
public final static int MOUSE_EXIT
```

The mouse exit event.

MOUSE_DRAG

```
public final static int MOUSE_DRAG
```

The mouse drag event.

SCROLL_LINE_UP

```
public final static int SCROLL_LINE_UP
```

The line up scroll event.

SCROLL_LINE_DOWN

```
public final static int SCROLL_LINE_DOWN
```

The line down scroll event.

SCROLL_PAGE_UP

```
public final static int SCROLL_PAGE_UP
```

The page up scroll event.

SCROLL_PAGE_DOWN

```
public final static int SCROLL_PAGE_DOWN
```

The page down scroll event.

SCROLL_ABSOLUTE

```
public final static int SCROLL_ABSOLUTE
```

The absolute scroll event.

LIST_SELECT

public final static int LIST_SELECT

LIST_DESELECT

public final static int LIST_DESELECT

ACTION_EVENT

public final static int ACTION_EVENT

An action event.

LOAD_FILE

public final static int LOAD_FILE

A file loading event.

SAVE_FILE

public final static int SAVE_FILE

A file saving event.

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

Applies to

Declaration

public Object target

Description

The target component.

when

Applies to

Class java.awt.Event

Declaration

public long when

Description

The time stamp.

id

Applies to

Class java.awt.Event

Declaration

```
public int id
```

Description

The type of this event.

x

Applies to

Class `java.awt.Event`

Declaration

```
public int x
```

Description

The x coordinate of the event.

y

Applies to

Class `java.awt.Event`

Declaration

```
public int y
```

Description

The y coordinate of the event.

key

Applies to

Class java.awt.Event

Declaration

```
public int key
```

Description

The key that was pressed in a keyboard event.

modifiers

Applies to

Class java.awt.Event

Declaration

```
public int modifiers
```

Description

The state of the modifier keys.

clickCount

Applies to

Class `java.awt.Event`

Declaration

```
public int clickCount
```

Description

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

Applies to

Class java.awt.Event

Declaration

public Object arg

Description

An arbitraty argument.

evt

Applies to

Class `java.awt.Event`

Declaration

```
public Event evt
```

Description

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

Event

Applies to

Class java.awt.Event

Declaration

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

Description

```
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

```
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

```
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

translate

Applies to

Class java.awt.Event

Declaration

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

Description

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

shiftDown

Applies to

Class `java.awt.Event`

Declaration

```
public boolean shiftDown()
```

Description

Checks if the shift key is down.

See Also:

`modifiers`, `controlDown`, `metaDown`

controlDown

Applies to

Class `java.awt.Event`

Declaration

```
public boolean controlDown()
```

Description

Checks if the control key is down.

See Also:

modifiers, shiftDown, metaDown

metaDown

Applies to

Class `java.awt.Event`

Declaration

```
public boolean metaDown()
```

Description

Checks if the meta key is down.

See Also:

`modifiers`, `shiftDown`, `controlDown`

paramString

Applies to

Class java.awt.Event

Declaration

protected String paramString()

Description

Returns the parameter String of this Event.

toString

Applies to

Class java.awt.Event

Declaration

```
public String toString()
```

Description

Returns the String representation of this Event's values.

Overrides:

toString in class Object

Class sun.tools.debug.RemoteDouble

java.lang.Object

```
|
+----sun.tools.debug.RemoteValue
      |
      +----sun.tools.debug.RemoteDouble
```

Declaration

```
public class RemoteDouble
    extends RemoteValue
```

Description

The RemoteDouble class extends RemoteValue for doubles.

See Also:

RemoteValue, RemoteDebugger

Methods

```
get()
toString()
typeName()
```


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

java.lang.Object

```
|
+----sun.tools.debug.Field
      |
      +----sun.tools.debug.RemoteField
```

Declaration

```
public class RemoteField
    extends Field
    implements AgentConstants
```

Description

A RemoteField allows access to a variable or method of an object or class in a remote Java interpreter.

See Also:

RemoteStackVariable

Methods

```
getModifiers()
getName()
getType()
isStatic()
toString()
```

getName

```
public String getName()
```

Returns the name of the field.

getType

```
public String getType()
```

Returns a type string describing the field.

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.

isStatic

```
public boolean isStatic()
```

Returns whether the field is static (a class variable or method).

toString

```
public String toString()
```

Returns a String that represents the value of this Object.

Overrides:

toString in class Object

Class java.io.File

```
java.lang.Object
|
+----java.io.File
```

Declaration

```
public class File
    extends Object
```

Description

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.

Variables

[pathSeparator](#)
[pathSeparatorChar](#)
[separator](#)
[separatorChar](#)

Constructors

[File\(String\)](#)
[File\(String, String\)](#)
[File\(File, String\)](#)

Method

[canRead\(\)](#)

canWrite()

delete()

equals(Object)

exists()

getAbsolutePath()

getName()

getParent()08

getPath()

hashCode()

isAbsolute()

isDirectory()

isFile()

lastModified()

length()

list()

list(FilenameFilter)

mkdir()

mkdirs()

renameTo(File)

toString()

separator

Applies to

Class java.io.File

Declaration

```
public final static String separator
```

Description

The system dependent file separator String.

separatorChar

Applies to

Class java.io.File

Declaration

```
public final static char separatorChar
```

Description

The system dependent file separator character.

pathSeparator

Applies to

Class java.io.File

Declaration

```
public final static String pathSeparator
```

Description

The system dependent path separator string.

pathSeparatorChar

Applies to

Class java.io.File

Declaration

```
public final static char pathSeparatorChar
```

Description

The system dependent path separator character.

File

Applies to

Class java.io.File

Declaration

```
public File(String path)
public File(String path,
            String name)
public File(File dir,
            String name)
```

Description

public File(String path)

Creates a File object.

Parameters:

path - the file path

Throws: NullPointerException

If the file path is equal to null.

public File(String path, String name)

Creates a File object from the specified directory.

Parameters:

path - the directory path

name - the file name

```
public File(File dir,  
            String name)
```

Creates a File object (given a directory File object).

Parameters:

dir - the directory

name - the file name

getName

Applies to

Class java.io.File

Declaration

```
public String getName()
```

Description

Gets the name of the file. This method does not include the directory.

Returns:

the file name.

getPath

Applies to

Class java.io.File

Declaration

```
public String getPath()
```

Description

Gets the path of the file.

Returns:

the file path.

getAbsolutePath

Applies to

Class java.io.File

Declaration

```
public String getAbsolutePath()
```

Description

Gets the absolute path of the file.

Returns:

the absolute file path.

getParent

Applies to

Class java.io.File

Declaration

```
public String getParent()
```

Description

Gets the name of the parent directory.

Returns:

the parent directory, or null if one is not found.

exists

Applies to

Class java.io.File

Declaration

```
public boolean exists()
```

Description

Returns a boolean indicating whether or not a file exists.

canWrite

Applies to

Class java.io.File

Declaration

```
public boolean canWrite()
```

Description

Returns a boolean indicating whether or not a writable file exists.

canRead

Applies to

Class java.io.File

Declaration

```
public boolean canRead()
```

Description

Returns a boolean indicating whether or not a readable file exists.

isFile

Applies to

Class java.io.File

Declaration

```
public boolean isFile()
```

Description

Returns a boolean indicating whether or not a normal file exists.

isDirectory

Applies to

Class java.io.File

Declaration

```
public boolean isDirectory()
```

Description

Returns a boolean indicating whether or not a directory file exists.

isAbsolute

Applies to

Class java.io.File

Declaration

```
public boolean isAbsolute()
```

Description

Returns a boolean indicating whether the file name is absolute.

lastModified

Applies to

Class java.io.File

Declaration

```
public long lastModified()
```

Description

Returns the last modification time. The return value should only be used to compare modification dates. It is meaningless as an absolute time.

length

Applies to

Class java.io.File

Declaration

```
public long length()
```

Description

Returns the length of the file.

mkdir

Applies to

Class java.io.File

Declaration

```
public boolean mkdir()
```

Description

Creates a directory and returns a boolean indicating the success of the creation.

renameTo

Applies to

Class java.io.File

Declaration

```
public boolean renameTo(File dest)
```

Description

Renames a file and returns a boolean indicating whether or not this method was successful.

Parameters:

dest - the new file name

mkdirs

Applies to

Class java.io.File

Declaration

```
public boolean mkdirs()
```

Description

Creates all directories in this path. This method returns true if all directories in this path are created.

list

Applies to

Class java.io.File

Declaration

```
public String[] list()  
public String[] list(FilenameFilter filter)
```

Description

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 ".." .

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

Applies to

Class java.io.File

Declaration

```
public boolean delete()
```

Description

Deletes the specified file. Returns true if the file could be deleted.

hashCode

Applies to

Class java.io.File

Declaration

```
public int hashCode()
```

Description

Computes a hashcode for the file.

Overrides:

hashCode in class Object

equals

Applies to

Class java.io.File

Declaration

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

Description

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

Applies to

Class java.io.File

Declaration

```
public String toString()
```

Description

Returns a String object representing this file's path.

Overrides:

toString in class Object

Class java.io.FileDescriptor

java.lang.Object

|

+----java.io.FileDescriptor

Declaration

```
public final class FileDescriptor
    extends Object
```

Variables

```
err
in
out
```

Constructors

```
FileDescriptor()
```

Methods

```
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.

Interface `java.awt.peer.FileDialogPeer`

Declaration

```
public interface FileDialogPeer
    extends Object
    extends DialogPeer
```

Methods

```
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.io.FilenameFilter`

Declaration

```
public interface FilenameFilter  
    extends Object
```

Description

A filter interface for file names.

See Also:

`File`

Methods

```
accept(File, String)
```

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.awt.image.FilteredImageSource

java.lang.Object

|

+----java.awt.image.FilteredImageSource

Declaration

```
public class FilteredImageSource
```

```
    extends Object
```

```
    implements ImageProducer
```

Description

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

Constructors

```
FilteredImageSource(ImageProducer, ImageFilter)
```

Methods

```
addConsumer(ImageConsumer)
```

```
isConsumer(ImageConsumer)
```

```
removeConsumer(ImageConsumer)
```

```
requestTopDownLeftRightResend(ImageConsumer)
```

startProduction(ImageConsumer)

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

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

removeConsumer

```
public synchronized void removeConsumer(ImageConsumer ic)
```

Removes an ImageConsumer from the list of consumers interested in data for this image.

See Also:

ImageConsumer

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

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.FlowLayout

java.lang.Object

|

+----java.awt.FlowLayout

Declaration

```
public class FlowLayout
    extends Object
    implements LayoutManager
```

Description

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.

Variables

CENTER
LEFT
RIGHT

Constructors

FlowLayout()
FlowLayout(int)
FlowLayout(int, int, int)

Methods

addLayoutComponent(String, Component)
layoutContainer(Container)
minimumLayoutSize(Container)
preferredLayoutSize(Container)
removeLayoutComponent(Component)
toString()

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

java.lang.Object

|

+----java.awt.Font

Declaration

```
public class Font
    extends Object
```

Description

A class that produces font objects.

Variables

BOLD

ITALIC

PLAIN

name

size

style

Constructors

Font(String, int, int)

Methods

equals(Object)

getFamily()

getFont(String)

getFont(String, Font)

getName()

getSize()

getStyle()

hashCode()

isBold()

isItalic()

isPlain()

toString()

PLAIN

Applies to

Class `java.awt.Font`

Declaration

```
public final static int PLAIN
```

Description

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

BOLD

Applies to

Class `java.awt.Font`

Declaration

```
public final static int BOLD
```

Description

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

ITALIC

Applies to

Class `java.awt.Font`

Declaration

```
public final static int ITALIC
```

Description

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

name

Applies to

Class java.awt.Font

Declaration

protected String name

Description

The logical name of this font.

style

Applies to

Class `java.awt.Font`

Declaration

```
protected int style
```

Description

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

size

Applies to

Class `java.awt.Font`

Declaration

protected int size

Description

The point size of this font.

Font

Applies to

Class `java.awt.Font`

Declaration

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

Description

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`

getFamily

Applies to

Class `java.awt.Font`

Declaration

```
public String getFamily()
```

Description

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

See Also:

`getName`

getName

Applies to

Class `java.awt.Font`

Declaration

```
public String getName()
```

Description

Gets the logical name of the font.

See Also:

`getFamily`

getStyle

Applies to

Class `java.awt.Font`

Declaration

```
public int getStyle()
```

Description

Gets the style of the font.

See Also:

`isPlain`, `isBold`, `isItalic`

getSize

Applies to

Class `java.awt.Font`

Declaration

```
public int getSize()
```

Description

Gets the point size of the font.

isPlain

Applies to

Class `java.awt.Font`

Declaration

```
public boolean isPlain()
```

Description

Returns true if the font is plain.

See Also:

`getStyle`

isBold

Applies to

Class `java.awt.Font`

Declaration

```
public boolean isBold()
```

Description

Returns true if the font is bold.

See Also:

`getStyle`

isItalic

Applies to

Class `java.awt.Font`

Declaration

```
public boolean isItalic()
```

Description

Returns true if the font is italic.

See Also:

`getStyle`

getFont

Applies to

Class java.awt.Font

Declaration

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

Description

public static Font getFont(String nm)

Gets a font from the system properties list.

Parameters:

nm - the property name

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

Applies to

Class `java.awt.Font`

Declaration

```
public int hashCode()
```

Description

Returns a hashcode for this font.

Overrides:

hashCode in class `Object`

equals

Applies to

Class java.awt.Font

Declaration

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

Description

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

toString

Applies to

Class `java.awt.Font`

Declaration

```
public String toString()
```

Description

Converts this object to a String representation.

Overrides:

toString in class Object

Class java.awt.FontMetrics

java.lang.Object

|

+----java.awt.FontMetrics

Declaration

```
public class FontMetrics
    extends Object
```

Description

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

Constructors

[FontMetrics\(Font\)](#)

Methods

[bytesWidth\(byte\[\], int, int\)](#)

[charWidth\(int\)](#)

[charWidth\(char\)](#)

[charsWidth\(char\[\], int, int\)](#)

[getAscent\(\)](#)

[getDescent\(\)](#)

[getFont\(\)](#)

[getHeight\(\)](#)

[getLeading\(\)](#)

[getMaxAdvance\(\)](#)

[getMaxAscent\(\)](#)

[getMaxDescent\(\)](#)

[getMaxDescent\(\)](#)

[getWidths\(\)](#)

stringWidth(String)

toString()

FontMetrics

Applies to

Class `java.awt.FontMetrics`

Declaration

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

Description

Creates a new FontMetrics object with the specified font.

Parameters:

font - the font

See Also:

Font

getFont

Applies to

Class `java.awt.FontMetrics`

Declaration

```
public Font getFont()
```

Description

Gets the font.

getLeading

Applies to

Class `java.awt.FontMetrics`

Declaration

```
public int getLeading()
```

Description

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

Applies to

Class `java.awt.FontMetrics`

Declaration

```
public int getAscent()
```

Description

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

See Also:

`getMaxAscent`

getDescent

Applies to

Class `java.awt.FontMetrics`

Declaration

```
public int getDescent()
```

Description

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

See Also:

`getMaxDescent`

getHeight

Applies to

Class `java.awt.FontMetrics`

Declaration

```
public int getHeight()
```

Description

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

Applies to

Class `java.awt.FontMetrics`

Declaration

```
public int getMaxAscent()
```

Description

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

Applies to

Class `java.awt.FontMetrics`

Declaration

```
public int getMaxDescent()
```

Description

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

See Also:

`getDescent`

getMaxDecent

Applies to

Class `java.awt.FontMetrics`

Declaration

```
public int getMaxDecent()
```

Description

For backward compatibility only.

See Also:

`getMaxDescent`

getMaxAdvance

Applies to

Class `java.awt.FontMetrics`

Declaration

```
public int getMaxAdvance()
```

Description

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

Returns:

-1 if the max advance is not known.

charWidth

Applies to

Class `java.awt.FontMetrics`

Declaration

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

Description

public int charWidth(int ch)

Returns the width of the specified character in this Font.

Parameters:

ch - the specified font

See Also:

stringWidth

public int charWidth(char ch)

Returns the width of the specified character in this Font.

Parameters:

ch - the specified font

See Also:

stringWidth

stringWidth

Applies to

Class `java.awt.FontMetrics`

Declaration

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

Description

Returns the width of the specified String in this Font.

Parameters:

str - the String to be checked

See Also:

`charsWidth`, `bytesWidth`

charsWidth

Applies to

Class `java.awt.FontMetrics`

Declaration

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

Description

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

Applies to

Class `java.awt.FontMetrics`

Declaration

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

Description

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

Applies to

Class `java.awt.FontMetrics`

Declaration

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

Description

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

toString

Applies to

Class java.awt.FontMetrics

Declaration

```
public String toString()
```

Description

Returns the String representation of this FontMetric's values.

Overrides:

toString in class Object

Interface `java.awt.peer.FramePeer`

Declaration

```
public interface FramePeer
    extends Object
    extends WindowPeer
```

Methods

```
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)
```

setResizable

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

setCursor

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

Class java.awt.Graphics

java.lang.Object

|

+----java.awt.Graphics

Declaration

```
public class Graphics
    extends Object
```

Description

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

Constructors

```
Graphics()
```

Methods

```
clearRect(int, int, int, int)
clipRect(int, int, int, int)
copyArea(int, int, int, int, int, int)
create()
create(int, int, int, int)
dispose()
draw3DRect(int, int, int, int, boolean)
drawArc(int, int, int, int, int, int)
drawBytes(byte[], int, int, int, int)
drawChars(char[], int, int, int, int)
drawImage(Image, int, int, ImageObserver)
drawImage(Image, int, int, int, int, ImageObserver)
drawImage(Image, int, int, Color, ImageObserver)
drawImage (Image, int, int, int, int, Color, ImageObserver)
drawLine(int, int, int, int)
```

drawOval(int, int, int, int)
drawPolygon(int[], int[], int)
drawPolygon(Polygon)
drawRect(int, int, int, int)
drawRoundRect(int, int, int, int, int, int)
drawString(String, int, int)
fill3DRect(int, int, int, int, boolean)
fillArc(int, int, int, int, int, int)
fillOval(int, int, int, int)
fillPolygon(int[], int[], int)
fillPolygon(Polygon)
fillRect(int, int, int, int)
fillRoundRect(int, int, int, int, int, int)
finalize()
getClipRect()
getColor()
getFont()
getFontMetrics()
getFontMetrics(Font)
setColor(Color)
setFont(Font)
setPaintMode()
setXORMode(Color)
toString()
translate(int, int)

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

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

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

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`

`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

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

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

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

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

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

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

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

drawPolygon

```
public void drawPolygon(Polygon p)
```

Draws a polygon defined by the specified point.

Parameters:

p - the specified polygon

See Also:

fillPolygon

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

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

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

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

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

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

java.lang.Object

|

+----java.awt.GridBagConstraints

Declaration

```
public class GridBagConstraints
    extends Object
    implements Cloneable
```

Description

GridBagConstraints is used to specify constraints for components laid out using the GridBagLayout class.

See Also:

GridBagLayout

Variables

BOTH
CENTER
EAST
HORIZONTAL
NONE
NORTH
NORTHEAST
NORTHWEST
RELATIVE
REMAINDER
SOUTH
SOUTHEAST
SOUTHWEST
VERTICAL

WEST
anchor
fill
gridheight
gridwidth
gridx
gridy
insets
ipadx
ipady
weightx
weighty

Constructors

GridBagConstraints()

Methods

clone()

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

gridx

public int gridx

gridy

public int gridy

gridwidth

public int gridwidth

gridheight

public int gridheight

weightx

public double weightx

weighty

public double weighty

anchor

public int anchor

fill

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

java.lang.Object

|

+----java.awt.GridBagLayout

Declaration

```
public class GridBagLayout
    extends Object
    implements LayoutManager
```

Description

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 `GridBagConstraints.NONE` (the default), `GridBagConstraints.HORIZONTAL` (make the component wide enough to fill its display area horizontally, but don't change its height), `GridBagConstraints.VERTICAL` (make the component tall enough to fill its display area vertically, but don't change its width), and `GridBagConstraints.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,
                               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();
    }
}

```

Variables

MAXGRIDSIZE
MINSIZE
PREFERRED_SIZE
columnWeights
columnWidths
comptable
defaultConstraints
layoutInfo
rowHeights
rowWeights

Constructors

GridBagLayout()

Methods

AdjustForGravity(GridBagConstraints, Rectangle)
ArrangeGrid(Container)
DumpConstraints (GridBagConstraints)
DumpLayoutInfo(GridBagLayoutInfo)
GetLayoutInfo(Container, int)
GetMinSize (Container, GridBagLayoutInfo)
addLayoutComponent (String, Component)
getConstraints(Component)
getLayoutDimensions()
getLayoutOrigin()
getLayoutWeights()
layoutContainer(Container)
location(int, int)
lookupConstraints(Component)
minimumLayoutSize(Container)
preferredLayoutSize(Container)
removeLayoutComponent(Component)
setConstraints(Component, GridBagConstraints)
toString()

MAXGRIDSIZE

protected final static int MAXGRIDSIZE

MINSIZE

protected final static int MINSIZE

PREFERREDSIZE

protected final static int PREFERREDSIZE

comptable

protected Hashtable comptable

defaultConstraints

protected GridBagConstraints defaultConstraints

layoutInfo

protected GridBagLayoutInfo layoutInfo

columnWidths

public int columnWidths[]

rowHeights

public int rowHeights[]

columnWeights

public double columnWeights[]

rowWeights

```
public double rowWeights[]
```

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

java.lang.Object

|

+----java.awt.GridLayout

Declaration

```
public class GridLayout
```

```
    extends Object
```

```
    implements LayoutManager
```

Description

A layout manager for a container that lays out grids.

Constructors

```
    GridLayout(int, int)
```

```
    GridLayout(int, int, int, int)
```

Methods

```
    addLayoutComponent(String, Component)
```

```
    layoutContainer(Container)
```

```
    minimumLayoutSize(Container)
```

```
    preferredLayoutSize(Container)
```

```
    removeLayoutComponent(Component)
```

```
    toString()
```

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 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

Class `java.awt.Image`

`java.lang.Object`

|

+----`java.awt.Image`

Declaration

```
public class Image
```

```
extends Object
```

Description

The image class is an abstract class. The image must be obtained in a platform specific way.

Variables

[UndefinedProperty](#)

Constructors

[Image\(\)](#)

Methods

[flush\(\)](#)

[getGraphics\(\)](#)

[getHeight\(ImageObserver\)](#)

[getProperty\(String, ImageObserver\)](#)

[getSource\(\)](#)

[getWidth\(ImageObserver\)](#)

UndefinedProperty

Applies to

Class `java.awt.Image`

Declaration

```
public final static Object UndefinedProperty
```

Description

The UndefinedProperty object should be returned whenever a property which was not defined for a particular image is fetched.

Image

Applies to

Class `java.awt.Image`

Declaration

```
public Image()
```

getWidth

Applies to

Class `java.awt.Image`

Declaration

```
public abstract int getWidth(ImageObserver observer)
```

Description

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

Applies to

Class java.awt.Image

Declaration

```
public abstract int getHeight(ImageObserver observer)
```

Description

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

Applies to

Class `java.awt.Image`

Declaration

```
public abstract ImageProducer getSource()
```

Description

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

Applies to

Class `java.awt.Image`

Declaration

```
public abstract Graphics getGraphics()
```

Description

Gets a graphics object to draw into this image. This will only work for off-screen images.

See Also:

Graphics

getProperty

Applies to

Class `java.awt.Image`

Declaration

```
public abstract Object getProperty(String name,  
                                   ImageObserver observer)
```

Description

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

Applies to

Class `java.awt.Image`

Declaration

```
public abstract void flush()
```

Description

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`

Declaration

```
public interface ImageConsumer
    extends Object
```

Description

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

Variables

COMPLETESCANLINES
IMAGEABORTED
IMAGEERROR
RANDOMPIXELORDER
SINGLEFRAME
SINGLEFRAMEDONE
SINGLEPASS
STATICIMAGEDONE
TOPDOWNLEFTRIGHT

Methods

imageComplete(int)
setColorModel(ColorModel)
setDimensions(int, int)
setHints(int)
setPixels (int, int, int, int, ColorModel, byte[], int, int)
setPixels (int, int, int, int, ColorModel, int[], int, int)
setProperties(Hashtable)

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

setDimensions

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

The dimensions of the source image are reported using the setDimensions method call.

setProperties

```
public abstract void setProperties(Hashtable props)
```

Sets the extensible list of properties associated with this image.

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

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.

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

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

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 occurred. 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

java.lang.Object

|

+----java.awt.image.ImageFilter

Declaration

```
public class ImageFilter
```

```
    extends Object
```

```
    implements ImageConsumer, Cloneable
```

Description

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

Variables

consumer

Constructors

ImageFilter()

Methods

clone()

getFilterInstance(ImageConsumer)

imageComplete(int)

resendTopDownLeftRight(ImageProducer)
setColorModel(ColorModel)
setDimensions(int, int)
setHints(int)
setPixels (int, int, int, int, ColorModel, byte[], int, int)
setPixels (int, int, int, int, ColorModel, int[], int, int)
setProperties(Hashtable)

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

Class `java.awt.image.CropImageFilter`

`java.lang.Object`

```
|
+----java.awt.image.ImageFilter
      |
      +----java.awt.image.CropImageFilter
```

Declaration

```
public class CropImageFilter
    extends ImageFilter
```

Description

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`

Constructors

`CropImageFilter(int, int, int, int)`

Methods

```
setDimensions(int, int)
setPixels (int, int, int, int, ColorModel, byte[], int, int)
setPixels (int, int, int, int, ColorModel, int[], int, int)
setProperties(Hashtable)
```

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

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

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.RGBImageFilter

java.lang.Object

```
|
+----java.awt.image.ImageFilter
      |
      +----java.awt.image.RGBImageFilter
```

Declaration

```
public class RGBImageFilter
    extends ImageFilter
```

Description

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

Variables

canFilterIndexColorModel

newmodel

origmodel

Constructors

RGBImageFilter()

Methods

filterIndexColorModel(IndexColorModel)

filterRGB(int, int, int)

filterRGBPixels(int, int, int, int, int[], int, int)

setColorModel(ColorModel)

setPixels(int, int, int, int, ColorModel, byte[], int, int)

setPixels(int, int, int, int, ColorModel, int[], int, int)

substituteColorModel(ColorModel, ColorModel)

origmodel

protected ColorModel origmodel

newmodel

protected ColorModel newmodel

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

RGBImageFilter

```
public RGBImageFilter()
```

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`

`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

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

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

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

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

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`

Interface `java.awt.image.ImageObserver`

Declaration

```
public interface ImageObserver
    extends Object
```

Description

An asynchronous update interface for receiving notifications about Image information as the Image is constructed.

Variables

ABORT
ALLBITS
ERROR
FRAMEBITS
HEIGHT
PROPERTIES
SOMEBITS
WIDTH

Methods

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

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`

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`

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

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

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`

Declaration

```
public interface ImageProducer  
    extends Object
```

Description

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

Methods

```
addConsumer(ImageConsumer)  
isConsumer(ImageConsumer)  
removeConsumer(ImageConsumer)  
requestTopDownLeftRightResend(ImageConsumer)  
startProduction(ImageConsumer)
```

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.net.InetAddress

java.lang.Object

|

+----java.net.InetAddress

Declaration

```
public final class InetAddress
    extends Object
```

Description

A class that represents Internet addresses.

Methods

```
equals(Object)
getAddress()
getAllByName(String)
getByName(String)
getHostName()
getLocalHost()
hashCode()
toString()
```


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 an 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.io.InputStream

java.lang.Object

|

+----java.io.InputStream

Declaration

```
public class InputStream
    extends Object
```

Description

An abstract class representing an input stream of bytes. All InputStreams are based on this class.

See Also:

OutputStream, FilterInputStream, BufferedInputStream, DataInputStream , ByteArrayInputStream, PushbackInputStream

Constructors

InputStream()

Methods

available()

close()

mark(int)

markSupported()

read()

read(byte[])

read(byte[], int, int)

reset()

skip(long)

InputStream

```
public InputStream()
```

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.

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()
```

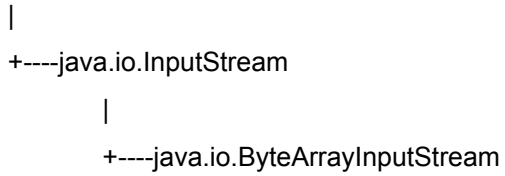
Returns a boolean indicating whether or not this stream type supports mark/reset.

Returns:

true if this stream type supports mark/reset; false otherwise.

Class java.io.ByteArrayInputStream

java.lang.Object



Declaration

```
public class ByteArrayInputStream
    extends InputStream
```

Description

This class implements a buffer that can be used as an InputStream.

Variables

```
buf
count
pos
```

Constructors

```
ByteArrayInputStream(byte[])
ByteArrayInputStream(byte[], int, int)
```

Methods

```
available()
read()
read(byte[], int, int)
reset()
skip(long)
```

buf

protected byte buf[]

The buffer where data is stored.

pos

protected int pos

The current position in the buffer.

count

protected int count

The number of characters to use in the buffer.

ByteArrayInputStream

public ByteArrayInputStream(byte buf[])

Creates an ByteArrayInputStream from the specified array of bytes.

Parameters:

buf - the input buffer (not copied)

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

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.FileInputStream

java.lang.Object

```
|
+----java.io.InputStream
      |
      +----java.io.FileInputStream
```

Declaration

```
public class FileInputStream
    extends InputStream
```

Description

File input stream, can be constructed from a file descriptor or a file name.

See Also:

FileOutputStream, File

Constructors

```
FileInputStream(String)
FileInputStream(File)
FileInputStream(FileDescriptor)
```

Methods

```
available()
close()
finalize()
getFD()
read()
read(byte[])
read(byte[], int, int)
skip(long)
```


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.

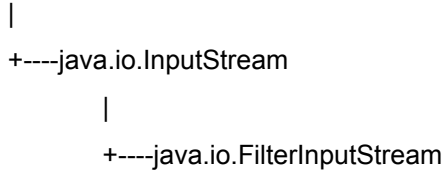
Overrides:

finalize in class Object



Class java.io.FilterInputStream

java.lang.Object



Declaration

```
public class FilterInputStream
    extends InputStream
```

Description

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.

Variables

in

Constructors

FilterInputStream(InputStream)

Methods

available()
close()
mark(int)
markSupported()
read()
read(byte[])
read(byte[], int, int)
reset()

skip(long)

in

protected InputStream in

The actual input stream.

FilterInputStream

protected FilterInputStream(InputStream in)

Creates an input stream filter.

Parameters:

in - the input stream

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

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

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 to be 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 parser, it just chugs along happily. If the stream is not of that type, the parser should throw 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()
```

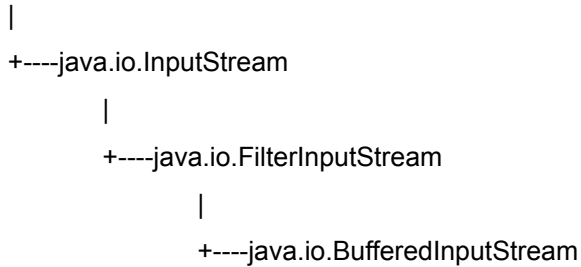
Returns true if this stream type supports mark/reset.

Overrides:

markSupported in class `InputStream`

Class java.io.BufferedInputStream

java.lang.Object



Declaration

```
public class BufferedInputStream
    extends FilterInputStream
```

Description

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.

Variables

```
buf
count
marklimit
markpos
pos
```

Constructors

```
BufferedInputStream(InputStream)
BufferedInputStream(InputStream, int)
```

Methods

```
available()
mark(int)
```

markSupported()
read()
read(byte[], int, int)
reset()
skip(long)

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

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`

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

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

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

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`

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()
```

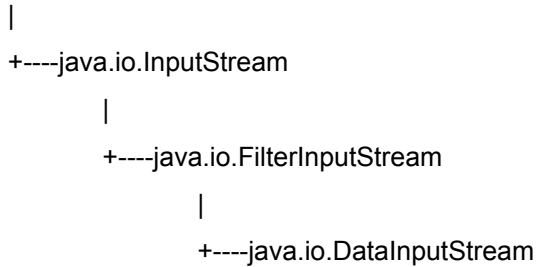
Returns a boolean indicating if this stream type supports mark/reset.

Overrides:

markSupported in class FilterInputStream

Class java.io.DataInputStream

java.lang.Object



Declaration

```
public class DataInputStream
    extends FilterInputStream
    implements DataInput
```

Description

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`

Constructors

`DataInputStream(InputStream)`

Methods

```
read(byte[])
read(byte[], int, int)
readBoolean()
readByte()
readChar()
```

readDouble()
readFloat()
readFully(byte[])
readFully(byte[], int, int)
readInt()
readLine()
readLong()
readShort()
readUTF()
readUTF(DataInput)
readUnsignedByte()
readUnsignedShort()
skipBytes(int)

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
```

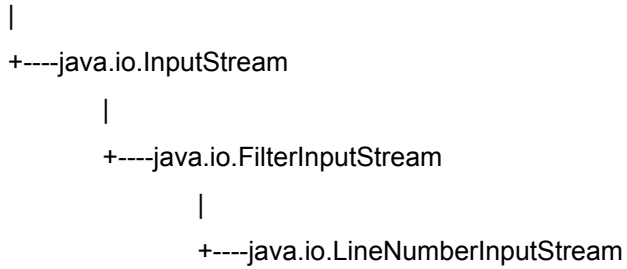
Reads a UTF format String from the given input stream.

Returns:

the String.

Class java.io.LineNumberInputStream

java.lang.Object



Declaration

```
public class LineNumberInputStream
    extends FilterInputStream
```

Description

An input stream that keeps track of line numbers.

Constructors

```
LineNumberInputStream(InputStream)
```

Methods

```
available()
getLineNumber()
mark(int)
read()
read(byte[], int, int)
reset()
setLineNumber(int)
skip(long)
```

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
```

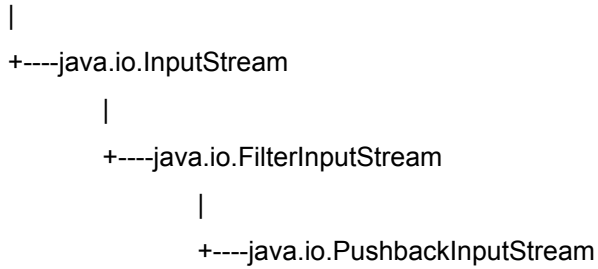
Repositions the stream to the last marked position. If the stream has not been marked, or if the mark has been invalidated, an `IOException` is thrown. Stream marks are intended to be used in situations where you need to read ahead a little to see what's in the stream. Often this is most easily done 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.PushbackInputStream

java.lang.Object



Declaration

```
public class PushbackInputStream
    extends FilterInputStream
```

Description

An input stream that has a 1 byte push back buffer.

Variables

pushBack

Constructors

PushbackInputStream(InputStream)

Methods

available()
markSupported()
read()
read(byte[], int, int)
unread(int)

pushBack

protected int pushBack

Push back character.

PushbackInputStream

public PushbackInputStream(InputStream in)

Creates a PushbackInputStream.

Parameters:

in - the input stream

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.PipedInputStream

java.lang.Object

```
|
+----java.io.InputStream
      |
      +----java.io.PipedInputStream
```

Declaration

```
public class PipedInputStream
    extends InputStream
```

Description

PipedInputStream must be connected to a PipedOutputStream to be useful. A thread reading from a PipedInputStream receives data from a thread writing to the PipedOutputStream it is connected to.

See Also:

PipedOutputStream

Constructors

```
PipedInputStream(PipedOutputStream)
PipedInputStream()
```

Methods

```
close()
connect(PipedOutputStream)
read()
read(byte[], int, int)
```

PipedInputStream

```
public PipedInputStream(PipedOutputStream src) throws IOException
```

Creates an input file from the specified PipedOutputStream.

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:

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.SequenceInputStream

java.lang.Object

```
|
+----java.io.InputStream
      |
      +----java.io.SequenceInputStream
```

Declaration

```
public class SequenceInputStream
    extends InputStream
```

Description

Converts a sequence of input streams into an InputStream.

Constructors

```
SequenceInputStream(Enumeration)
SequenceInputStream(InputStream, InputStream)
```

Methods

```
close()
read()
read(byte[], int, int)
```

SequenceInputStream

```
public SequenceInputStream(Enumeration e)
```

Constructs a new SequenceInputStream initialized to the specified list.

Parameters:

e - the list

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

read

```
public int read() throws IOException
```

Reads a stream, and upon reaching an EOF, flips to the next stream.

Overrides:

read in class `InputStream`

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

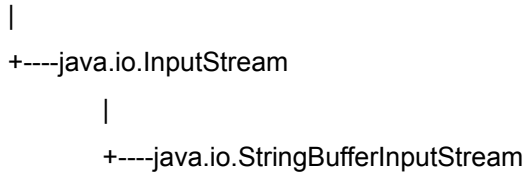
If an I/O error has occurred.

Overrides:

close in class InputStream

Class java.io.StringBufferInputStream

java.lang.Object



Declaration

```
public class StringBufferInputStream
    extends InputStream
```

Description

This class implements a String buffer that can be used as an InputStream.

Variables

```
buffer
count
pos
```

Constructors

```
StringBufferInputStream(String)
```

Methods

```
available()
read()
read(byte[], int, int)
reset()
skip(long)
```

buffer

protected String buffer

The buffer where data is stored.

pos

protected int pos

The position in the buffer.

count

protected int count

The number of characters to use in the buffer.

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.awt.Insets

java.lang.Object

|

+----java.awt.Insets

Declaration

```
public class Insets
```

```
    extends Object
```

```
    implements Cloneable
```

Description

The insets of a container. This class is used to layout containers.

See Also:

LayoutManager, Container

Variables

bottom

left

right

top

Constructors

Insets(int, int, int, int)

Methods

clone()

toString()

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.

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

toString

```
public String toString()
```

Returns a String object representing this Inset's values.

Overrides:

toString in class Object

clone

```
public Object clone()
```

Creates a clone of the object.

Overrides:

clone in class Object

Interface `java.awt.peer.LabelPeer`

Declaration

```
public interface LabelPeer
    extends Object
    extends ComponentPeer
```

Methods

```
setAlignment(int)
setText(String)
```


setText

```
public abstract void setText(String label)
```

setAlignment

```
public abstract void setAlignment(int alignment)
```

Interface `java.awt.LayoutManager`

Declaration

```
public interface LayoutManager  
    extends Object
```

Description

Defines the interface for classes that know how to layout Containers.

See Also:

Container

Methods

```
addLayoutComponent(String, Component)  
layoutContainer(Container)  
minimumLayoutSize(Container)  
preferredLayoutSize(Container)  
removeLayoutComponent(Component)
```

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 to 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

Interface `java.awt.peer.ListPeer`

Declaration

```
public interface ListPeer
    extends Object
    extends ComponentPeer
```

Methods

```
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)
```

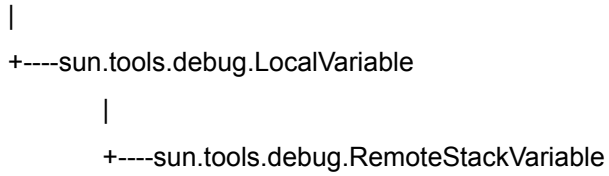
package sun.tools.debug

Variables

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

Class sun.tools.debug.RemoteStackVariable

java.lang.Object



Declaration

```
public class RemoteStackVariable
    extends LocalVariable
```

Description

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

Methods

```
getName()
getValue()
inScope()
```

getName

```
public String getName()
```

Return the name of a stack variable or argument.

getValue

```
public RemoteValue getValue()
```

Return the value of a stack variable or argument.

inScope

```
public boolean inScope()
```

Return whether variable is in scope.

Class java.lang.Math

java.lang.Object

|

+----java.lang.Math

Declaration

```
public final class Math
    extends Object
```

Description

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.

Methods

E

PI

Methods

IEEERemainder(double, double)

abs(int)

abs(long)

abs(float)

abs(double)

acos(double)

asin(double)

atan(double)

atan2(double, double)

ceil(double)

cos(double)

exp(double)

floor(double)
log(double)
max(int, int)
max(long, long)
max(float, float)
max(double, double)
min(int, int)
min(long, long)
min(float, float)
min(double, double)
pow(double, double)
random()
rint(double)
round(float)
round(double)
sin(double)
sqrt(double)
tan(double)

E

```
public final static double E
```

The float representation of the value E. E is equivalent to 2.7182818284590452354f in Java.

PI

```
public final static double PI
```

The float representation of the value Pi. Pi is equivalent to 3.14159265358979323846f in Java.

sin

```
public static double sin(double a)
```

Returns the trigonometric sine of an angle.

Parameters:

a - an assigned angle that is measured in radians

cos

```
public static double cos(double a)
```

Returns the trigonometric cosine of an angle.

Parameters:

a - an assigned angle that is measured in radians

tan

```
public static double tan(double a)
```

Returns the trigonometric tangent of an angle.

Parameters:

a - an assigned angle that is measured in radians

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)

acos

```
public static double acos(double a)
```

Returns the arc cosine of a, in the range of 0.0 through π .

Parameters:

a - (-1.0)

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.

exp

```
public static double exp(double a)
```

Returns the exponential number $e(2.718\dots)$ raised to the power of a .

Parameters:

a - an assigned value

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.

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.awt.MediaTracker`

`java.lang.Object`

|

+----`java.awt.MediaTracker`

Declaration

```
public class MediaTracker
extends Object
```

Description

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 identification 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.ERROR) != 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);
    }
}
}

```

Variables

ABORTED

COMPLETE

ERROR

LOADING

Constructors

MediaTracker(Component)

Methods

addImage(Image, int)

addImage(Image, int, int, int)

checkAll()

checkAll(boolean)

checkID(int)

checkID(int, boolean)

getErrorsAny()

getErrorsID(int)

isErrorAny()

isErrorID(int)

statusAll(boolean)

statusID(int, boolean)

waitForAll()

waitForAll(long)

waitForID(int)

waitForID(int, long)

LOADING

Applies to

Class `java.awt.MediaTracker`

Declaration

```
public final static int LOADING
```

Description

Flag indicating some media is currently being loaded.

See Also:

`statusAll`, `statusID`

ABORTED

Applies to

Class `java.awt.MediaTracker`

Declaration

```
public final static int ABORTED
```

Description

Flag indicating the download of some media was aborted.

See Also:

`statusAll`, `statusID`

ERRORED

Applies to

Class `java.awt.MediaTracker`

Declaration

```
public final static int ERRORED
```

Description

Flag indicating the download of some media encountered an error.

See Also:

`statusAll`, `statusID`

COMPLETE

Applies to

Class `java.awt.MediaTracker`

Declaration

```
public final static int COMPLETE
```

Description

Flag indicating the download of media completed successfully.

See Also:

`statusAll`, `statusID`

MediaTracker

Applies to

Class `java.awt.MediaTracker`

Declaration

```
public MediaTracker(Component comp)
```

Description

Creates a Media tracker to track images for a given Component.

Parameters:

comp - the component on which the images will eventually be drawn

addImage

Applies to

Class `java.awt.MediaTracker`

Declaration

```
public void addImage(Image image,  
                    int id)  
  
public synchronized void addImage(Image image,  
                                   int id,  
                                   int w,  
                                   int h)
```

Description

```
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

```
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

checkAll

Applies to

Class `java.awt.MediaTracker`

Declaration

```
public boolean checkAll()  
public synchronized boolean checkAll(boolean load)
```

Description

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

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`

isErrorAny

Applies to

Class `java.awt.MediaTracker`

Declaration

```
public synchronized boolean isErrorAny()
```

Description

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

Applies to

Class `java.awt.MediaTracker`

Declaration

```
public synchronized Object[] getErrorsAny()
```

Description

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

Applies to

Class `java.awt.MediaTracker`

Declaration

`public void waitForAll() throws InterruptedException`

`public synchronized boolean waitForAll(long ms) throws InterruptedException`

Description

`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.

`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

Applies to

Class `java.awt.MediaTracker`

Declaration

```
public int statusAll(boolean load)
```

Description

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

Applies to

Class `java.awt.MediaTracker`

Declaration

```
public boolean checkID(int id)
public synchronized boolean checkID(int id, boolean load)
```

Description

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.

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

Applies to

Class `java.awt.MediaTracker`

Declaration

```
public synchronized boolean isErrorID(int id)
```

Description

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

Applies to

Class `java.awt.MediaTracker`

Declaration

```
public synchronized Object[] getErrorsID(int id)
```

Description

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

Applies to

Class `java.awt.MediaTracker`

Declaration

`public void waitForID(int id) throws InterruptedException`

`public synchronized boolean waitForID(int id, long ms) throws InterruptedException`

Description

`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.

`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

Applies to

Class `java.awt.MediaTracker`

Declaration

```
public int statusID(int id, boolean load)
```

Description

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.image.MemoryImageSource`

`java.lang.Object`

|

+----`java.awt.image.MemoryImageSource`

Declaration

```
public class MemoryImageSource
    extends Object
    implements ImageProducer
```

Description

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 < h; y++) {
    int red = (y * 255) / (h - 1);
    for (int x = 0; x < w; x++) {
        int blue = (x * 255) / (w - 1);
        pix[index++] = (255 << 24) | (red << 16) | blue;
    }
}
Image img = createImage(new MemoryImageSource(w, h, pix, 0, w));
```

See Also:

`ImageProducer`

Constructors

MemoryImageSource(int, int, ColorModel, byte[], int, int)

MemoryImageSource(int, int, ColorModel, byte[], int, int, Hashtable)

MemoryImageSource(int, int, ColorModel, int[], int, int)

MemoryImageSource(int, int, ColorModel, int[], int, int, Hashtable)

MemoryImageSource(int, int, int[], int, int)

MemoryImageSource(int, int, int[], int, int, Hashtable)

Methods

addConsumer(ImageConsumer)

isConsumer(ImageConsumer)

removeConsumer(ImageConsumer)

requestTopDownLeftRightResend(ImageConsumer)

startProduction(ImageConsumer)

MemoryImageSource

Applies to

Class `java.awt.image.MemoryImageSource`

Declaration

```
public MemoryImageSource(int w, int h, ColorModel cm, byte pix[], int off, int scan)
public MemoryImageSource(int w, int h, ColorModel cm, byte pix[],
                        int off, int scan, Hashtable props)
public MemoryImageSource(int w, int h, ColorModel cm, int pix[], int off, int scan)
public MemoryImageSource(int w, int h, ColorModel cm, int pix[], int off,
                        int scan, Hashtable props)
public MemoryImageSource(int w, int h, int pix[], int off, int scan)
public MemoryImageSource(int w, int h, int pix[], int off, int scan, Hashtable props)
```

Description

`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.

**`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.

`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.

```
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.

```
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.

```
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](#)

addConsumer

Applies to

Class `java.awt.image.MemoryImageSource`

Declaration

```
public synchronized void addConsumer(ImageConsumer ic)
```

Description

Adds an ImageConsumer to the list of consumers interested in data for this image.

See Also:

ImageConsumer

isConsumer

Applies to

Class `java.awt.image.MemoryImageSource`

Declaration

```
public synchronized boolean isConsumer(ImageConsumer ic)
```

Description

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

removeConsumer

Applies to

Class `java.awt.image.MemoryImageSource`

Declaration

```
public synchronized void removeConsumer(ImageConsumer ic)
```

Description

Remove an ImageConsumer from the list of consumers interested in data for this image.

See Also:

ImageConsumer

startProduction

Applies to

Class `java.awt.image.MemoryImageSource`

Declaration

```
public void startProduction(ImageConsumer ic)
```

Description

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`

requestTopDownLeftRightResend

Applies to

Class `java.awt.image.MemoryImageSource`

Declaration

```
public void requestTopDownLeftRightResend(ImageConsumer ic)
```

Description

Requests that a given ImageConsumer have the image data delivered one more time in top-down, left-right order.

See Also:

ImageConsumer

Interface `java.awt.peer.MenuBarPeer`

Declaration

```
public interface MenuBarPeer
    extends Object
    extends MenuComponentPeer
```

Methods

```
addHelpMenu(Menu)  
addMenu(Menu)  
delMenu(int)
```

addMenu

Applies to

Interface `java.awt.peer.MenuBarPeer`

Declaration

```
public abstract void addMenu(Menu m)
```


delMenu

Applies to

Interface `java.awt.peer.MenuBarPeer`

Declaration

```
public abstract void delMenu(int index)
```

addHelpMenu

Applies to

Interface `java.awt.peer.MenuBarPeer`

Declaration

```
public abstract void addHelpMenu(Menu m)
```

Class java.awt.MenuComponent

java.lang.Object

|

+----java.awt.MenuComponent

Declaration

```
public class MenuComponent
    extends Object
```

Description

The super class of all menu related components.

Constructors

[MenuComponent\(\)](#)

Methods

[getFont\(\)](#)

[getParent\(\)](#)

[getPeer\(\)](#)

[paramString\(\)](#)

[postEvent\(Event\)](#)

[removeNotify\(\)](#)

[setFont\(Font\)](#)

[toString\(\)](#)

MenuComponent

Applies to

Class `java.awt.MenuComponent`

Declaration

```
public MenuComponent()
```

getParent

Applies to

Declaration

```
public MenuContainer getParent()
```

Description

Returns the parent container.

getPeer

Applies to

Class `java.awt.MenuComponent`

Declaration

```
public MenuComponentPeer getPeer()
```

Description

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

Applies to

Class `java.awt.MenuComponent`

Declaration

```
public Font getFont()
```

Description

Gets the font used for this MenuItem.

Returns:

the font if one is used; null otherwise.

setFont

Applies to

Class `java.awt.MenuComponent`

Declaration

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

Description

Sets the font to be used for this MenuItem to the specified font.

Parameters:

f - the font to be set

removeNotify

Applies to

Class `java.awt.MenuComponent`

Declaration

```
public void removeNotify()
```

Description

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

Applies to

Class `java.awt.MenuComponent`

Declaration

```
public boolean postEvent(Event evt)
```

Description

Posts the specified event to the menu.

Parameters:

evt - the event which is to take place

paramString

Applies to

Class `java.awt.MenuComponent`

Declaration

```
protected String paramString()
```

Description

Returns the String parameter of this MenuComponent.

toString

Applies to

Class `java.awt.MenuComponent`

Declaration

```
public String toString()
```

Description

Returns the String representation of this MenuComponent's values.

Overrides:

toString in class Object



Class `java.awt.MenuBar`

`java.lang.Object`

|

+----`java.awt.MenuComponent`

|

+----`java.awt.MenuBar`

Declaration

```
public class MenuBar
```

```
    extends MenuComponent
```

```
    implements MenuContainer
```

Description

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`

Constructors

[MenuBar\(\)](#)

Methods

[add\(Menu\)](#)

[addNotify\(\)](#)

[countMenus\(\)](#)

[getHelpMenu\(\)](#)

[getMenu\(int\)](#)

[remove\(int\)](#)

[remove\(MenuComponent\)](#)

[removeNotify\(\)](#)

setHelpMenu(Menu)

MenuBar

Applies to

Class java.awt.MenuBar

Declaration

```
public MenuBar()
```

Description

Creates a new menu bar.

addNotify

Applies to

Class `java.awt.MenuBar`

Declaration

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

Description

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.

removeNotify

Applies to

Class `java.awt.MenuBar`

Declaration

```
public void removeNotify()
```

Description

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

Applies to

Class java.awt.MenuBar

Declaration

```
public Menu getHelpMenu()
```

Description

Gets the help menu on the menu bar.

setHelpMenu

Applies to

Class java.awt.MenuBar

Declaration

```
public synchronized void setHelpMenu(Menu m)
```

Description

Sets the help menu to the specified menu on the menu bar.

Parameters:

m - the menu to be set

add

Applies to

Class java.awt.MenuBar

Declaration

```
public synchronized Menu add(Menu m)
```

Description

Adds the specified menu to the menu bar.

Parameters:

m - the menu to be added to the menu bar

remove

Applies to

Class `java.awt.MenuBar`

Declaration

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

Description

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

public synchronized void remove(MenuComponent m)

Removes the specified menu from the menu bar.

Parameters:

m - the menu to be removed

countMenus

Applies to

Class java.awt.MenuBar

Declaration

```
public int countMenus()
```

Description

Counts the number of menus on the menu bar.

getMenu

Applies to

Class java.awt.MenuBar

Declaration

```
public Menu getMenu(int i)
```

Description

Gets the specified menu.

Parameters:

i - the menu to be returned

Class java.awt.MenuItem

java.lang.Object

|

+----java.awt.MenuComponent

|

+----java.awt.MenuItem

Declaration

```
public class MenuItem
```

```
    extends MenuComponent
```

Description

A String item that represents a choice in a menu.

Constructors

[MenuItem\(String\)](#)

Methods

[addNotify\(\)](#)

[disable\(\)](#)

[enable\(\)](#)

[enable\(boolean\)](#)

[getLabel\(\)](#)

[isEnabled\(\)](#)

[paramString\(\)](#)

[setLabel\(String\)](#)

MenuItem

Applies to

Class `java.awt.MenuItem`

Declaration

```
public MenuItem(String label)
```

Description

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.

addNotify

Applies to

Class `java.awt.MenuItem`

Declaration

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

Description

Creates the menu item's peer. The peer allows us to modify the appearance of the menu item without changing its functionality.

getLabel

Applies to

Class `java.awt.MenuItem`

Declaration

```
public String getLabel()
```

Description

Gets the label for this menu item.

setLabel

Applies to

Class java.awt.MenuItem

Declaration

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

Description

Sets the label to be the specified label.

Parameters:

label - the label for this menu item

isEnabled

Applies to

Class `java.awt.MenuItem`

Declaration

```
public boolean isEnabled()
```

Description

Checks whether the menu item is enabled.

enable

Applies to

Class `java.awt.MenuItem`

Declaration

```
public void enable()
```

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

Description

```
public void enable()
```

Makes this menu item selectable by the user.

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

Conditionally enables a component.

Parameters:

cond - enabled if true; disabled otherwise.

See Also:

enable, disable

disable

Applies to

Class `java.awt.MenuItem`

Declaration

```
public void disable()
```

Description

Makes this menu item unselectable by the user.

paramString

Applies to

Class `java.awt.MenuItem`

Declaration

```
public String paramString()
```

Description

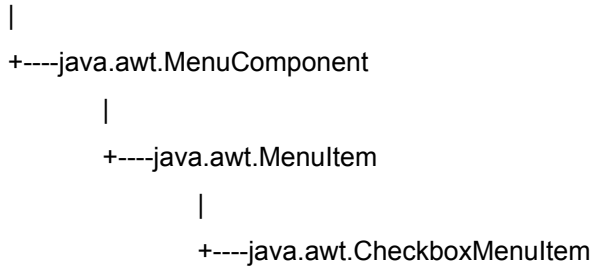
Returns the String parameter of the menu item.

Overrides:

paramString in class `MenuComponent`

Class java.awt.CheckboxMenuItem

java.lang.Object



Declaration

```
public class CheckboxMenuItem
    extends MenuItem
```

Description

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

Constructors

[CheckboxMenuItem\(String\)](#)

Methods

[addNotify\(\)](#)

[getState\(\)](#)

[paramString\(\)](#)

[setState\(boolean\)](#)

CheckboxMenuItem

Applies to

Class `java.awt.CheckboxMenuItem`

Declaration

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

Description

Creates the checkbox item with the specified label.

Parameters:

label - the button label

addNotify

Applies to

Class `java.awt.CheckboxMenuItem`

Declaration

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

Description

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`

getState

Applies to

Class `java.awt.CheckboxMenuItem`

Declaration

```
public boolean getState()
```

Description

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

setState

Applies to

Class `java.awt.CheckboxMenuItem`

Declaration

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

Description

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

Parameters:

t - the specified state of the checkbox

paramString

Applies to

Class `java.awt.CheckboxMenuItem`

Declaration

```
public String paramString()
```

Description

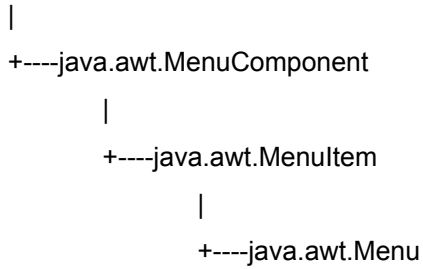
Returns the parameter String of this button.

Overrides:

paramString in class `MenuItem`

Class java.awt.Menu

java.lang.Object



Declaration

```
public class Menu
    extends MenuItem
    implements MenuContainer
```

Description

A Menu that is a component of a menu bar.

Constructors

[Menu\(String\)](#)
[Menu\(String, boolean\)](#)

Methods

[add\(MenuItem\)](#)
[add\(String\)](#)
[addNotify\(\)](#)
[addSeparator\(\)](#)
[countItems\(\)](#)
[getItem\(int\)](#)
[isTearOff\(\)](#)
[remove\(int\)](#)
[remove\(MenuComponent\)](#)

removeNotify()

Menu

Applies to

Class `java.awt.Menu`

Declaration

```
public Menu(String label)
public Menu(String label, boolean tearOff)
```

Description

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

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

Applies to

Class java.awt.Menu

Declaration

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

Description

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

Applies to

Class `java.awt.Menu`

Declaration

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

Description

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

Applies to

Class `java.awt.Menu`

Declaration

```
public boolean isTearOff()
```

Description

Returns true if this is a tear-off menu.

countItems

Applies to

Class `java.awt.Menu`

Declaration

```
public int countItems()
```

Description

Returns the number of elements in this menu.

getItem

Applies to

Class java.awt.Menu

Declaration

```
public MenuItem getItem(int index)
```

Description

Returns the item located at the specified index of this menu.

Parameters:

index - the position of the item to be returned

add

Applies to

Class `java.awt.Menu`

Declaration

```
public synchronized MenuItem add(MenuItem mi)
public void add(String label)
```

Description

```
public synchronized MenuItem add(MenuItem mi)
```

Adds the specified item to this menu.

Parameters:

mi - the item to be added

```
public void add(String label)
```

Adds an item with with the specified label to this menu.

Parameters:

label - the text on the item

addSeparator

Applies to

Class `java.awt.Menu`

Declaration

```
public void addSeparator()
```

Description

Adds a separator line, or a hypen, to the menu at the current position.

remove

Applies to

Class `java.awt.Menu`

Declaration

```
public synchronized void remove(int index)  
public synchronized void remove(MenuComponent item)
```

Description

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

public synchronized void remove(MenuComponent item)

Deletes the specified item from this menu.

Parameters:

item - the item to be removed from the menu

Interface `java.awt.peer.MenuComponentPeer`

Declaration

```
public interface MenuComponentPeer  
    extends Object
```

Methods

`dispose()`

dispose

Applies to

Interface `java.awt.peer.MenuComponentPeer`

Declaration

```
public abstract void dispose()
```

Interface `java.awt.MenuContainer`

Declaration

```
public interface MenuContainer  
    extends Object
```

Description

The super class of all menu related containers.

Methods

[getFont\(\)](#)

[postEvent\(Event\)](#)

[remove\(MenuComponent\)](#)

getFont

Applies to

Interface java.awt.MenuContainer

Declaration

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

postEvent

Applies to

Interface java.awt.MenuContainer

Declaration

```
public abstract boolean postEvent(Event evt)
```

remove

Applies to

Interface `java.awt.MenuContainer`

Declaration

```
public abstract void remove(MenuComponent comp)
```

Interface `java.awt.peer.MenuItemPeer`

public interface MenuItemPeer

extends Object

extends MenuComponentPeer

Methods

`disable()`

`enable()`

`setLabel(String)`

setLabel

Applies to

Interface `java.awt.peer.MenuItemPeer`

Declaration

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

enable

Applies to

Interface `java.awt.peer.MenuItemPeer`

Declaration

```
public abstract void enable()
```

disable

Applies to

Interface `java.awt.peer.MenuItemPeer`

Declaration

```
public abstract void disable()
```

Interface `java.awt.peer.MenuPeer`

Declaration

```
public interface MenuPeer
    extends Object
    extends MenuItemPeer
```

Methods

```
addItem(MenuItem)  
addSeparator()  
delItem(int)
```

addSeparator

Applies to

Interface java.awt.peer.MenuPeer

Declaration

```
public abstract void addSeparator()
```

addItem

Applies to

Interface java.awt.peer.MenuPeer

Declaration

```
public abstract void addItem(Menuitem item)
```

delItem

Applies to

Interface java.awt.peer.MenuPeer

Declaration

```
public abstract void delItem(int index)
```

Class java.lang.Number

java.lang.Object

|

+----java.lang.Number

Declaration

```
public class Number
    extends Object
```

Description

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

Constructors

[Number\(\)](#)

Methods

[doubleValue\(\)](#)

[floatValue\(\)](#)

[intValue\(\)](#)

[longValue\(\)](#)

Number

Applies to

Class `java.lang.Number`

Declaration

```
public Number()
```

intValue

Applies to

Class `java.lang.Number`

Declaration

```
public abstract int intValue()
```

Description

Returns the value of the number as an int. This may involve rounding if the number is not already an integer.

longValue

Applies to

Class `java.lang.Number`

Declaration

```
public abstract long longValue()
```

Description

Returns the value of the number as a long. This may involve rounding if the number is not already a long.

floatValue

Applies to

Class `java.lang.Number`

Declaration

```
public abstract float floatValue()
```

Description

Returns the value of the number as a float. This may involve rounding if the number is not already a float.

doubleValue

Applies to

Class `java.lang.Number`

Declaration

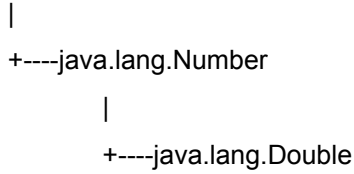
```
public abstract double doubleValue()
```

Description

Returns the value of the number as a double. This may involve rounding if the number is not already a double.

Class java.lang.Double

java.lang.Object



Declaration

```
public final class Double
    extends Number
```

Description

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.

Variables

MAX_VALUE

MIN_VALUE

NEGATIVE_INFINITY

NaN

POSITIVE_INFINITY

Constructors

Double(double)

Double(String)

Methods

doubleToLongBits(double)

doubleValue()

equals(Object)

floatValue()

hashCode()

intValue()

isInfinite(double)

isInfinite()

isNaN(double)

isNaN()

longBitsToDouble(long)

longValue()

toString(double)

toString()

valueOf(String)

POSITIVE_INFINITY

Applies to

Class `java.lang.Double`

Declaration

```
public final static double POSITIVE_INFINITY
```

Description

Positive infinity.

NEGATIVE_INFINITY

Applies to

Class `java.lang.Double`

Declaration

```
public final static double NEGATIVE_INFINITY
```

Description

Negative infinity.

NaN

Applies to

Class `java.lang.Double`

Declaration

```
public final static double NaN
```

Description

Not-a-Number. Note: is not equal to anything, including itself

MAX_VALUE

Applies to

Class `java.lang.Double`

Declaration

```
public final static double MAX_VALUE
```

Description

The maximum value a double can have. The greatest maximum value that a double can have is 1.79769313486231570e+308d.

MIN_VALUE

Applies to

Class `java.lang.Double`

Declaration

```
public final static double MIN_VALUE
```

Description

The minimum value a double can have. The lowest minimum value that a double can have is 4.94065645841246544e-324d.

Double

Applies to

Class `java.lang.Double`

Declaration

```
public Double(double value)
public Double(String s) throws NumberFormatException
```

Description

public Double(double value)

Constructs a Double wrapper for the specified double value.

Parameters:

value - the initial value of the 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.

toString

Applies to

Class `java.lang.Double`

Declaration

```
public static String toString(double d)
public String toString()
```

Description

public static String toString(double d)

Returns a String representation for the specified double value.

Parameters:

d - the double to be converted

public String toString()

Returns a String representation of this Double object.

Overrides:

toString in class Object

valueOf

Applies to

Class `java.lang.Double`

Declaration

```
public static Double valueOf(String s) throws NumberFormatException
```

Description

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

Applies to

Class `java.lang.Double`

Declaration

```
public static boolean isNaN(double v)  
public boolean isNaN()
```

Description

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

public boolean isNaN()

Returns true if this Double value is the special Not-a-Number (NaN) value.

isInfinite

Applies to

Class `java.lang.Double`

Declaration

```
public static boolean isInfinite(double v)  
public boolean isInfinite()
```

Description

public static boolean isInfinite(double v)

Returns true if the specified number is infinitely large in magnitude.

Parameters:

v - the value to be tested

public boolean isInfinite()

Returns true if this Double value is infinitely large in magnitude.

intValue

Applies to

Class `java.lang.Double`

Declaration

```
public int intValue()
```

Description

Returns the integer value of this Double (by casting to an int).

Overrides:

intValue in class Number

longValue

Applies to

Class `java.lang.Double`

Declaration

```
public long longValue()
```

Description

Returns the long value of this Double (by casting to a long).

Overrides:

longValue in class Number

floatValue

Applies to

Class `java.lang.Double`

Declaration

```
public float floatValue()
```

Description

Returns the float value of this Double.

Overrides:

floatValue in class `Number`

doubleValue

Applies to

Class `java.lang.Double`

Declaration

```
public double doubleValue()
```

Description

Returns the double value of this Double.

Overrides:

doubleValue in class Number

hashCode

Applies to

Class java.lang.Double

Declaration

```
public int hashCode()
```

Description

Returns a hashcode for this Double.

Overrides:

hashCode in class Object

equals

Applies to

Class java.lang.Double

Declaration

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

Description

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

Applies to

Class `java.lang.Double`

Declaration

```
public static long doubleToLongBits(double value)
```

Description

Returns the bit representation of a double-precision floating-point value

longBitsToDouble

Applies to

Class `java.lang.Double`

Declaration

```
public static double longBitsToDouble(long bits)
```

Description

Returns the double-precision floating-point value corresponding to the given long bit representation.

Class java.lang.Float

java.lang.Object

|

+----java.lang.Number

|

+----java.lang.Float

Declaration

```
public final class Float
    extends Number
```

Description

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.

Variables

MAX_VALUE

MIN_VALUE

NEGATIVE_INFINITY

NaN

POSITIVE_INFINITY

Constructors

Float(float)

Float(double)

Float(String)

Methods

doubleValue()

equals(Object)

floatToIntBits(float)

floatValue()

hashCode()

intBitsToFloat(int)

intValue()

isInfinite(float)

isInfinite()

isNaN(float)

isNaN()

longValue()

toString(float)

toString()

valueOf(String)

POSITIVE_INFINITY

Applies to

Class `java.lang.Float`

Declaration

```
public final static float POSITIVE_INFINITY
```

Description

Positive infinity.

NEGATIVE_INFINITY

Applies to

Class `java.lang.Float`

Declaration

```
public final static float NEGATIVE_INFINITY
```

Description

Negative infinity.

NaN

Applies to

Class `java.lang.Float`

Declaration

```
public final static float NaN
```

Description

Not-a-Number. Note: is not equal to anything, including itself

MAX_VALUE

Applies to

Class java.lang.Float

Declaration

```
public final static float MAX_VALUE
```

Description

The maximum value a float can have. The largest maximum value possible is 3.40282346638528860e+38.

MIN_VALUE

Applies to

Class `java.lang.Float`

Declaration

```
public final static float MIN_VALUE
```

Description

The minimum value a float can have. The lowest minimum value possible is 1.40129846432481707e-45.

Float

Applies to

Class `java.lang.Float`

Declaration

```
public Float(float value)
public Float(double value)
public Float(String s) throws NumberFormatException
```

Description

public Float(float value)

Constructs a Float wrapper for the specified float value.

Parameters:

value - the value of the Float

public Float(double value)

Constructs a Float wrapper for the specified double value.

Parameters:

value - the value of the 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.

toString

Applies to

Class java.lang.Float

Declaration

```
public static String toString(float f)
public String toString()
```

Description

```
public static String toString(float f)
```

Returns a String representation for the specified float value.

Parameters:

f - the float to be converted

```
public String toString()
```

Returns a String representation of this Float object.

Overrides:

toString in class Object

valueOf

Applies to

Class `java.lang.Float`

Declaration

```
public static Float valueOf(String s) throws NumberFormatException
```

Description

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.

isNaN

Applies to

Class `java.lang.Float`

Declaration

```
public static boolean isNaN(float v)  
public boolean isNaN()
```

Description

```
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

```
public boolean isNaN()
```

Returns true if this Float value is Not-a-Number (NaN).

isInfinite

Applies to

Class java.lang.Float

Declaration

```
public static boolean isInfinite(float v)  
public boolean isInfinite()
```

Description

```
public static boolean isInfinite(float v)
```

Returns true if the specified number is infinitely large in magnitude.

Parameters:

v - the value to be tested

```
public boolean isInfinite()
```

Returns true if this Float value is infinitely large in magnitude.

intValue

Applies to

Class `java.lang.Float`

Declaration

```
public int intValue()
```

Description

Returns the integer value of this Float (by casting to an int).

Overrides:

intValue in class Number

longValue

Applies to

Class `java.lang.Float`

Declaration

```
public long longValue()
```

Description

Returns the long value of this Float (by casting to a long).

Overrides:

longValue in class Number

floatValue

Applies to

Class `java.lang.Float`

Declaration

```
public float floatValue()
```

Description

Returns the float value of this Float object.

Overrides:

floatValue in class Number

doubleValue

Applies to

Class java.lang.Float

Declaration

```
public double doubleValue()
```

Description

Returns the double value of this Float.

Overrides:

doubleValue in class Number

hashCode

Applies to

Class java.lang.Float

Declaration

```
public int hashCode()
```

Description

Returns a hashcode for this Float.

Overrides:

hashCode in class Object

equals

Applies to

Class java.lang.Float

Declaration

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

Description

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

floatToIntBits

Applies to

Class `java.lang.Float`

Declaration

```
public static int floatToIntBits(float value)
```

Description

Returns the bit representation of a single-float value

intBitsToFloat

Applies to

Class `java.lang.Float`

Declaration

```
public static float intBitsToFloat(int bits)
```

Description

Returns the single-float corresponding to a given bit representation.

Class java.lang.Integer

java.lang.Object

|

+----java.lang.Number

|

+----java.lang.Integer

Declaration

```
public final class Integer
    extends Number
```

Description

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.

Variables

MAX_VALUE

MIN_VALUE

Constructors

Integer(int)

Integer(String)

Methods

doubleValue()

equals(Object)

floatValue()

getInteger(String)

getInteger(String, int)

getInteger(String, Integer)

hashCode()

intValue()

longValue()

parseInt(String, int)

parseInt(String)

toString(int, int)

toString(int)

toString()

valueOf(String, int)

valueOf(String)

MIN_VALUE

Applies to

Class `java.lang.Integer`

Declaration

```
public final static int MIN_VALUE
```

Description

The minimum value an Integer can have. The lowest minimum value an Integer can have is 0x80000000.

MAX_VALUE

Applies to

Class `java.lang.Integer`

Declaration

```
public final static int MAX_VALUE
```

Description

The maximum value an Integer can have. The greatest maximum value an Integer can have is 0x7ffffff.

Integer

Applies to

Class java.lang.Integer

Declaration

```
public Integer(int value)
public Integer(String s) throws NumberFormatException
```

Description

public Integer(int value)

Constructs an Integer object initialized to the specified int value.

Parameters:

value - the initial value of the 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.

toString

Applies to

Class java.lang.Integer

Declaration

```
public static String toString(int i, int radix)
public static String toString(int i)
public String toString()
```

Description

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

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

public String toString()

Returns a String object representing this Integer's value.

Overrides:

toString in class Object

parseInt

Applies to

Class `java.lang.Integer`

Declaration

```
public static int parseInt(String s, int radix) throws NumberFormatException  
public static int parseInt(String s) throws NumberFormatException
```

Description

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.

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

Applies to

Class `java.lang.Integer`

Declaration

```
public static Integer valueOf(String s, int radix) throws NumberFormatException  
public static Integer valueOf(String s) throws NumberFormatException
```

Description

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.

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

Applies to

Class java.lang.Integer

Declaration

```
public int intValue()
```

Description

Returns the value of this Integer as an int.

Overrides:

intValue in class Number

longValue

Applies to

Class `java.lang.Integer`

Declaration

```
public long longValue()
```

Description

Returns the value of this Integer as a long.

Overrides:

longValue in class Number

floatValue

Applies to

Class java.lang.Integer

Declaration

```
public float floatValue()
```

Description

Returns the value of this Integer as a float.

Overrides:

floatValue in class Number

doubleValue

Applies to

Class `java.lang.Integer`

Declaration

```
public double doubleValue()
```

Description

Returns the value of this Integer as a double.

Overrides:

doubleValue in class Number

hashCode

Applies to

Class java.lang.Integer

Declaration

```
public int hashCode()
```

Description

Returns a hashcode for this Integer.

Overrides:

hashCode in class Object

equals

Applies to

Class java.lang.Integer

Declaration

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

Description

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

Applies to

Class `java.lang.Integer`

Declaration

```
public static Integer getInteger(String nm)
public static Integer getInteger(String nm, int val)
public static Integer getInteger(String nm, Integer val)
```

Description

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

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

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.Long

java.lang.Object

|

+----java.lang.Number

|

+----java.lang.Long

Declaration

```
public final class Long
    extends Number
```

Description

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.

Variables

MAX_VALUE

MIN_VALUE

Constructors

Long(long)

Long(String)

Methods

doubleValue()

equals(Object)

floatValue()

getLong(String)

getLong(String, long)

getLong(String, Long)

hashCode()

intValue()

longValue()

parseLong(String, int)

parseLong(String)

toString(long, int)

toString(long)

toString()

valueOf(String, int)

valueOf(String)

MIN_VALUE

Applies to

Class `java.lang.Long`

Declaration

```
public final static long MIN_VALUE
```

Description

The minimum value a Long can have. The lowest minimum value that a Long can have is 0x8000000000000000.

MAX_VALUE

Applies to

Class `java.lang.Long`

Declaration

```
public final static long MAX_VALUE
```

Description

The maximum value a Long can have. The largest maximum value that a Long can have is 0x7fffffffffffffff.

Long

Applies to

Class `java.lang.Long`

Declaration

```
public Long(long value)
public Long(String s) throws NumberFormatException
```

Description

public Long(long value)

Constructs a Long object initialized to the specified value.

Parameters:

value - the initial value of the 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

Applies to

Class `java.lang.Long`

Declaration

```
public static String toString(long i, int radix)
public static String toString(long i)
public String toString()
```

Description

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

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

public String toString()

Returns a String object representing this Long's value.

Overrides:

toString in class Object

parseLong

Applies to

Class `java.lang.Long`

Declaration

```
public static long parseLong(String s, int radix) throws NumberFormatException  
public static long parseLong(String s) throws NumberFormatException
```

Description

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.

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

Applies to

Class `java.lang.Long`

Declaration

```
public static Long valueOf(String s, int radix) throws NumberFormatException  
public static Long valueOf(String s) throws NumberFormatException
```

Description

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.

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

Applies to

Class java.lang.Long

Declaration

```
public int intValue()
```

Description

Returns the value of this Long as an int.

Overrides:

intValue in class Number

longValue

Applies to

Class `java.lang.Long`

Declaration

```
public long longValue()
```

Description

Returns the value of this Long as a long.

Overrides:

longValue in class Number

floatValue

Applies to

Class `java.lang.Long`

Declaration

```
public float floatValue()
```

Description

Returns the value of this Long as a float.

Overrides:

floatValue in class Number

doubleValue

Applies to

Class `java.lang.Long`

Declaration

```
public double doubleValue()
```

Description

Returns the value of this Long as a double.

Overrides:

doubleValue in class Number

hashCode

Applies to

Class java.lang.Long

Declaration

```
public int hashCode()
```

Description

Computes a hashcode for this Long.

Overrides:

hashCode in class Object

equals

Applies to

Class java.lang.Long

Declaration

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

Description

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

Applies to

Class `java.lang.Long`

Declaration

```
public static Long getLong(String nm)
public static Long getLong(String nm, long val)
public static Long getLong(String nm, Long val)
```

Description

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

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

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.util.Observable

java.lang.Object

|

+----java.util.Observable

Declaration

```
public class Observable
    extends Object
```

Description

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.

Constructors

Observable()

Methods

addObserver(Observer)

clearChanged()

countObservers()

deleteObserver(Observer)

deleteObservers()

hasChanged()

notifyObservers()

notifyObservers(Object)

setChanged()

Observable

Applies to

Class `java.util.Observable`

Declaration

```
public Observable()
```

addObserver

Applies to

Class java.util.Observable

Declaration

```
public synchronized void addObserver(Observer o)
```

Description

Adds an observer to the observer list.

Parameters:

o - the observer to be added

deleteObserver

Applies to

Class `java.util.Observable`

Declaration

```
public synchronized void deleteObserver(Observer o)
```

Description

Deletes an observer from the observer list.

Parameters:

o - the observer to be deleted

notifyObservers

Applies to

Class java.util.Observable

Declaration

```
public void notifyObservers()  
public synchronized void notifyObservers(Object arg)
```

Description

Notifies all observers if an observable change occurs.

public synchronized void notifyObservers(Object arg)

Notifies all observers of the specified observable change which occurred.

Parameters:

arg - what is being notified

deleteObservers

Applies to

Class `java.util.Observable`

Declaration

```
public synchronized void deleteObservers()
```

Description

Deletes observers from the observer list.

setChanged

Applies to

Class `java.util.Observable`

Declaration

```
protected synchronized void setChanged()
```

Description

Sets a flag to note an observable change.

clearChanged

Applies to

Class `java.util.Observable`

Declaration

```
protected synchronized void clearChanged()
```

Description

Clears an observable change.

hasChanged

Applies to

Class `java.util.Observable`

Declaration

```
public synchronized boolean hasChanged()
```

Description

Returns a true boolean if an observable change has occurred.

countObservers

Applies to

Class `java.util.Observable`

Declaration

```
public synchronized int countObservers()
```

Description

Counts the number of observers.

Interface `java.util.Observer`

Declaration

```
public interface Observer  
    extends Object
```

Description

When implemented, this interface allows all classes to be observable by instances of class `Observer`.

Methods

`update(Observable, Object)`

update

Applies to

Interface java.util.Observer

Declaration

```
public abstract void update(Observable o, Object arg)
```

Description

Called when observers in the observable list need to be updated.

Parameters:

o - the list of observers

arg - the argument being notified

Class java.io.OutputStream

java.lang.Object

|

+----java.io.OutputStream

Declaration

```
public class OutputStream
    extends Object
```

Description

Abstract class representing an output stream of bytes. All OutputStreams are based on this class.

See Also:

InputStream, FilterOutputStream, BufferedOutputStream, DataOutputStream , ByteArrayOutputStream

Constructors

[OutputStream\(\)](#)

Methods

[close\(\)](#)

[flush\(\)](#)

[write\(int\)](#)

[write\(byte\[\]\)](#)

[write\(byte\[\], int, int\)](#)

OutputStream

Applies to

Class java.io.OutputStream

Declaration

```
public OutputStream()
```


write

Applies to

Declaration

```
public abstract void write(int b) throws IOException  
public void write(byte b[]) throws IOException  
public void write(byte b[], int off, int len) throws IOException
```

Description

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.

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.

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

Applies to

Class java.io.OutputStream

Declaration

```
public void flush() throws IOException
```

Description

Flushes the stream. This will write any buffered output bytes.

Throws: IOException

If an I/O error has occurred.

close

Applies to

Class java.io.OutputStream

Declaration

```
public void close() throws IOException
```

Description

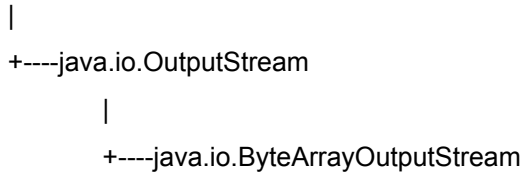
Closes the stream. This method must be called to release any resources associated with the stream.

Throws: IOException

If an I/O error has occurred.

Class java.io.ByteArrayOutputStream

java.lang.Object



Declaration

```
public class ByteArrayOutputStream
    extends OutputStream
```

Description

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()`.

Variables

buf
count

Constructors

[ByteArrayOutputStream\(\)](#)
[ByteArrayOutputStream\(int\)](#)

Methods

[reset\(\)](#)
[size\(\)](#)
[toByteArray\(\)](#)
[toString\(\)](#)
[toString\(int\)](#)
[write\(int\)](#)
[write\(byte\[\], int, int\)](#)

writeTo(OutputStream)

buf

Applies to

Class `java.io.ByteArrayOutputStream`

Declaration

protected byte buf[]

Description

The buffer where data is stored.

count

Applies to

Class `java.io.ByteArrayOutputStream`

Declaration

protected int count

Description

The number of bytes in the buffer.

ByteArrayOutputStream

Applies to

Class `java.io.ByteArrayOutputStream`

Declaration

```
public ByteArrayOutputStream()  
public ByteArrayOutputStream(int size)
```

Description

public ByteArrayOutputStream()

Creates a new ByteArrayOutputStream.

public ByteArrayOutputStream(int size)

Creates a new ByteArrayOutputStream with the specified initial size.

Parameters:

size - the initial size

write

Applies to

Class `java.io.ByteArrayOutputStream`

Declaration

```
public synchronized void write(int b)
public synchronized void write(byte b[], int off, int len)
```

Description

public synchronized void write(int b)

Writes a byte to the buffer.

Parameters:

b - the byte

Overrides:

write in class `OutputStream`

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

Applies to

Class `java.io.ByteArrayOutputStream`

Declaration

```
public synchronized void writeTo(OutputStream out) throws IOException
```

Description

Writes the contents of the buffer to another stream.

Parameters:

out - the output stream to write to

reset

Applies to

Class `java.io.ByteArrayOutputStream`

Declaration

```
public synchronized void reset()
```

Description

Resets the buffer so that you can use it again without throwing away the already allocated buffer.

toByteArray

Applies to

Class `java.io.ByteArrayOutputStream`

Declaration

```
public synchronized byte[] toByteArray()
```

Description

Returns a copy of the input data.

size

Applies to

Class `java.io.ByteArrayOutputStream`

Declaration

```
public int size()
```

Description

Returns the current size of the buffer.

toString

Applies to

Class `java.io.ByteArrayOutputStream`

Declaration

```
public String toString()  
public String toString(int hibyte)
```

Description

public String toString()

Converts input data to a string.

Returns:

the string.

Overrides:

toString in class Object

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

Class java.io.FileOutputStream

java.lang.Object

```
|
+----java.io.OutputStream
      |
      +----java.io.FileOutputStream
```

Declaration

```
public class FileOutputStream
    extends OutputStream
```

Description

File output stream, can be constructed from a file descriptor or a file name.

See Also:

FileInputStream, File

Constructors

[FileOutputStream\(String\)](#)

[FileOutputStream\(File\)](#)

[FileOutputStream\(FileDescriptor\)](#)

Methods

[close\(\)](#)

[finalize\(\)](#)

[getFD\(\)](#)

[write\(int\)](#)

[write\(byte\[\]\)](#)

[write\(byte\[\], int, int\)](#)

FileOutputStream

Applies to

Class java.io.FileOutputStream

Declaration

```
public FileOutputStream(String name) throws IOException  
public FileOutputStream(File file) throws IOException  
public FileOutputStream(FileDescriptor fdObj)
```

Description

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.

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.

public FileOutputStream(FileDescriptor fdObj)

write

Applies to

Class `java.io.FileOutputStream`

Declaration

```
public void write(int b) throws IOException  
public void write(byte b[]) throws IOException  
public void write(byte b[], int off, int len) throws IOException
```

Description

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

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

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

Applies to

Class java.io.FileOutputStream

Declaration

public void close() throws IOException

Description

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

Applies to

Class `java.io.FileOutputStream`

Declaration

```
public final FileDescriptor getFD() throws IOException
```

Description

Returns the file descriptor associated with this stream.

Returns:

the file descriptor.

finalize

Applies to

Class java.io.FileOutputStream

Declaration

protected void finalize() throws IOException

Description

Closes the stream when garbage is collected.

Overrides:

finalize in class Object

Class java.io.FilterOutputStream

java.lang.Object



Declaration

```
public class FilterOutputStream
    extends OutputStream
```

Description

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.

Variables

out

Constructors

[FilterOutputStream\(OutputStream\)](#)

Methods

[close\(\)](#)

[flush\(\)](#)

[write\(int\)](#)

[write\(byte\[\]\)](#)

[write\(byte\[\], int, int\)](#)

out

Applies to

Class `java.io.FilterOutputStream`

Declaration

protected OutputStream out

Description

The actual output stream.

FilterOutputStream

Applies to

Class `java.io.FilterOutputStream`

Declaration

```
public FilterOutputStream(OutputStream out)
```

Description

Creates an output stream filter.

Parameters:

out - the output stream

write

Applies to

Class `java.io.FilterOutputStream`

Declaration

```
public void write(int b) throws IOException  
public void write(byte b[]) throws IOException  
public void write(byte b[], int off, int len) throws IOException
```

Description

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

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

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

Applies to

Class `java.io.FilterOutputStream`

Declaration

```
public void flush() throws IOException
```

Description

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

Applies to

Class `java.io.FilterOutputStream`

Declaration

`public void close() throws IOException`

Description

Closes the stream. This method must be called to release any resources associated with the stream.

Throws: `IOException`

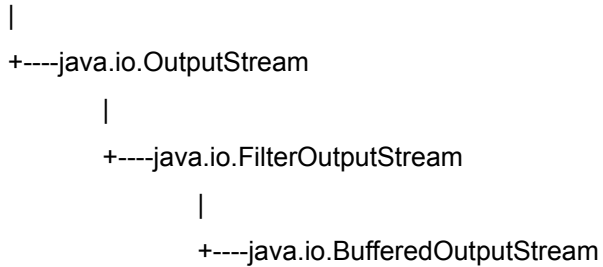
If an I/O error has occurred.

Overrides:

`close` in class `OutputStream`

Class java.io.BufferedOutputStream

java.lang.Object



Declaration

```
public class BufferedOutputStream
    extends FilterOutputStream
```

Description

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.

Variables

buf
count

Constructors

[BufferedOutputStream\(OutputStream\)](#)
[BufferedOutputStream\(OutputStream, int\)](#)

Methods

[flush\(\)](#)
[write\(int\)](#)
[write\(byte\[\], int, int\)](#)

buf

Applies to

java.io.BufferedOutputStream

Declaration

protected byte buf[]

Description

The buffer where data is stored.

count

Applies to

java.io.BufferedOutputStream

Declaration

protected int count

Description

The number of bytes in the buffer.

BufferedOutputStream

Applies to

[java.io.BufferedOutputStream](#)

Declaration

```
public BufferedOutputStream(OutputStream out)
public BufferedOutputStream(OutputStream out, int size)
```

Description

```
public BufferedOutputStream(OutputStream out)
```

Creates a new buffered stream with a default buffer size.

Parameters:

out - the output stream

```
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

write

Applies to

[java.io.BufferedOutputStream](#)

Declaration

```
public synchronized void write(int b) throws IOException
public synchronized void write(byte b[],
                                int off,
                                int len) throws IOException
```

Description

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`

**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

Applies to

[java.io.BufferedOutputStream](#)

Declaration

```
public synchronized void flush() throws IOException
```

Description

Flushes the stream. This will write any buffered output bytes.

Throws: `IOException`

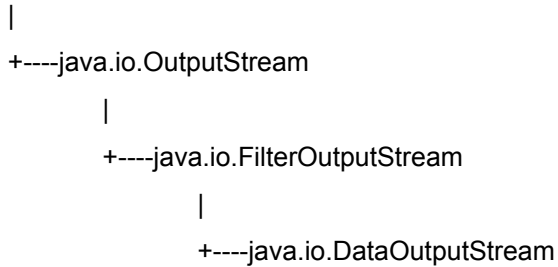
If an I/O error has occurred.

Overrides:

flush in class `FilterOutputStream`

Class java.io.DataOutputStream

java.lang.Object



Declaration

```
public class DataOutputStream
    extends FilterOutputStream
    implements DataOutput
```

Description

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`.

Variables

written

Constructors

DataOutputStream(OutputStream)

Methods

flush()

size()

write(int)

write(byte[], int, int)

writeBoolean(boolean)

writeByte(int)

writeBytes(String)

writeChar(int)

writeChars(String)

writeDouble(double)

writeFloat(float)

writeInt(int)

writeLong(long)

writeShort(int)

writeUTF(String)

written

Applies to

Class `java.io.DataOutputStream`

Declaration

protected int written

Description

The number of bytes written so far.

DataOutputStream

Applies to

Class `java.io.DataOutputStream`

Declaration

```
public DataOutputStream(OutputStream out)
```

Description

Creates a new `DataOutputStream`.

Parameters:

out - the output stream

write

Applies to

Class `java.io.DataOutputStream`

Declaration

```
public synchronized void write(int b) throws IOException
public synchronized void write(byte b[],
                                int off,
                                int len) throws IOException
```

Description

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`

```
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

Applies to

Class `java.io.DataOutputStream`

Declaration

```
public void flush() throws IOException
```

Description

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

Applies to

Class `java.io.DataOutputStream`

Declaration

```
public final void writeBoolean(boolean v) throws IOException
```

Description

Writes a boolean.

Parameters:

v - the boolean to be written

writeByte

Applies to

Class `java.io.DataOutputStream`

Declaration

```
public final void writeByte(int v) throws IOException
```

Description

Writes an 8 bit byte.

Parameters:

v - the byte value to be written

writeShort

Applies to

Class `java.io.DataOutputStream`

Declaration

```
public final void writeShort(int v) throws IOException
```

Description

Writes a 16 bit short.

Parameters:

v - the short value to be written

writeChar

Applies to

Class `java.io.DataOutputStream`

Declaration

```
public final void writeChar(int v) throws IOException
```

Description

Writes a 16 bit char.

Parameters:

v - the char value to be written

writeInt

Applies to

Class `java.io.DataOutputStream`

Declaration

```
public final void writeInt(int v) throws IOException
```

Description

Writes a 32 bit int.

Parameters:

v - the integer value to be written

writeLong

Applies to

Class `java.io.DataOutputStream`

Declaration

```
public final void writeLong(long v) throws IOException
```

Description

Writes a 64 bit long.

Parameters:

v - the long value to be written

writeFloat

Applies to

Class `java.io.DataOutputStream`

Declaration

```
public final void writeFloat(float v) throws IOException
```

Description

Writes a 32 bit float.

Parameters:

v - the float value to be written

writeDouble

Applies to

Class `java.io.DataOutputStream`

Declaration

```
public final void writeDouble(double v) throws IOException
```

Description

Writes a 64 bit double.

Parameters:

v - the double value to be written

writeBytes

Applies to

Class `java.io.DataOutputStream`

Declaration

```
public final void writeBytes(String s) throws IOException
```

Description

Writes a String as a sequence of bytes.

Parameters:

s - the String of bytes to be written

writeChars

Applies to

Class `java.io.DataOutputStream`

Declaration

```
public final void writeChars(String s) throws IOException
```

Description

Writes a String as a sequence of chars.

Parameters:

s - the String of chars to be written

writeUTF

Applies to

Class `java.io.DataOutputStream`

Declaration

```
public final void writeUTF(String str) throws IOException
```

Description

Writes a String in UTF format.

Parameters:

str - the String in UTF format

size

Applies to

Class `java.io.DataOutputStream`

Declaration

```
public final int size()
```

Description

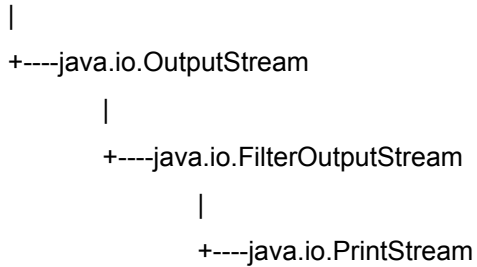
Returns the number of bytes written.

Returns:

the number of bytes written thus far.

Class java.io.PrintStream

java.lang.Object



Declaration

```
public class PrintStream
    extends FilterOutputStream
```

Description

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);
```

Constructors

[PrintStream\(OutputStream\)](#)

[PrintStream\(OutputStream, boolean\)](#)

Methods

checkError()

close()

flush()

print(Object)

print(String)

print(char[])

print(char)

print(int)

print(long)

print(float)

print(double)

print(boolean)

println()

println(Object)

println(String)

println(char[])

println(char)

println(int)

println(long)

println(float)

println(double)

println(boolean)

write(int)

write(byte[], int, int)

PrintStream

```
public PrintStream(OutputStream out)
```

Creates a new PrintStream.

Parameters:

out - the output stream

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

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

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

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

`println`

```
public synchronized void println(long l)
```

Prints a long followed by a newline.

Parameters:

l - the long to be printed

println

```
public synchronized void println(float f)
```

Prints a float followed by a newline.

Parameters:

f - the float to be printed

println

```
public synchronized void println(double d)
```

Prints a double followed by a newline.

Parameters:

d - the double to be printed

`println`

```
public synchronized void println(boolean b)
```

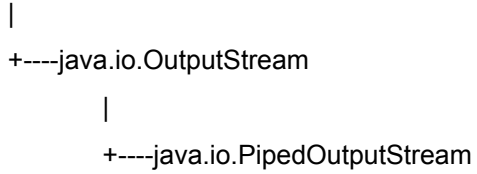
Prints a boolean followed by a newline.

Parameters:

b - the boolean to be printed

Class java.io.PipedOutputStream

java.lang.Object



Declaration

```
public class PipedOutputStream
    extends OutputStream
```

Description

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

Constructors

```
PipedOutputStream(PipedInputStream)
PipedOutputStream()
```

Methods

```
close()
connect(PipedInputStream)
write(int)
write(byte[], int, int)
```

PipedOutputStream

```
public PipedOutputStream(PipedInputStream snk) throws IOException
```

Creates an output file connected to the specified PipedInputStream.

Parameters:

snk - The InputStream to connect to.

PipedOutputStream

```
public PipedOutputStream()
```

Creates an output file that isn't connected to anything (yet). It must be connected before being used.

connect

```
public void connect(PipedInputStream snk) throws IOException
```

Connect this output stream to a receiver.

Parameters:

snk - The InputStream to connect to.

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

Interface `java.awt.peer.PanelPeer`

`public interface PanelPeer`

`extends Object`

`extends ContainerPeer`

Class java.awt.image.PixelGrabber

java.lang.Object

|

+----java.awt.image.PixelGrabber

Declaration

```
public class PixelGrabber
    extends Object
    implements ImageConsumer
```

Description

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
```

Methods

PixelGrabber(Image, int, int, int, int, int[], int, int)
PixelGrabber(ImageProducer, int, int, int, int, int[], int, int)
grabPixels()
grabPixels(long)
imageComplete(int)
setColorModel(ColorModel)
setDimensions(int, int)
setHints(int)
setPixels(int, int, int, int, ColorModel, byte[], int, int)
setPixels(int, int, int, int, ColorModel, int[], int, int)
setProperties(Hashtable)
status()

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

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 $\text{pix}[(j - y) * \text{scansize} + (i - x) + \text{off}]$.

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

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.

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.

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

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.

setHints

```
public void setHints(int hints)
```

The setHints method is part of the ImageConsumer API which this class must implement to retrieve the pixels.

setProperties

```
public void setProperties(Hashtable props)
```

The setProperties method is part of the ImageConsumer API which this class must implement to retrieve the pixels.

setColorModel

```
public void setColorModel(ColorModel model)
```

The setColorModel 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,  
                     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.

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.Point

java.lang.Object

|

+----java.awt.Point

Declaration

```
public class Point
    extends Object
```

Description

An x,y coordinate.

Variables

x

y

Constructors

Point(int, int)

Methods

equals(Object)

hashCode()

move(int, int)

toString()

translate(int, int)

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

java.lang.Object

|

+----java.awt.Polygon

Declaration

```
public class Polygon
    extends Object
```

Description

A polygon consists of a list of x and y coordinates.

Variables

npoints

xpoints

ypoints

Constructors

Polygon()

Polygon(int[], int[], int)

Methods

addPoint(int, int)

getBoundingBox()

inside(int, int)

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.lang.Process

java.lang.Object

|

+----java.lang.Process

Declaration

```
public class Process
    extends Object
```

Description

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 stdin and/or stdout 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.

Constructors

```
Process()
```

Methods

```
destroy()
exitValue()
getErrorStream()
getInputStream()
getOutputStream()
waitFor()
```

Process

```
public Process()
```

getOutputStream

```
public abstract OutputStream getOutputStream()
```

Returns a Stream connected to the input of the child process. This stream is traditionally buffered.

getInputStream

```
public abstract InputStream getInputStream()
```

Returns a Stream connected to the output of the child process. This stream is traditionally buffered.

getErrorStream

```
public abstract InputStream getErrorStream()
```

Returns the an InputStream connected to the error stream of the child process. This stream is traditionally unbuffered.

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: IllegalStateException

If the subprocess has not yet terminated.

destroy

public abstract void destroy()

Kills the subprocess.

Class java.util.Random

```
java.lang.Object
|
+----java.util.Random
```

Declaration

```
public class Random
    extends Object
```

Description

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

Constructors

Random()

Random(long)

Methods

nextDouble()

nextFloat()

nextGaussian()

nextInt()

nextLong()

setSeed(long)

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

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

nextInt

```
public int nextInt()
```

Generates a pseudorandom uniformly distributed int value.

Returns:

an integer value.

nextLong

```
public long nextLong()
```

Generate a pseudorandom uniformly distributed long value.

Returns:

A long integer value

nextFloat

```
public float nextFloat()
```

Generates a pseudorandom uniformly 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 uniformly 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.io.RandomAccessFile

java.lang.Object

|

+----java.io.RandomAccessFile

Declaration

```
public class RandomAccessFile
    extends Object
    implements DataOutput, DataInput
```

Description

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.

Constructors

```
RandomAccessFile(String, String)
RandomAccessFile(File, String)
```

Methods

```
close()
getFD()
getFilePointer()
length()
read()
read(byte[], int, int)
read(byte[])
readBoolean()
readByte()
readChar()
readDouble()
```

readFloat()
readFully(byte[])
readFully(byte[], int, int)
readInt()
readLine()
readLong()
readShort()
readUTF()
readUnsignedByte()
readUnsignedShort()
seek(long)
skipBytes(int)
write(int)
write(byte[])
write(byte[], int, int)
writeBoolean(boolean)
writeByte(int)
writeBytes(String)
writeChar(int)
writeChars(String)
writeDouble(double)
writeFloat(float)
writeInt(int)
writeLong(long)
writeShort(int)
writeUTF(String)

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
```

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.

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

writeLong

public final void writeLong(long v) throws IOException

Writes a long.

Parameters:

v - the long

writeFloat

public final void writeFloat(float v) throws IOException

writeDouble

public final void writeDouble(double v) throws IOException

writeBytes

public final void writeBytes(String s) throws IOException

Writes a String as a sequence of bytes.

Parameters:

s - the String

writeChars

```
public final void writeChars(String s) throws IOException
```

Writes a String as a sequence of chars.

Parameters:

s - the String

writeUTF

```
public final void writeUTF(String str) throws IOException
```

Writes a String in UTF format.

Parameters:

str - the String

Class java.awt.Rectangle

java.lang.Object

|

+----java.awt.Rectangle

Declaration

```
public class Rectangle
    extends Object
```

Description

A rectangle defined by x, y, width and height.

Variables

height

width

x

y

Constructors

Rectangle()

Rectangle(int, int, int, int)

Rectangle(int, int)

Rectangle(Point, Dimension)

Rectangle(Point)

Rectangle(Dimension)

Methods

add(int, int)

add(Point)

add(Rectangle)

equals(Object)

grow(int, int)
hashCode()
inside(int, int)
intersection(Rectangle)
intersects(Rectangle)
isEmpty()
move(int, int)
reshape(int, int, int, int)
resize(int, int)
toString()
translate(int, int)
union(Rectangle)

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.

height

```
public int height
```

The height of the rectangle.

Rectangle

```
public Rectangle()
```

Constructs a new rectangle.

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

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

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

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

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

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 sun.tools.debug.RemoteDebugger

java.lang.Object

|

+----sun.tools.debug.RemoteDebugger

Declaration

```
public class RemoteDebugger
    extends Object
```

Description

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.

Constructors

```
RemoteDebugger(String, String, DebuggerCallback, boolean)
RemoteDebugger(String, DebuggerCallback, boolean)
```

Methods

```
close()
findClass(String)
freeMemory()
gc(RemoteObject[])
get(Integer)
getExceptionCatchList()
getSourcePath()
itrace(boolean)
listBreakpoints()
listClasses()
listThreadGroups(RemoteThreadGroup)
run(int, String[])
setSourcePath(String)
```

totalMemory()

trace(boolean)

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

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.

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

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

totalMemory

public int totalMemory() throws Exception

Report the total memory usage of the Java interpreter being debugged.

freeMemory

```
public int freeMemory() throws Exception
```

Report the free memory available to the Java interpreter being debugged.

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

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".

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.

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.

setSourcePath

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.RemoteValue

java.lang.Object

|

+----sun.tools.debug.RemoteValue

Declaration

```
public class RemoteValue
    extends Object
    implements AgentConstants
```

Description

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

Methods

```
description()
fromHex(String)
getType()
isObject()
toHex(int)
typeName()
```

getType

```
public final int getType()
```

Returns the RemoteValue's type.

isObject

```
public final boolean isObject()
```

Returns whether the RemoteValue is an Object (as opposed to a primitive type, such as int).

typeName

```
public abstract String typeName() throws Exception
```

Returns the RemoteValue's type as a string.

description

```
public String description()
```

Return a description of the RemoteValue.

toHex

```
public static String toHex(int n)
```

Convert an int to a hexadecimal string.

fromHex

```
public static int fromHex(String hexStr)
```

Convert hexadecimal strings to ints.

Class sun.tools.debug.RemoteBoolean

java.lang.Object

```
|
+----sun.tools.debug.RemoteValue
    |
    +----sun.tools.debug.RemoteBoolean
```

Declaration

```
public class RemoteBoolean
    extends RemoteValue
```

Description

The RemoteBoolean class extends RemoteValue for booleans.

See Also:

RemoteValue, RemoteDebugger

Methods

```
get()
toString()
typeName()
```

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

java.lang.Object

```
|
+----sun.tools.debug.RemoteValue
|
+----sun.tools.debug.RemoteByte
```

Declaration

```
public class RemoteByte
    extends RemoteValue
```

Description

The RemoteByte class extends RemoteValue for bytes.

See Also:

RemoteValue, RemoteDebugger

Methods

```
get()
toString()
typeName()
```

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

java.lang.Object

```
|
+----sun.tools.debug.RemoteValue
    |
    +----sun.tools.debug.RemoteChar
```

Declaration

```
public class RemoteChar
    extends RemoteValue
```

Description

The RemoteChar class extends RemoteValue for chars.

See Also:

RemoteValue, RemoteDebugger

Methods

```
get()
toString()
typeName()
```

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.RemoteDouble

java.lang.Object

```
|
+----sun.tools.debug.RemoteValue
    |
    +----sun.tools.debug.RemoteDouble
```

Declaration

```
public class RemoteDouble
    extends RemoteValue
```

Description

The RemoteDouble class extends RemoteValue for doubles.

See Also:

RemoteValue, RemoteDebugger

Methods

```
get()
toString()
typeName()
```


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.RemoteFloat

java.lang.Object

```
|
+----sun.tools.debug.RemoteValue
|
+----sun.tools.debug.RemoteFloat
```

Declaration

```
public class RemoteFloat
    extends RemoteValue
```

Description

The RemoteFloat class extends RemoteValue for floats.

See Also:

RemoteValue, RemoteDebugger

Methods

```
get()
toString()
typeName()
```

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

java.lang.Object

```
|
+----sun.tools.debug.RemoteValue
      |
      +----sun.tools.debug.RemoteInt
```

Declaration

```
public class RemoteInt
    extends RemoteValue
```

Description

The RemoteInt class extends RemoteValue for ints.

See Also:

RemoteValue, RemoteDebugger

Constructors

RemoteInt(int)

Methods

get()
toString()
typeName()

RemoteInt

```
public RemoteInt(int i)
```

get

```
public int get()
```

Return the int's value.

typeName

```
public String typeName()
```

Print this RemoteValue's type ("int").

Overrides:

typeName in class RemoteValue

toString

```
public String toString()
```

Return the int's value as a string.

Overrides:

toString in class Object

Class sun.tools.debug.RemoteLong

java.lang.Object

```
|
+----sun.tools.debug.RemoteValue
      |
      +----sun.tools.debug.RemoteLong
```

Declaration

```
public class RemoteLong
    extends RemoteValue
```

Description

The RemoteLong class extends RemoteValue for longs.

See Also:

RemoteValue, RemoteDebugger

Methods

```
get()
toString()
typeName()
```

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()
```

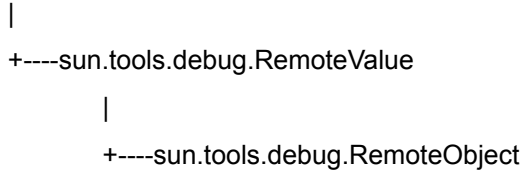
Return the long's value as a string.

Overrides:

toString in class Object

Class sun.tools.debug.RemoteObject

java.lang.Object



Declaration

```
public class RemoteObject
    extends RemoteValue
```

Description

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

Methods

```
description()
getClazz()
getField(int)
getField(String)
getFieldValue(int)
getFieldValue(String)
getFields()
getIdx()
```

toString()
typeName()

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()
```

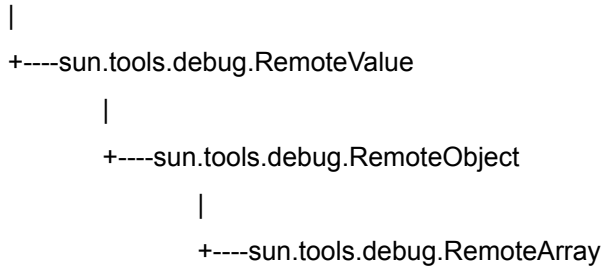
Return object as a string.

Overrides:

toString in class Object

Class sun.tools.debug.RemoteArray

java.lang.Object



Declaration

```
public class RemoteArray
    extends RemoteObject
```

Description

The RemoteArray class allows remote debugging of arrays.

See Also:

RemoteValue, RemoteDebugger

Methods

```
arrayTypeName(int)
description()
getElement(int)
getElementType()
getElements()
getElements(int, int)
getSize()
toString()
typeName()
```

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()
```

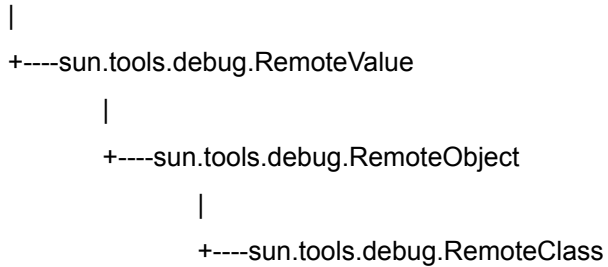
Return a string version of the array.

Overrides:

toString in class RemoteObject

Class sun.tools.debug.RemoteClass

java.lang.Object



Declaration

```
public class RemoteClass
    extends RemoteObject
```

Description

The RemoteClass class allows access to a class in a remote Java interpreter.

See Also:

RemoteDebugger

Methods

```
catchExceptions()
clearBreakpoint(int)
clearBreakpointLine(int)
clearBreakpointMethod(RemoteField)
description()
getClassLoader()
getField(int)
getField(String)
getFieldValue(int)
getFieldValue(String)
getFields()
```

getInstanceField(int)
getInstanceFields()
getInterfaces()
getMethod(String)
getMethodNames()
getMethods()
getName()
getSourceFile()
getSourceFileName()
getStaticFields()
getSuperclass()
ignoreExceptions()
isInterface()
setBreakpointLine(int)
setBreakpointMethod(RemoteField)
toString()
typeName()

getName

```
public String getName() throws Exception
```

Returns the name of the class.

typeName

```
public String typeName() throws Exception
```

Returns the name of the class as its type.

Overrides:

typeName in class RemoteObject

isInterface

```
public boolean isInterface() throws Exception
```

Is this RemoteClass an interface?

getSuperclass

```
public RemoteClass getSuperclass() throws Exception
```


Return the superclass for this class.

getClassLoader

```
public RemoteObject getClassLoader() throws Exception
```

Return the classloader for this class.

getInterfaces

```
public RemoteClass[] getInterfaces() throws Exception
```

Return the interfaces for this class.

getSourceFileName

```
public String getSourceFileName()
```

Get the name of the source file referenced by this stackframe.

getSourceFile

```
public InputStream getSourceFile() throws Exception
```

Get the source file referenced by this stackframe.

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()
```

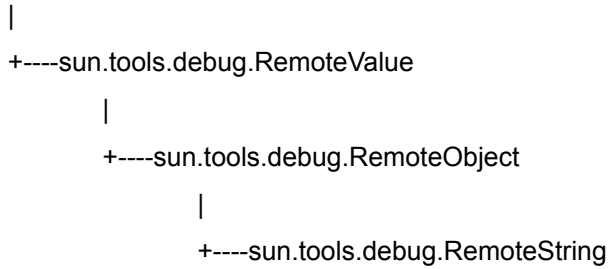
Return a (somewhat verbose) description.

Overrides:

toString in class RemoteObject

Class sun.tools.debug.RemoteString

java.lang.Object



Declaration

```
public class RemoteString
    extends RemoteObject
```

Description

The RemoteString class allows access to a string in a remote Java interpreter.

See Also:

RemoteDebugger

Methods

```
description()
toString()
typeName()
```

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()
```

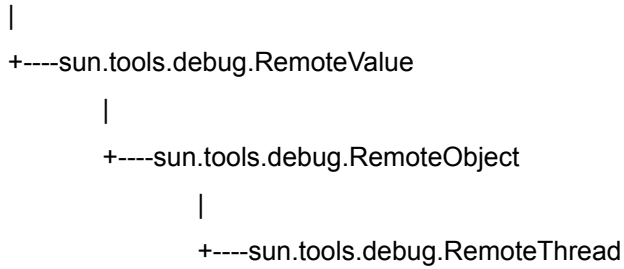
Return the string value, or "null"

Overrides:

toString in class RemoteObject

Class sun.tools.debug.RemoteThread

java.lang.Object



Declaration

```
public class RemoteThread
    extends RemoteObject
```

Description

The RemoteThread class allows access to a thread in a remote Java interpreter.

See Also:

RemoteDebugger, RemoteThreadGroup

Methods

```
cont()
down(int)
unter).
dumpStack()
getCurrentFrame()
getCurrentFrameIndex()
getName()
getStackVariable(String)
getStackVariables()
getStatus()
isSuspended()
```

next()
resetCurrentFrameIndex()
resume()
setCurrentFrameIndex(int)
step(boolean)
stop()
suspend()
up(int)

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
```

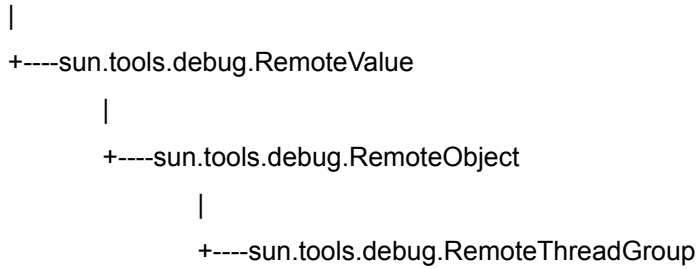
Return the arguments and local variable from the current stackframe.

Returns:

an array of RemoteValues.

Class sun.tools.debug.RemoteThreadGroup

java.lang.Object



Declaration

```
public class RemoteThreadGroup
    extends RemoteObject
```

Description

The `RemoteThreadGroup` class allows access to a threadgroup in a remote Java interpreter.

See Also:

`RemoteDebugger`, `RemoteThreadGroup`

Methods

```
getName()
listThreads(boolean)
stop()
```

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.RemoteShort

java.lang.Object

```
|
+----sun.tools.debug.RemoteValue
      |
      +----sun.tools.debug.RemoteShort
```

Declaration

```
public class RemoteShort
    extends RemoteValue
```

Description

The RemoteShort class extends RemoteValue for shorts.

See Also:

RemoteValue, RemoteDebugger

Methods

[get\(\)](#)

[toString\(\)](#)

[typeName\(\)](#)

get

Applies to

Class `sun.tools.debug.RemoteShort`

Declaration

```
public short get()
```

Description

Return the short's value.

typeName

Applies to

Class `sun.tools.debug.RemoteShort`

Declaration

```
public String typeName()
```

Description

Print this RemoteValue's type ("short").

Overrides:

typeName in class RemoteValue

toString

Applies to

Class `sun.tools.debug.RemoteShort`

Declaration

```
public String toString()
```

Description

Return the short's value as a string.

Overrides:

toString in class `Object`

Interface java.lang.Runnable

Declaration

```
public interface Runnable  
extends Object
```

Description

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

Methods

run()

run

Applies to

Interface `java.lang.Runnable`

```
public abstract void run()
```

Description

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

java.lang.Object

|

+----java.lang.Runtime

Declaration

public class Runtime

extends Object

Methods

[exec\(String\)](#)

[exec\(String, String\[\]\)](#)

[exec\(String\[\]\)](#)

[exec\(String\[\], String\[\]\)](#)

[exit\(int\)](#)

[freeMemory\(\)](#)

[gc\(\)](#)

[getLocalizedInputStream\(InputStream\)](#)

[getLocalizedOutputStream\(OutputStream\)](#)

[getRuntime\(\)](#)

[load\(String\)](#)

[loadLibrary\(String\)](#)

[runFinalization\(\)](#)

[totalMemory\(\)](#)

[traceInstructions\(boolean\)](#)

[traceMethodCalls\(boolean\)](#)

getRuntime

Applies to

Class java.lang.Runtime

Declaration

```
public static Runtime getRuntime()
```

Description

Returns the runtime.

exit

Applies to

Class java.lang.Runtime

Declaration

```
public void exit(int status)
```

Description

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

Applies to

[Class java.lang.Runtime](#)

Declaration

```
public Process exec(String command) throws IOException
public Process exec(String command, String envp[]) throws IOException
public Process exec(String cmdarray[]) throws IOException
public Process exec(String cmdarray[], String envp[]) throws IOException
```

Description

public Process exec(String command) throws IOException

Executes the system command specified in the parameter. Returns a Process which has methods for obtaining 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

Applies to

[Class java.lang.Runtime](#)

public Process exec(String command, String envp[]) throws IOException

Executes the system command specified in the parameter. Returns a Process which has methods for obtaining 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

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 obtaining 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

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 obtaining 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

Applies to

Class `java.lang.Runtime`

Declaration

```
public long freeMemory()
```

Description

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

Applies to

Class java.lang.Runtime

Declaration

```
public long totalMemory()
```

Description

Returns the total number of bytes in system memory.

gc

Applies to

Class java.lang.Runtime

Declaration

```
public void gc()
```

Description

Runs the garbage collector.

runFinalization

Applies to

Class `java.lang.Runtime`

Declaration

```
public void runFinalization()
```

Description

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

Applies to

Class java.lang.Runtime

Declaration

```
public void traceInstructions(boolean on)
```

Description

Enables/Disables tracing of instructions.

Parameters:

on - start tracing if true

traceMethodCalls

Applies to

Class java.lang.Runtime

Declaration

```
public void traceMethodCalls(boolean on)
```

Description

Enables/Disables tracing of method calls.

Parameters:

on - start tracing if true

load

Applies to

Class `java.lang.Runtime`

Declaration

```
public synchronized void load(String filename)
```

Description

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

Applies to

Class `java.lang.Runtime`

Declaration

```
public synchronized void loadLibrary(String libname)
```

Description

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

Applies to

Class `java.lang.Runtime`

Declaration

```
public InputStream getLocalizedInputStream(InputStream in)
```

Description

Localize an input stream. A localized input stream will automatically translate the input from the local format to UNICODE.

getLocalizedOutputStream

Applies to

Class `java.lang.Runtime`

Declaration

```
public OutputStream getLocalizedOutputStream(OutputStream out)
```

Description

Localize an output stream. A localized output stream will automatically translate the output from UNICODE to the local format.

Class java.lang.Runtime

java.lang.Object

|

+----java.lang.Runtime

Declaration

public class Runtime

extends Object

Methods

[exec\(String\)](#)

[exec\(String, String\[\]\)](#)

[exec\(String\[\]\)](#)

[exec\(String\[\], String\[\]\)](#)

[exit\(int\)](#)

[freeMemory\(\)](#)

[gc\(\)](#)

[getLocalizedInputStream\(InputStream\)](#)

[getLocalizedOutputStream\(OutputStream\)](#)

[getRuntime\(\)](#)

[load\(String\)](#)

[loadLibrary\(String\)](#)

[runFinalization\(\)](#)

[totalMemory\(\)](#)

[traceInstructions\(boolean\)](#)

[traceMethodCalls\(boolean\)](#)

getRuntime

Applies to

Class `java.lang.Runtime`

Declaration

```
public static Runtime getRuntime()
```

Description

Returns the runtime.

exit

Applies to

Class java.lang.Runtime

Declaration

```
public void exit(int status)
```

Description

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

Applies to

Class `java.lang.Runtime`

Declaration

```
public Process exec(String command) throws IOException
public Process exec(String command, String envp[]) throws IOException
public Process exec(String cmdarray[]) throws IOException
public Process exec(String cmdarray[], String envp[]) throws IOException
```

Description

public Process exec(String command) throws IOException

Executes the system command specified in the parameter. Returns a Process which has methods for obtaining 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

public Process exec(String command, String envp[]) throws IOException

Executes the system command specified in the parameter. Returns a Process which has methods for obtaining 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

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 obtaining 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
-

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 obtaining 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

Applies to

Class java.lang.Runtime

Declaration

```
public long freeMemory()
```

Description

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

Applies to

Class java.lang.Runtime

Declaration

```
public long totalMemory()
```

Description

Returns the total number of bytes in system memory.

gc

Applies to

Class java.lang.Runtime

Declaration

```
public void gc()
```

Description

Runs the garbage collector.

runFinalization

Applies to

Class `java.lang.Runtime`

Declaration

```
public void runFinalization()
```

Description

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

Applies to

Class java.lang.Runtime

Declaration

```
public void traceInstructions(boolean on)
```

Description

Enables/Disables tracing of instructions.

Parameters:

on - start tracing if true

traceMethodCalls

Applies to

Class java.lang.Runtime

Declaration

```
public void traceMethodCalls(boolean on)
```

Description

Enables/Disables tracing of method calls.

Parameters:

on - start tracing if true

load

Applies to

Class `java.lang.Runtime`

Declaration

```
public synchronized void load(String filename)
```

Description

Loads a dynamic library, given a complete path name. If you use this from `java_g` it will automatically 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

Applies to

Class java.lang.Runtime

Declaration

```
public synchronized void loadLibrary(String libname)
```

Description

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

Applies to

Class `java.lang.Runtime`

Declaration

```
public InputStream getLocalizedInputStream(InputStream in)
```

Description

Localize an input stream. A localized input stream will automatically translate the input from the local format to UNICODE.

getLocalizedOutputStream

Applies to

Class `java.lang.Runtime`

Declaration

```
public OutputStream getLocalizedOutputStream(OutputStream out)
```

Description

Localize an output stream. A localized output stream will automatically translate the output from UNICODE to the local format.

Interface `java.awt.peer.ScrollbarPeer`

Declaration

```
public interface ScrollbarPeer
    extends Object
    extends ComponentPeer
```

Methods

```
setLineIncrement(int)  
setPageIncrement(int)  
setValue(int)  
setValues(int, int, int, int)
```

setValue

Declaration

```
public abstract void setValue(int value)
```

setValues

Declaration

```
public abstract void setValues(int value,  
                               int visible,  
                               int minimum,  
                               int maximum)
```

setLineIncrement

Declaration

```
public abstract void setLineIncrement(int l)
```


setPageIncrement

```
public abstract void setPageIncrement(int l)
```

Class java.lang.SecurityManager

java.lang.Object

|

+----java.lang.SecurityManager

Declaration

```
public class SecurityManager
```

```
extends Object
```

Description

An abstract class that can be subclassed to implement a security policy. It allows the inspection of the classloaders on the execution stack.

Variables

inCheck

Constructors

SecurityManager()

Methods

checkAccept(String, int)

checkAccess(Thread)

checkAccess(ThreadGroup)

checkConnect(String, int)

checkConnect(String, int, Object)

checkCreateClassLoader()

checkDelete(String)

checkExec(String)

checkExit(int)

checkLink(String)

checkListen(int)

checkPackageAccess(String)
checkPackageDefinition(String)
checkPropertiesAccess()
checkPropertyAccess(String)
checkPropertyAccess(String, String)
checkRead(FileDescriptor)
checkRead(String)
checkRead(String, Object)
checkSetFactory()
checkTopLevelWindow(Object)
checkWrite(FileDescriptor)
checkWrite(String)
classDepth(String)
classLoaderDepth()
currentClassLoader()
getClassContext()
getInCheck()
getSecurityContext()
inClass(String)
inClassLoader()

inCheck

Declaration

protected boolean inCheck

SecurityManager

Declaration

`protected SecurityManager()`

Description

Constructs a new SecurityManager.

Throws: `SecurityException`

If the security manager cannot be created.

getInCheck

Declaration

```
public boolean getInCheck()
```

Description

Returns whether there is a security check in progress.

getClassContext

Declaration

```
protected Class[] getClassContext()
```

Description

Gets the context of this Class.

currentClassLoader

Declaration

```
protected ClassLoader currentClassLoader()
```

Description

The current ClassLoader on the execution stack.

classDepth

Declaration

```
protected int classDepth(String name)
```

Description

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

Declaration

```
protected int classLoaderDepth()
```

inClass

Declaration

protected boolean inClass(String name)

Description

Returns true if the specified String is in this Class.

Parameters:

name - the name of the class

inClassLoader

Declaration

```
protected boolean inClassLoader()
```

Description

Returns a boolean indicating whether or not the current ClassLoader is equal to null.

getSecurityContext

Declaration

```
public Object getSecurityContext()
```

Description

Returns an implementation-dependent Object which encapsulates enough information about the current execution environment to perform some of the security checks later.

checkCreateClassLoader

Declaration

```
public void checkCreateClassLoader()
```

Description

Checks to see if the ClassLoader has been created.

Throws: `SecurityException`

If a security error has occurred.

checkAccess

Declaration

```
public void checkAccess(Thread g)
public void checkAccess(ThreadGroup g)
```

Description

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.

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

Declaration

```
public void checkExit(int status)
```

Description

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.

checkExec

Declaration

```
public void checkExec(String cmd)
```

Description

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.

checkLink

Declaration

```
public void checkLink(String lib)
```

Description

Checks to see if the specified linked library exists.

Parameters:

lib - the name of the library

Throws: `SecurityException`

If the library does not exist.

checkRead

Declaration

```
public void checkRead(FileDescriptor fd)
public void checkRead(String file)
public void checkRead(String file, Object context)
```

Description

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.

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.

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

Declaration

```
public void checkWrite(FileDescriptor fd)
public void checkWrite(String file)
```

Description

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.

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

Declaration

```
public void checkDelete(String file)
```

Description

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

Declaration

```
public void checkConnect(String host, int port)
public void checkConnect(String host, int port, Object context)
```

Description

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.

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

Declaration

```
public void checkListen(int port)
```

Description

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

Declaration

```
public void checkAccept(String host, int port)
```

Description

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

Declaration

```
public void checkPropertiesAccess()
```

Description

Checks to see who has access to the System properties.

Throws: `SecurityException`

If a security error has occurred.

checkPropertyAccess

Declaration

```
public void checkPropertyAccess(String key)
public void checkPropertyAccess(String key, String def)
```

Description

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.

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

Declaration

```
public boolean checkTopLevelWindow(Object window)
```

Description

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

Declaration

```
public void checkPackageAccess(String pkg)
```

Description

Checks to see if an applet can access a package.

checkPackageDefinition

Declaration

```
public void checkPackageDefinition(String pkg)
```

Description

Checks to see if an applet can define classes in a package.

checkSetFactory

Declaration

```
public void checkSetFactory()
```

Description

Checks to see if an applet can set a networking-related object factory.

Class java.net.ServerSocket

java.lang.Object

|

+----java.net.ServerSocket

Declaration

public final class ServerSocket

extends Object

Description

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.

Constructors

ServerSocket(int)

ServerSocket(int, int)

Methods

[accept\(\)](#)

[close\(\)](#)

[getInetAddress\(\)](#)

[getLocalPort\(\)](#)

[setSocketFactory\(SocketImplFactory\)](#)

[toString\(\)](#)

ServerSocket

Applies to

Class `java.net.ServerSocket`

Declaration

```
public ServerSocket(int port) throws IOException  
public ServerSocket(int port, int count) throws IOException
```

Description

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.

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 anonymous port by specifying the port number to be 0.

Parameters:

port - the specified port

count - the amount of time to listen for a connection

getInetAddress

Applies to

Class `java.net.ServerSocket`

Declaration

```
public InetAddress getInetAddress()
```

Description

Gets the address to which the socket is connected.

getLocalPort

Applies to

Class `java.net.ServerSocket`

Declaration

```
public int getLocalPort()
```

Description

Gets the port on which the socket is listening.

accept

Applies to

Class `java.net.ServerSocket`

Declaration

```
public Socket accept() throws IOException
```

Description

Accepts a connection. This method will block until the connection is made.

Throws: `IOException`

IO error when waiting for the connection.

close

Applies to

Class `java.net.ServerSocket`

Declaration

`public void close()` throws `IOException`

Description

Closes the server socket.

Throws: `IOException`

IO error when closing the socket.

toString

Applies to

Class `java.net.ServerSocket`

Declaration

```
public String toString()
```

Description

Returns the implementation address and implementation port of this `ServerSocket` as a `String`.

Overrides:

toString in class `Object`

setSocketFactory

Applies to

Class `java.net.ServerSocket`

Declaration

```
public static synchronized void setSocketFactory(SocketImplFactory fac) throws IOException
```

Description

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

java.lang.Object

|

+----java.net.Socket

Declaration

public final class Socket

extends Object

Description

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.

Constructors

[Socket\(String, int\)](#)

[Socket\(String, int, boolean\)](#)

[Socket\(InetAddress, int\)](#)

[Socket\(InetAddress, int, boolean\)](#)

Methods

[close\(\)](#)

[getInetAddress\(\)](#)

[getInputStream\(\)](#)

[getLocalPort\(\)](#)

[getOutputStream\(\)](#)

[getPort\(\)](#)

[setSocketImplFactory\(SocketImplFactory\)](#)

[toString\(\)](#)

Socket

Applies to

Class `java.net.Socket`

Declaration

```
public Socket(String host, int port) throws UnknownHostException, IOException  
public Socket(String host, int port, boolean stream) throws IOException  
public Socket(InetAddress address, int port) throws IOException  
public Socket(InetAddress address, int port, boolean stream) throws IOException
```

Description

`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

`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

`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

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

Applies to

Class `java.net.Socket`

Declaration

```
public InetAddress getInetAddress()
```

Description

Gets the address to which the socket is connected.

getPort

Applies to

Class `java.net.Socket`

Declaration

```
public int getPort()
```

Description

Gets the remote port to which the socket is connected.

getLocalPort

Applies to

Class `java.net.Socket`

Declaration

```
public int getLocalPort()
```

Description

Gets the local port to which the socket is connected.

getInputStream

Applies to

Class `java.net.Socket`

Declaration

```
public InputStream getInputStream() throws IOException
```

Description

Gets an `InputStream` for this socket.

getOutputStream

Applies to

Class `java.net.Socket`

Declaration

```
public OutputStream getOutputStream() throws IOException
```

Description

Gets an OutputStream for this socket.

close

Applies to

Class `java.net.Socket`

Declaration

public synchronized void close() throws IOException

Description

Closes the socket.

toString

Applies to

Class `java.net.Socket`

Declaration

```
public String toString()
```

Description

Converts the Socket to a String.

Overrides:

toString in class Object

setSocketImplFactory

Applies to

Class `java.net.Socket`

Declaration

```
public static synchronized void setSocketImplFactory(SocketImplFactory fac) throws IOException
```

Description

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.SocketImpl

java.lang.Object

|

+----java.net.SocketImpl

Declaration

public class SocketImpl

extends Object

Description

This is the Socket implementation class. It is an abstract class that must be subclassed to provide an actual implementation.

Variables

address

fd

localport

port

Constructors

SocketImpl()

Methods

accept(SocketImpl)

available()

bind(InetAddress, int)

close()

connect(String, int)

connect(InetAddress, int)

create(boolean)

getFileDescriptor()

getInetAddress()

getInputStream()

getLocalPort()

getOutputStream()

getPort()

listen(int)

toString()

fd

Applies to

Class `java.net.SocketImpl`

Declaration

protected FileDescriptor fd

Description

The file descriptor object

address

Applies to

Class `java.net.SocketImpl`

Declaration

protected InetAddress address

Description

The internet address where the socket will make a connection.

port

Applies to

Class `java.net.SocketImpl`

Declaration

protected int port

Description

The port where the socket will make a connection.

localport

Applies to

Class `java.net.SocketImpl`

Declaration

protected int localport

SocketImpl

Applies to

Class `java.net.SocketImpl`

Declaration

```
public SocketImpl()
```

create

Applies to

Class `java.net.SocketImpl`

Declaration

protected abstract void create(boolean stream) throws IOException

Description

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

Applies to

Class `java.net.SocketImpl`

Declaration

protected abstract void connect(String host, int port) throws IOException

protected abstract void connect(InetAddress address, int port) throws IOException

Description

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

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

Applies to

Class `java.net.SocketImpl`

Declaration

```
protected abstract void bind(InetAddress host,  
                             int port) throws IOException
```

Description

Binds the socket to the specified port on the specified host.

Parameters:

host - the host

port - the port

listen

Applies to

Class `java.net.SocketImpl`

Declaration

protected abstract void listen(int count) throws IOException

Description

Listens for connections over a specified amount of time.

Parameters:

count - the amount of time this socket will listen for connections

accept

Applies to

Class `java.net.SocketImpl`

Declaration

protected abstract void accept(SocketImpl s) throws IOException

Description

Accepts a connection.

Parameters:

s - the accepted connection

getInputStream

Applies to

Class `java.net.SocketImpl`

Declaration

protected abstract InputStream getInputStream() throws IOException

Description

Gets an InputStream for this socket.

getOutputStream

Applies to

Class `java.net.SocketImpl`

Declaration

protected abstract OutputStream getOutputStream() throws IOException

Description

Gets an OutputStream for this socket.

available

Applies to

Class `java.net.SocketImpl`

Declaration

protected abstract int available() throws IOException

Description

Returns the number of bytes that can be read without blocking.

close

Applies to

Class `java.net.SocketImpl`

Declaration

protected abstract void close() throws IOException

Description

Closes the socket.

getFileDescriptor

Applies to

Class `java.net.SocketImpl`

Declaration

```
protected FileDescriptor getFileDescriptor()
```

getInetAddress

Applies to

Class `java.net.SocketImpl`

Declaration

```
protected InetAddress getInetAddress()
```

getPort

Applies to

Class `java.net.SocketImpl`

Declaration

```
protected int getPort()
```

getLocalPort

Applies to

Class `java.net.SocketImpl`

Declaration

```
protected int getLocalPort()
```

toString

Applies to

Class `java.net.SocketImpl`

Declaration

```
public String toString()
```

Description

Returns the address and port of this Socket as a String.

Overrides:

toString in class Object

Interface `java.net.SocketImplFactory`

Declaration

```
public interface SocketImplFactory  
extends Object
```

Description

This interface defines a factory for `SocketImpl` instances. It is used by the `socket` class to create socket implementations that implement various policies.

Methods

`createSocketImpl()`

createSocketImpl

Applies to

Interface `java.net.SocketImplFactory`

Declaration

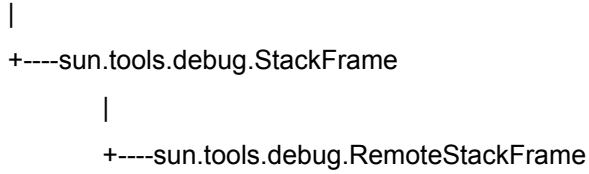
```
public abstract SocketImpl createSocketImpl()
```

Description

Creates a new `SocketImpl` instance.

Class sun.tools.debug.RemoteStackFrame

java.lang.Object



Declaration

```
public class RemoteStackFrame
    extends StackFrame
```

Description

The RemoteStackFrame class provides access to a stackframe of a suspended thread.

See Also:

RemoteDebugger, RemoteThread

Methods

- [getLineNumber\(\)](#)
- [getLocalVariable\(String\)](#)
- [getLocalVariables\(\)](#)
- [getMethodName\(\)](#)
- [getPC\(\)](#)
- [getRemoteClass\(\)](#)

getLocalVariable

Applies to

Class `sun.tools.debug.RemoteStackFrame`

Declaration

```
public RemoteStackVariable getLocalVariable(String name) throws Exception
```

Description

Return a specific (named) stack variable. A slot number of -1 indicates that the variable is not currently in scope.

getLocalVariables

Applies to

Class `sun.tools.debug.RemoteStackFrame`

Declaration

```
public RemoteStackVariable[] getLocalVariables() throws Exception
```

Description

Return an array of all valid local variables and method arguments for this stack frame.

getLineNumber

Applies to

Class `sun.tools.debug.RemoteStackFrame`

Declaration

```
public int getLineNumber()
```

Description

Return the source file line number.

getMethodName

Applies to

Class `sun.tools.debug.RemoteStackFrame`

Declaration

```
public String getMethodName()
```

Description

Get the method name referenced by this stackframe.

getPC

Applies to

Class `sun.tools.debug.RemoteStackFrame`

Declaration

```
public int getPC()
```

Description

Get the program counter referenced by this stackframe.

getRemoteClass

Applies to

Class `sun.tools.debug.RemoteStackFrame`

Declaration

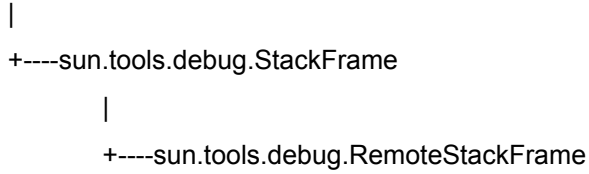
```
public RemoteClass getRemoteClass()
```

Description

Get the class this stackframe references.

Class sun.tools.debug.RemoteStackFrame

java.lang.Object



Declaration

```
public class RemoteStackFrame
    extends StackFrame
```

Description

The RemoteStackFrame class provides access to a stackframe of a suspended thread.

See Also:

RemoteDebugger, RemoteThread

Methods

- [getLineNumber\(\)](#)
- [getLocalVariable\(String\)](#)
- [getLocalVariables\(\)](#)
- [getMethodName\(\)](#)
- [getPC\(\)](#)
- [getRemoteClass\(\)](#)

getLocalVariable

Applies to

Class `sun.tools.debug.RemoteStackFrame`

Declaration

```
public RemoteStackVariable getLocalVariable(String name) throws Exception
```

Description

Return a specific (named) stack variable. A slot number of -1 indicates that the variable is not currently in scope.

getLocalVariables

Applies to

Class `sun.tools.debug.RemoteStackFrame`

Declaration

```
public RemoteStackVariable[] getLocalVariables() throws Exception
```

Description

Return an array of all valid local variables and method arguments for this stack frame.

getLineNumber

Applies to

Class `sun.tools.debug.RemoteStackFrame`

Declaration

```
public int getLineNumber()
```

Description

Return the source file line number.

getMethodName

Applies to

Class `sun.tools.debug.RemoteStackFrame`

Declaration

```
public String getMethodName()
```

Description

Get the method name referenced by this stackframe.

getPC

Applies to

Class `sun.tools.debug.RemoteStackFrame`

Declaration

```
public int getPC()
```

Description

Get the program counter referenced by this stackframe.

getRemoteClass

Applies to

Class sun.tools.debug.RemoteStackFrame

Declaration

```
public RemoteClass getRemoteClass()
```

Description

Get the class this stackframe references.

Class java.io.StreamTokenizer

java.lang.Object

|

+----java.io.StreamTokenizer

Declaration

```
public class StreamTokenizer
```

```
extends Object
```

Description

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.

Variables

TT_EOF

TT_EOL

TT_NUMBER

TT_WORD

nval

sval

ttype

Constructors

StreamTokenizer(InputStream)

Methods

commentChar(int)

eollsSignificant(boolean)

lineno()

lowerCaseMode(boolean)

nextToken()

ordinaryChar(int)

ordinaryChars(int, int)

parseNumbers()

pushBack()

quoteChar(int)

resetSyntax()

slashSlashComments(boolean)

slashStarComments(boolean)

toString()

whitespaceChars(int, int)

wordChars(int, int)

ttype

Applies to

Class `java.io.StreamTokenizer`

Declaration

```
public int ttype
```

Description

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

Applies to

Class `java.io.StreamTokenizer`

Declaration

```
public final static int TT_EOF
```

Description

The End-of-file token.

TT_EOL

Applies to

Class `java.io.StreamTokenizer`

Declaration

```
public final static int TT_EOL
```

Description

The End-of-line token.

TT_NUMBER

Applies to

Class `java.io.StreamTokenizer`

Declaration

```
public final static int TT_NUMBER
```

Description

The number token. This value is in `nval`.

TT_WORD

Applies to

Class `java.io.StreamTokenizer`

Declaration

```
public final static int TT_WORD
```

Description

The word token. This value is in sval.

sval

Applies to

Class `java.io.StreamTokenizer`

Declaration

```
public String sval
```

Description

The Stream value.

nval

Applies to

Class `java.io.StreamTokenizer`

Declaration

```
public double nval
```

Description

The number value.

StreamTokenizer

Applies to

Class `java.io.StreamTokenizer`

Declaration

```
public StreamTokenizer(InputStream I)
```

Description

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

resetSyntax

Applies to

Class `java.io.StreamTokenizer`

Declaration

```
public void resetSyntax()
```

Description

Resets the syntax table so that all characters are special.

wordChars

Applies to

Class `java.io.StreamTokenizer`

Declaration

```
public void wordChars(int low,  
                      int hi)
```

Description

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

Applies to

Class `java.io.StreamTokenizer`

Declaration

```
public void whitespaceChars(int low,  
                             int hi)
```

Description

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

Applies to

Class `java.io.StreamTokenizer`

Declaration

```
public void ordinaryChars(int low,  
                           int hi)
```

Description

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

Applies to

Class `java.io.StreamTokenizer`

Declaration

```
public void ordinaryChar(int ch)
```

Description

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

Applies to

Class `java.io.StreamTokenizer`

Declaration

```
public void commentChar(int ch)
```

Description

Specifies that this character starts a single line comment.

Parameters:

ch - the character

quoteChar

Applies to

Class `java.io.StreamTokenizer`

Declaration

```
public void quoteChar(int ch)
```

Description

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

Applies to

Class `java.io.StreamTokenizer`

Declaration

```
public void parseNumbers()
```

Description

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.

eollsSignificant

Applies to

Class `java.io.StreamTokenizer`

Declaration

```
public void eollsSignificant(boolean flag)
```

Description

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

Applies to

Class `java.io.StreamTokenizer`

Declaration

```
public void slashStarComments(boolean flag)
```

Description

If the flag is true, recognize C style(/*) comments.

slashSlashComments

Applies to

Class `java.io.StreamTokenizer`

Declaration

```
public void slashSlashComments(boolean flag)
```

Description

If the flag is true, recognize C++ style(`//`) comments.

lowerCaseMode

Applies to

Class `java.io.StreamTokenizer`

Declaration

```
public void lowerCaseMode(boolean fl)
```

Description

Examines a boolean to decide whether TT_WORD tokens are forced to be lower case.

Parameters:

fl - the boolean flag

nextToken

Applies to

Class `java.io.StreamTokenizer`

Declaration

```
public int nextToken() throws IOException
```

Description

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

Applies to

Class `java.io.StreamTokenizer`

Declaration

```
public void pushBack()
```

Description

Pushes back a stream token.

lineno

Applies to

Class `java.io.StreamTokenizer`

Declaration

```
public int lineno()
```

Description

Return the current line number.

toString

Applies to

Class `java.io.StreamTokenizer`

Declaration

```
public String toString()
```

Description

Returns the String representation of the stream token.

Overrides:

toString in class Object

Class java.lang.String

java.lang.Object

|

+----java.lang.String

Declaration

```
public final class String
```

```
extends Object
```

Description

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

Constructors

[String\(\)](#)

[String\(String\)](#)

[String\(char\[\]\)](#)

[String\(char\[\], int, int\)](#)

[String\(byte\[\], int, int, int\)](#)

[String\(byte\[\], int\)](#)

[String\(StringBuffer\)](#)

Methods

[charAt\(int\)](#)

[compareTo\(String\)](#)

[concat\(String\)](#)

[copyValueOf\(char\[\], int, int\)](#)

[copyValueOf\(char\[\]\)](#)

[endsWith\(String\)](#)

[equals\(Object\)](#)

[equalsIgnoreCase\(String\)](#)

[getBytes\(int, int, byte\[\], int\)](#)

[getChars\(int, int, char\[\], int\)](#)

[hashCode\(\)](#)

[indexOf\(int\)](#)

[indexOf\(int, int\)](#)

[indexOf\(String\)](#)

[indexOf\(String, int\)](#)

[intern\(\)](#)

[lastIndexOf\(int\)](#)

[lastIndexOf\(int, int\)](#)

[lastIndexOf\(String\)](#)

[lastIndexOf\(String, int\)](#)

[length\(\)](#)

[regionMatches\(int, String, int, int\)](#)

[regionMatches\(boolean, int, String, int, int\)](#)

[replace\(char, char\)](#)

[startsWith\(String, int\)](#)

startsWith(String)

substring(int)

substring(int, int)

toCharArray()

toLowerCase()

toString()

toUpperCase()

trim()

valueOf(Object)

valueOf(char[])

valueOf(char[], int, int)

valueOf(boolean)

valueOf(char)

valueOf(int)

valueOf(long)

valueOf(float)

valueOf(double)

String

Applies to

Class `java.lang.String`

Declaration

```
public String()  
public String(String value)  
public String(char value[])  
public String(char value[],int offset, int count)  
public String(byte ascii[], int hibyte, int offset, int count)  
public String(byte ascii[], int hibyte)  
public String(StringBuffer buffer)
```

Description

public String()

Constructs a new empty 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

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 hbyte as the upper byte of each character.

Parameters:

ascii - the byte that will be converted to characters

hbyte - the top 8 bits of each 16 bit Unicode character

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

Applies to

Class `java.lang.String`

Declaration

```
public int length()
```

Description

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

Applies to

Class `java.lang.String`

Declaration

```
public char charAt(int index)
```

Description

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

Applies to

Class `java.lang.String`

Declaration

```
public void getChars(int srcBegin,  
                    int srcEnd,  
                    char dst[],  
                    int dstBegin)
```

Description

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

Applies to

Class `java.lang.String`

Declaration

```
public void getBytes(int srcBegin,  
                    int srcEnd,  
                    byte dst[],  
                    int dstBegin)
```

Description

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

Applies to

Class java.lang.String

Declaration

```
public boolean equals(Object anObject)
```

Description

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

Applies to

Class `java.lang.String`

Declaration

```
public boolean equalsIgnoreCase(String anotherString)
```

Description

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

Applies to

Class java.lang.String

Declaration

```
public int compareTo(String anotherString)
```

Description

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

Applies to

Class `java.lang.String`

Declaration

```
public boolean regionMatches(int toffset, String other, int ooffset, int len)
public boolean regionMatches(boolean ignoreCase, int toffset, String other, int ooffset, int len)
```

Description

```
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.

```
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

Applies to

Class java.lang.String

Declaration

```
public boolean startsWith(String prefix, int toffset)
public boolean startsWith(String prefix)
```

Description

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.

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

Applies to

Class `java.lang.String`

Declaration

```
public boolean endsWith(String suffix)
```

Description

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

Applies to

Class `java.lang.String`

Declaration

```
public int hashCode()
```

Description

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

Applies to

Class `java.lang.String`

Declaration

```
public int indexOf(int ch)
public int indexOf(int ch,
                  int fromIndex)
public int indexOf(String str)
public int indexOf(String str,
                  int fromIndex)
```

Description

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

**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

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

**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

Applies to

Class `java.lang.String`

Declaration

```
public int lastIndexOf(int ch)
public int lastIndexOf(int ch,
                       int fromIndex)
public int lastIndexOf(String str)
public int lastIndexOf(String str,
                       int fromIndex)
```

Description

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

**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

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

**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

Applies to

Class `java.lang.String`

Declaration

```
public String substring(int beginIndex)
public String substring(int beginIndex,
                        int endIndex)
```

Description

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

**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

Applies to

Class java.lang.String

Declaration

```
public String concat(String str)
```

Description

Concatenates the specified string to the end of this String.

Parameters:

str - the String which is concatenated to the end of this String

replace

Applies to

Class `java.lang.String`

Declaration

```
public String replace(char oldChar,  
                      char newChar)
```

Description

Converts this String by replacing all occurrences of oldChar with newChar.

Parameters:

oldChar - the old character

newChar - the new character

toLowerCase

Applies to

Class `java.lang.String`

Declaration

```
public String toLowerCase()
```

Description

Converts all of the characters in this String to lower case.

Returns:

the String, converted to lowercase.

See Also:

toLowerCase, toUpperCase

toUpperCase

Applies to

Class java.lang.String

Declaration

```
public String toUpperCase()
```

Description

Converts all of the characters in this String to upper case.

Returns:

the String, converted to uppercase.

See Also:

toUpperCase, toLowerCase

trim

Applies to

Class `java.lang.String`

Declaration

```
public String trim()
```

Description

Trims leading and trailing whitespace from this String.

Returns:

the String, with whitespace removed.

toString

Applies to

Class java.lang.String

Declaration

```
public String toString()
```

Description

Converts this String to a String.

Returns:

the String itself.

Overrides:

toString in class Object

toCharArray

Applies to

Class `java.lang.String`

Declaration

```
public char[] toCharArray()
```

Description

Converts this String to a character array. This creates a new array.

Returns:

an array of characters.

valueOf

Applies to

Class `java.lang.String`

Declaration

```
public static String valueOf(Object obj)
public static String valueOf(char data[])
public static String valueOf(char data[],
                             int offset,
                             int count)
public static String valueOf(boolean b)
public static String valueOf(char c)
public static String valueOf(int i)
public static String valueOf(long l)
public static String valueOf(float f)
public static String valueOf(double d)
```

Description

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

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 does not copy it to a new array).

Parameters:

data - the character array

```
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

```
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

public static String valueOf(double d)

Returns a String object that represents the value of the specified double.

Parameters:

d - the double

copyValueOf

Applies to

Class `java.lang.String`

Declaration

```
public static String copyValueOf(char data[],  
                                int offset,  
                                int count)  
public static String copyValueOf(char data[])
```

Description

```
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

```
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

intern

Applies to

Class `java.lang.String`

Declaration

```
public String intern()
```

Description

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

java.lang.Object

|

+----java.lang.StringBuffer

Declaration

```
public final class StringBuffer
```

```
extends Object
```

Description

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

Constructors

[StringBuffer\(\)](#)

[StringBuffer\(int\)](#)

[StringBuffer\(String\)](#)

Methods

append(Object)
append(String)
append(char[])
append(char[], int, int)
append(boolean)
append(char)
append(int)
append(long)
append(float)
append(double)
capacity()
charAt(int)
ensureCapacity(int)
getChars(int, int, char[], int)
insert(int, Object)
insert(int, String)
insert(int, char[])
insert(int, boolean)
insert(int, char)
insert(int, int)
insert(int, long)
insert(int, float)
insert(int, double)
length()
setCharAt(int, char)
setLength(int)
toString()

StringBuffer

Applies to

Class `java.lang.StringBuffer`

Declaration

```
public StringBuffer()  
public StringBuffer(int length)  
public StringBuffer(String str)
```

Description

public StringBuffer()

Constructs an empty String buffer.

public StringBuffer(int length)

Constructs an empty String buffer with the specified initial length.

Parameters:

length - the initial length

public StringBuffer(String str)

Constructs a String buffer with the specified initial value.

Parameters:

str - the initial value of the buffer

length

Applies to

Class `java.lang.StringBuffer`

Declaration

```
public int length()
```

Description

Returns the length (character count) of the buffer.

capacity

Applies to

Class `java.lang.StringBuffer`

Declaration

```
public int capacity()
```

Description

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

Applies to

Class `java.lang.StringBuffer`

Declaration

```
public synchronized void ensureCapacity(int minimumCapacity)
```

Description

Ensures that the capacity of the buffer is at least equal to the specified minimum.

Parameters:

minimumCapacity - the minimum desired capacity

setLength

Applies to

Class `java.lang.StringBuffer`

Declaration

```
public synchronized void setLength(int newLength)
```

Description

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

Applies to

Class `java.lang.StringBuffer`

Declaration

```
public synchronized char charAt(int index)
```

Description

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

Applies to

Class `java.lang.StringBuffer`

Declaration

```
public synchronized void getChars(int srcBegin,  
                                   int srcEnd,  
                                   char dst[],  
                                   int dstBegin)
```

Description

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

Applies to

Class `java.lang.StringBuffer`

Declaration

```
public synchronized void setCharAt(int index,  
                                   char ch)
```

Description

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

Applies to

Class `java.lang.StringBuffer`

Declaration

```
public synchronized StringBuffer append(Object obj)
public synchronized StringBuffer append(String str)
public synchronized StringBuffer append(char str[])
public synchronized StringBuffer append(char str[],
                                     int offset,
                                     int len)

public StringBuffer append(boolean b)
public synchronized StringBuffer append(char c)
public StringBuffer append(int i)
public StringBuffer append(long l)
public StringBuffer append(float f)
public StringBuffer append(double d)
```

Description

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.

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.

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.

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.

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.

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

Applies to

Class `java.lang.StringBuffer`

Declaration

```
public synchronized StringBuffer insert(int offset,  
                                         Object obj)  
public synchronized StringBuffer insert(int offset,  
                                         String str)  
public synchronized StringBuffer insert(int offset,  
                                         char str[])  
public StringBuffer insert(int offset,  
                           boolean b)  
public synchronized StringBuffer insert(int offset,  
                                         char c)  
public StringBuffer insert(int offset,  
                           int i)  
public StringBuffer insert(int offset,  
                           long l)  
public StringBuffer insert(int offset,  
                           float f)  
public StringBuffer insert(int offset,  
                           double d)
```

Description

```
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.

```
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.

```
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.

```
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.

```
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.

```
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.

```
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.

```
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.

```
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

Applies to

Class java.lang.StringBuffer

Declaration

```
public String toString()
```

Description

Converts to a String representing the data in the buffer.

Overrides:

toString in class Object

Class java.util.StringTokenizer

java.lang.Object

|

+----java.util.StringTokenizer

Declaration

```
public class StringTokenizer
```

```
extends Object
```

```
implements Enumeration
```

Description

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
```

Constructors

[StringTokenizer\(String, String, boolean\)](#)

StringTokenizer(String, String)

StringTokenizer(String)

Methods

countTokens()

hasMoreElements()

hasMoreTokens()

nextElement()

nextToken()

nextToken(String)

StringTokenizer

Applies to

Class `java.util.StringTokenizer`

Declaration

```
public StringTokenizer(String str,  
                      String delim,  
                      boolean returnTokens)  
  
public StringTokenizer(String str,  
                      String delim)  
  
public StringTokenizer(String str)
```

Description

```
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

```
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

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

Applies to

Class `java.util.StringTokenizer`

Declaration

```
public boolean hasMoreTokens()
```

Description

Returns true if more tokens exist.

nextToken

Applies to

Class `java.util.StringTokenizer`

Declaration

```
public String nextToken()  
public String nextToken(String delim)
```

Description

```
public String nextToken()
```

Returns the next token of the String.

Throws: `NoSuchElementException`

If there are no more tokens in the String.

```
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

Applies to

Class `java.util.StringTokenizer`

Declaration

```
public boolean hasMoreElements()
```

Description

Returns true if the Enumeration has more elements.

nextElement

Applies to

Class `java.util.StringTokenizer`

Declaration

```
public Object nextElement()
```

Description

Returns the next element in the Enumeration.

Throws: `NoSuchElementException`

If there are no more elements in the enumeration.

countTokens

Applies to

Class `java.util.StringTokenizer`

Declaration

```
public int countTokens()
```

Description

Returns the next number of tokens in the String using the current delimiter 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.lang.System

java.lang.Object

|

+----java.lang.System

Declaration

```
public final class System
```

```
extends Object
```

Description

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.

Variables

err

in

out

Methods

arraycopy(Object, int, Object, int, int)

currentTimeMillis()

exit(int)

gc()

getProperties()

getProperty(String)

getProperty(String, String)

getSecurityManager()

getenv(String)

load(String)

loadLibrary(String)

runFinalization()

setProperties(Properties)

setSecurityManager(SecurityManager)

in

Applies to

Class java.lang.System

Declaration

```
public static InputStream in
```

Description

Standard input stream. This stream is used for reading in character data.

out

Applies to

Class java.lang.System

Declaration

```
public static PrintStream out
```

Description

Standard output stream. This stream is used for printing messages.

err

Applies to

Class java.lang.System

Declaration

```
public static PrintStream err
```

Description

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

Applies to

Class java.lang.System

Declaration

```
public static void setSecurityManager(SecurityManager s)
```

Description

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

Applies to

Class java.lang.System

Declaration

```
public static SecurityManager getSecurityManager()
```

Description

Gets the system security interface.

currentTimeMillis

Applies to

Class java.lang.System

Declaration

```
public static long currentTimeMillis()
```

Description

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

Applies to

Class java.lang.System

Declaration

```
public static void arraycopy(Object src,  
                             int src_position,  
                             Object dst,  
                             int dst_position,  
                             int length)
```

Description

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

Applies to

Class java.lang.System

Declaration

```
public static Properties getProperties()
```

Description

Gets the System properties.

setPropertyies

Applies to

Class java.lang.System

Declaration

```
public static void setPropertyies(Properties props)
```

Description

Sets the System properties to the specified properties.

Parameters:

props - the properties to be set

getProperty

Applies to

Class java.lang.System

Declaration

```
public static String getProperty(String key)
public static String getProperty(String key,
                                String def)
```

Description

```
public static String getProperty(String key)
```

Gets the System property indicated by the specified key.

Parameters:

key - the name of the system property

```
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

Applies to

Class `java.lang.System`

Declaration

```
public static String getenv(String name)
```

Description

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

Applies to

Class java.lang.System

Declaration

```
public static void exit(int status)
```

Description

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

Applies to

Class java.lang.System

Declaration

```
public static void gc()
```

Description

Runs the garbage collector.

See Also:

gc

runFinalization

Applies to

Class java.lang.System

Declaration

```
public static void runFinalization()
```

Description

Runs the finalization methods of any objects pending finalization.

See Also:

gc

load

Applies to

Class java.lang.System

Declaration

```
public static void load(String filename)
```

Description

Loads a dynamic library, given a complete path name.

Parameters:

filename - the file to load

Throws: UnsatisfiedLinkError

If the file does not exist.

loadLibrary

Applies to

lass java.lang.System

Declaration

```
public static void loadLibrary(String libname)
```

Description

Loads a dynamic library with the specified library name.

Parameters:

libname - the name of the library

Throws: UnsatisfiedLinkError

If the library does not exist.

Interface `java.awt.peer.TextAreaPeer`

Declaration

```
public interface TextAreaPeer  
    extends Object  
    extends TextComponentPeer
```

Methods

```
insertText(String, int)  
minimumSize(int, int)  
preferredSize(int, int)  
replaceText(String, int, int)
```

insertText

Applies to

Interface `java.awt.peer.TextAreaPeer`

Declaration

```
public abstract void insertText(String txt,  
                                int pos)
```

replaceText

Applies to

Interface java.awt.peer.TextAreaPeer

Declaration

```
public abstract void replaceText(String txt,  
                                int start,  
                                int end)
```


preferredSize

Applies to

Interface `java.awt.peer.TextAreaPeer`

Declaration

```
public abstract Dimension preferredSize(int rows,  
                                       int cols)
```

minimumSize

Applies to

Interface `java.awt.peer.TextAreaPeer`

Declaration

```
public abstract Dimension minimumSize(int rows,  
                                       int cols)
```

Interface `java.awt.peer.TextComponentPeer`

Declaration

```
public interface TextComponentPeer  
    extends Object  
    extends ComponentPeer
```

Methods

```
getSelectionEnd()  
getSelectionStart()  
getText()  
select(int, int)  
setEditable(boolean)  
setText(String)
```

setEditable

Applies to

Interface `java.awt.peer.TextComponentPeer`

Declaration

```
public abstract void setEditable(boolean editable)
```

getText

Applies to

Interface `java.awt.peer.TextComponentPeer`

Declaration

```
public abstract String getText()
```

setText

Applies to

Interface `java.awt.peer.TextComponentPeer`

Declaration

```
public abstract void setText(String l)
```

getSelectionStart

Applies to

Interface `java.awt.peer.TextComponentPeer`

Declaration

```
public abstract int getSelectionStart()
```

getSelectionEnd

Applies to

Interface `java.awt.peer.TextComponentPeer`

Declaration

```
public abstract int getSelectionEnd()
```


select

Applies to

Interface `java.awt.peer.TextComponentPeer`

Declaration

```
public abstract void select(int selStart,  
                             int selEnd)
```

Interface `java.awt.peer.TextFieldPeer`

Declaration

```
public interface TextFieldPeer  
    extends Object  
    extends TextComponentPeer
```

Methods

[minimumSize\(int\)](#)

[preferredSize\(int\)](#)

[setEchoCharacter\(char\)](#)

setEchoCharacter

Applies to

Interface `java.awt.peer.TextFieldPeer`

Declaration

```
public abstract void setEchoCharacter(char c)
```

preferredSize

Applies to

Interface `java.awt.peer.TextFieldPeer`

Declaration

```
public abstract Dimension preferredSize(int cols)
```

minimumSize

Applies to

Interface `java.awt.peer.TextFieldPeer`

Declaration

```
public abstract Dimension minimumSize(int cols)
```

Class java.lang.Thread

java.lang.Object

|

+----java.lang.Thread

Declaration

```
public class Thread
extends Object
implements Runnable
```

Description

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

Variables

MAX_PRIORITY

MIN_PRIORITY

NORM_PRIORITY

Constructors

Thread()

Thread(Runnable)

Thread(ThreadGroup, Runnable)

Thread(String)

Thread(ThreadGroup, String)

Thread(Runnable, String)

Thread(ThreadGroup, Runnable, String)

Methods

activeCount()

checkAccess()

countStackFrames()

currentThread()

destroy()

dumpStack()

enumerate(Thread[])

getName()

getPriority()

getThreadGroup()

interrupt()

interrupted()

isAlive()

isDaemon()

isInterrupted()

join(long)

join(long, int)

join()

resume()

run()

setDaemon(boolean)

setName(String)

setPriority(int)

sleep(long)

sleep(long, int)

start()

stop()

stop(Throwable)

suspend()

toString()

yield()

MIN_PRIORITY

Applies to

Class `java.lang.Thread`

Declaration

```
public final static int MIN_PRIORITY
```

Description

The minimum priority that a Thread can have. The most minimal priority is equal to 1.

NORM_PRIORITY

Applies to

Class java.lang.Thread

Declaration

```
public final static int NORM_PRIORITY
```

Description

The default priority that is assigned to a Thread. The default priority is equal to 5.

MAX_PRIORITY

Applies to

Class `java.lang.Thread`

Declaration

```
public final static int MAX_PRIORITY
```

Description

The maximum priority that a Thread can have. The maximal priority value a Thread can have is 10.

Thread

Applies to

Class java.lang.Thread

Declaration

```
public Thread()  
public Thread(Runnable target)  
public Thread(ThreadGroup group,  
               Runnable target)  
public Thread(String name)  
public Thread(ThreadGroup group,  
               String name)  
public Thread(Runnable target,  
               String name)  
public Thread(ThreadGroup group,  
               Runnable target,  
               String name)
```

Description

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++;
        }
    }
}

```

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

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

public Thread(String name)

Constructs a new Thread with the specified name.

Parameters:

name - the name of the new 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

**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

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

Applies to

Class java.lang.Thread

Declaration

```
public static Thread currentThread()
```

Description

Returns a reference to the currently executing Thread object.

yield

Applies to

Class java.lang.Thread

Declaration

```
public static void yield()
```

Description

Causes the currently executing Thread object to yield. If there are other runnable Threads they will be scheduled next.

sleep

Applies to

Class java.lang.Thread

Declaration

```
public static void sleep(long millis) throws InterruptedException  
public static void sleep(long millis,  
                           int nanos) throws InterruptedException
```

Description

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.

**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

Applies to

Class `java.lang.Thread`

Declaration

```
public synchronized void start()
```

Description

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

Applies to

Class `java.lang.Thread`

Declaration

```
public void run()
```

Description

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

Applies to

Class `java.lang.Thread`

Declaration

```
public final void stop()  
public final synchronized void stop(Throwable o)
```

Description

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.

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

Applies to

Class `java.lang.Thread`

Declaration

```
public void interrupt()
```

Description

Send an interrupt to a thread.

interrupted

Applies to

Class `java.lang.Thread`

Declaration

```
public static boolean interrupted()
```

Description

Ask if you have been interrupted.

isInterrupted

Applies to

Class java.lang.Thread

Declaration

```
public boolean isInterrupted()
```

Description

Ask if another thread has been interrupted.

destroy

Applies to

Class `java.lang.Thread`

Declaration

```
public void destroy()
```

Description

Destroy a thread, without any cleanup, i.e. just toss its state; any monitors it has locked remain locked. A last resort.

isAlive

Applies to

Class `java.lang.Thread`

Declaration

```
public final boolean isAlive()
```

Description

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

Applies to

Class java.lang.Thread

Declaration

```
public final void suspend()
```

Description

Suspends this Thread's execution.

resume

Applies to

Class java.lang.Thread

Declaration

```
public final void resume()
```

Description

Resumes this Thread execution. This method is only valid after suspend() has been invoked.

setPriority

Applies to

Class `java.lang.Thread`

Declaration

```
public final void setPriority(int newPriority)
```

Description

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

Applies to

Class java.lang.Thread

Declaration

```
public final int getPriority()
```

Description

Gets and returns the Thread's priority.

See Also:

setPriority

setName

Applies to

Class java.lang.Thread

Declaration

```
public final void setName(String name)
```

Description

Sets the Thread's name.

Parameters:

name - the new name of the Thread

See Also:

getName

getName

Applies to

Class `java.lang.Thread`

Declaration

```
public final String getName()
```

Description

Gets and returns this Thread's name.

See Also:

setName

getThreadGroup

Applies to

Class `java.lang.Thread`

Declaration

```
public final ThreadGroup getThreadGroup()
```

Description

Gets and returns this Thread group.

activeCount

Applies to

Class java.lang.Thread

Declaration

```
public static int activeCount()
```

Description

Returns the current number of active Threads in this Thread group.

enumerate

Applies to

Class `java.lang.Thread`

Declaration

```
public static int enumerate(Thread tarray[])
```

Description

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

Applies to

Class `java.lang.Thread`

Declaration

```
public int countStackFrames()
```

Description

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

Applies to

Class `java.lang.Thread`

Declaration

```
public final synchronized void join(long millis) throws InterruptedException  
public final synchronized void join(long millis,  
                                     int nanos) throws InterruptedException  
public final void join() throws InterruptedException
```

Description

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.

**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.

public final void join() throws InterruptedException

Waits forever for this Thread to die.

Throws: InterruptedException

Another thread has interrupted this thread.

dumpStack

Applies to

Class java.lang.Thread

Declaration

```
public static void dumpStack()
```

Description

A debugging procedure to print a stack trace for the current Thread.

See Also:

printStackTrace

setDaemon

Applies to

Class java.lang.Thread

Declaration

```
public final void setDaemon(boolean on)
```

Description

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

Applies to

Class `java.lang.Thread`

Declaration

```
public final boolean isDaemon()
```

Description

Returns the daemon flag of the Thread.

See Also:

`setDaemon`

checkAccess

Applies to

Class java.lang.Thread

Declaration

```
public void checkAccess()
```

Description

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

Applies to

Class `java.lang.Thread`

Declaration

```
public String toString()
```

Description

Returns a String representation of the Thread, including the thread's name, priority and thread group.

Overrides:

toString in class Object

Class java.lang.ThreadGroup

java.lang.Object

|

+----java.lang.ThreadGroup

Declaration

```
public class ThreadGroup
extends Object
```

Description

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.

Constructors

[ThreadGroup\(String\)](#)

[ThreadGroup\(ThreadGroup, String\)](#)

Methods

[activeCount\(\)](#)

[activeGroupCount\(\)](#)

[checkAccess\(\)](#)

[destroy\(\)](#)

[enumerate\(Thread\[\]\)](#)

[enumerate\(Thread\[\], boolean\)](#)

[enumerate\(ThreadGroup\[\]\)](#)

[enumerate\(ThreadGroup\[\], boolean\)](#)

[getMaxPriority\(\)](#)

[getName\(\)](#)

[getParent\(\)](#)

isDaemon()

list()

parentOf(ThreadGroup)

resume()

setDaemon(boolean)

setMaxPriority(int)

stop()

suspend()

toString()

uncaughtException(Thread, Throwable)

ThreadGroup

Applies to

Class java.lang.ThreadGroup

Declaration

```
public ThreadGroup(String name)
public ThreadGroup(ThreadGroup parent,
                   String name)
```

Description

```
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

```
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

Applies to

Class java.lang.ThreadGroup

Declaration

```
public final String getName()
```

Description

Gets the name of this Thread group.

getParent

Applies to

Class `java.lang.ThreadGroup`

Declaration

```
public final ThreadGroup getParent()
```

Description

Gets the parent of this Thread group.

getMaxPriority

Applies to

Class `java.lang.ThreadGroup`

Declaration

```
public final int getMaxPriority()
```

Description

Gets the maximum priority of the group. Threads that are part of this group cannot have a higher priority than the maximum priority.

isDaemon

Applies to

Class `java.lang.ThreadGroup`

Declaration

```
public final boolean isDaemon()
```

Description

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

Applies to

Class java.lang.ThreadGroup

Declaration

```
public final void setDaemon(boolean daemon)
```

Description

Changes the daemon status of this group.

Parameters:

daemon - the daemon boolean which is to be set.

setMaxPriority

Applies to

Class `java.lang.ThreadGroup`

Declaration

```
public final synchronized void setMaxPriority(int pri)
```

Description

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

Applies to

Class `java.lang.ThreadGroup`

Declaration

```
public final boolean parentOf(ThreadGroup g)
```

Description

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

Applies to

Class java.lang.ThreadGroup

Declaration

```
public final void checkAccess()
```

Description

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

Applies to

Class java.lang.ThreadGroup

Declaration

```
public synchronized int activeCount()
```

Description

Returns an estimate of the number of active Threads in the Thread group.

enumerate

Applies to

Class `java.lang.ThreadGroup`

Declaration

```
public int enumerate(Thread list[])  
public int enumerate(Thread list[],  
                     boolean recurse)  
public int enumerate(ThreadGroup list[])  
public int enumerate(ThreadGroup list[],  
                     boolean recurse)
```

Description

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

**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 reappeared

Returns:

the number of Threads placed into the array.

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.

**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.

activeGroupCount

Applies to

Class `java.lang.ThreadGroup`

Declaration

```
public synchronized int activeGroupCount()
```

Description

Returns an estimate of the number of active groups in the Thread group.

stop

Applies to

Class `java.lang.ThreadGroup`

Declaration

```
public final synchronized void stop()
```

Description

Stops all the Threads in this Thread group and all of its sub groups.

suspend

Applies to

Class java.lang.ThreadGroup

Declaration

```
public final synchronized void suspend()
```

Description

Suspends all the Threads in this Thread group and all of its sub groups.

resume

Applies to

Class `java.lang.ThreadGroup`

Declaration

```
public final synchronized void resume()
```

Description

Resumes all the Threads in this Thread group and all of its sub groups.

destroy

Applies to

Class `java.lang.ThreadGroup`

Declaration

```
public final synchronized void destroy()
```

Description

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

Applies to

Class `java.lang.ThreadGroup`

Declaration

```
public synchronized void list()
```

Description

Lists this Thread group. Useful for debugging only.

uncaughtException

Applies to

Class `java.lang.ThreadGroup`

Declaration

```
public void uncaughtException(Thread t,  
                               Throwable e)
```

Description

Called when a thread in this group exists because of an uncaught exception.

toString

Applies to

Class `java.lang.ThreadGroup`

Declaration

```
public String toString()
```

Description

Returns a String representation of the Thread group.

Overrides:

toString in class Object

Class java.lang.Throwable

java.lang.Object

|

+----java.lang.Throwable

Declaration

```
public class Throwable
```

```
extends Object
```

Description

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();
}
```

Constructors

[Throwable\(\)](#)

[Throwable\(String\)](#)

Methods

[fillInStackTrace\(\)](#)

[getMessage\(\)](#)

[printStackTrace\(\)](#)

printStackTrace(PrintStream)

toString()

Throwable

Applies to

Class `java.lang.Throwable`

Declaration

```
public Throwable()  
public Throwable(String message)
```

Description

public Throwable()

Constructs a new Throwable with no detail message. The stack trace is automatically filled in.

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

Applies to

Class `java.lang.Throwable`

Declaration

```
public String getMessage()
```

Description

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

Applies to

Class java.lang.Throwable

Declaration

```
public String toString()
```

Description

Returns a short description of the Throwable.

Overrides:

toString in class Object

printStackTrace

Applies to

Class `java.lang.Throwable`

Declaration

```
public void printStackTrace()  
public void printStackTrace(PrintStream s)
```

Description

```
public void printStackTrace()
```

Prints the Throwable and the Throwable's stack trace.

```
public void printStackTrace(PrintStream s)
```

fillInStackTrace

Applies to

Class `java.lang.Throwable`

Declaration

```
public Throwable fillInStackTrace()
```

Fills in the execution 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.Error

java.lang.Object

|

+----java.lang.Throwable

|

+----java.lang.Error

Declaration

```
public class Error
    extends Throwable
```

Description

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.

Constructors

Error()

Error(String)

Error

Applies to

Class `java.lang.Error`

Declaration

```
public Error()  
public Error(String s)
```

Description

public Error()

Constructs an Error with no specified detail message. A detail message is a String that describes this particular error.

public Error(String s)

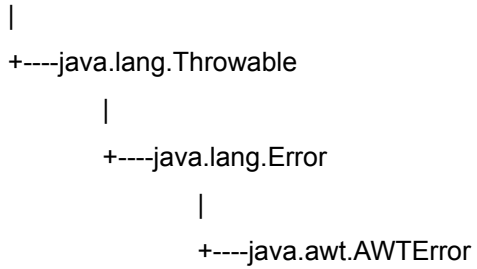
Constructs an Error with the specified detail message. A detail message is a String that describes this particular error

Parameters:

s - the detail message

Class java.awt.AWTErrors

java.lang.Object



Declaration

```
public class AWTErrors
    extends Error
```

Description

An AWT Error.

Constructors

[AWTErrors\(String\)](#)

AWTError

Applies to

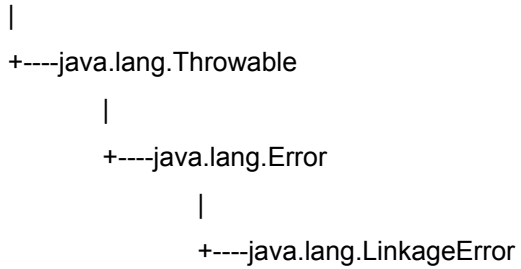
Class java.awt.AWTError

Declaration

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

Class java.lang.LinkageError

java.lang.Object



Declaration

```
public class LinkageError
    extends Error
```

Description

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.

Constructors

[LinkageError\(\)](#)

[LinkageError\(String\)](#)

LinkageError

Applies to

Declaration

```
public LinkageError()  
public LinkageError(String s)
```

Description

public LinkageError()

Constructs a LinkageError with no detail message. A detail message is a String that describes this particular exception.

public LinkageError(String s)

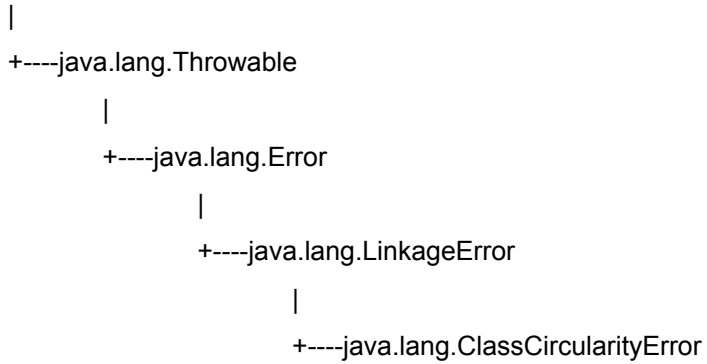
Constructs a LinkageError with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class `java.lang.ClassCircularityError`

`java.lang.Object`



Declaration

```
public class ClassCircularityError
    extends LinkageError
```

Description

Signals that a circularity has been detected when initializing a class.

Constructors

[ClassCircularityError\(\)](#)

[ClassCircularityError\(String\)](#)

ClassCircularityError

Applies to

Class `java.lang.ClassCircularityError`

Declaration

```
public ClassCircularityError()  
public ClassCircularityError(String s)
```

Description

public ClassCircularityError()

Constructs a ClassCircularityError with no detail message. A detail message is a String that describes this particular exception.

public ClassCircularityError(String s)

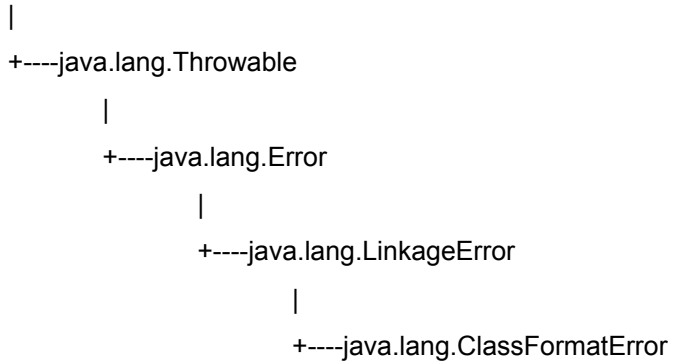
Constructs a ClassCircularityError with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class `java.lang.ClassFormatError`

`java.lang.Object`



Declaration

```
public class ClassFormatError
    extends LinkageError
```

Description

Signals an invalid file format has occurred.

Constructors

[ClassFormatError\(\)](#)

[ClassFormatError\(String\)](#)

ClassFormatError

Applies to

Class `java.lang.ClassFormatError`

Declaration

```
public ClassFormatError()  
public ClassFormatError()
```

Description

public ClassFormatError()

Constructs a ClassFormatError with no detail message. A detail message is a String that describes this particular exception.

public ClassFormatError(String s)

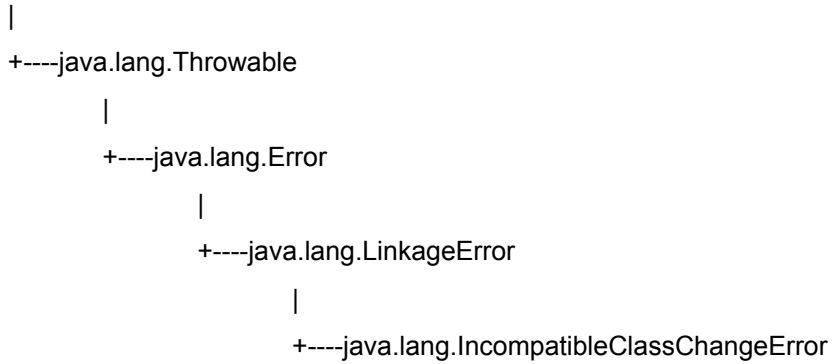
Constructs a ClassFormatError with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the String containing the detail message

Class `java.lang.IncompatibleClassChangeError`

`java.lang.Object`



Declaration

```
public class IncompatibleClassChangeError
    extends LinkageError
```

Description

Signals that an incompatible class change has occurred.

Constructors

[IncompatibleClassChangeError\(\)](#)

[IncompatibleClassChangeError\(String\)](#)

IncompatibleClassChangeError

Applies to

Class `java.lang.IncompatibleClassChangeError`

Declaration

```
public IncompatibleClassChangeError()  
public IncompatibleClassChangeError(String s)
```

Description

```
public IncompatibleClassChangeError()
```

Constructs an `IncompatibleClassChangeError` with no detail message. A detail message is a `String` that describes this particular exception.

```
public IncompatibleClassChangeError(String s)
```

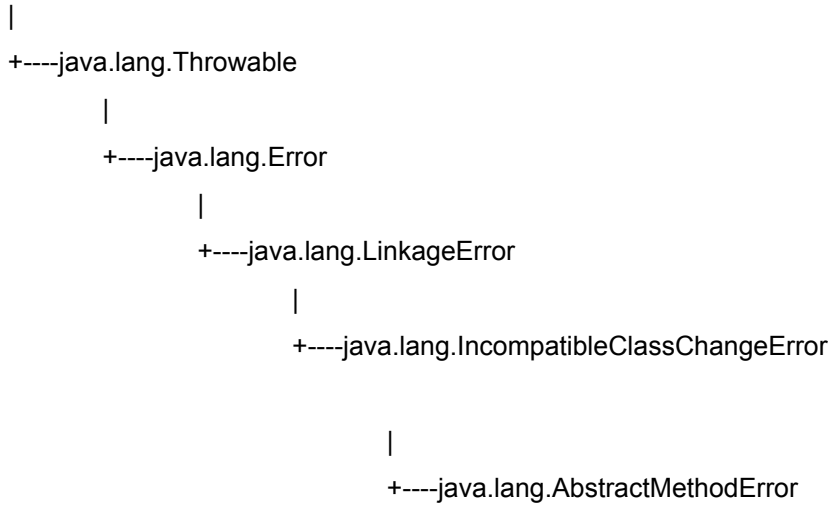
Constructs an `IncompatibleClassChangeError` with the specified detail message. A detail message is a `String` that describes this particular exception.

Parameters:

s - the detail message

Class java.lang.AbstractMethodError

java.lang.Object



Declaration

```
public class AbstractMethodError
    extends IncompatibleClassChangeError
```

Description

Signals an attempt to call an abstract method.

Constructors

[AbstractMethodError\(\)](#)

[AbstractMethodError\(String\)](#)

AbstractMethodError

Applies to

Class `java.lang.AbstractMethodError`

Declaration

```
public AbstractMethodError()  
public AbstractMethodError(String s)
```

Description

```
public AbstractMethodError()
```

Constructs an `AbstractMethodError` with no detail message. A detail message is a `String` that describes this particular exception.

```
public AbstractMethodError(String s)
```

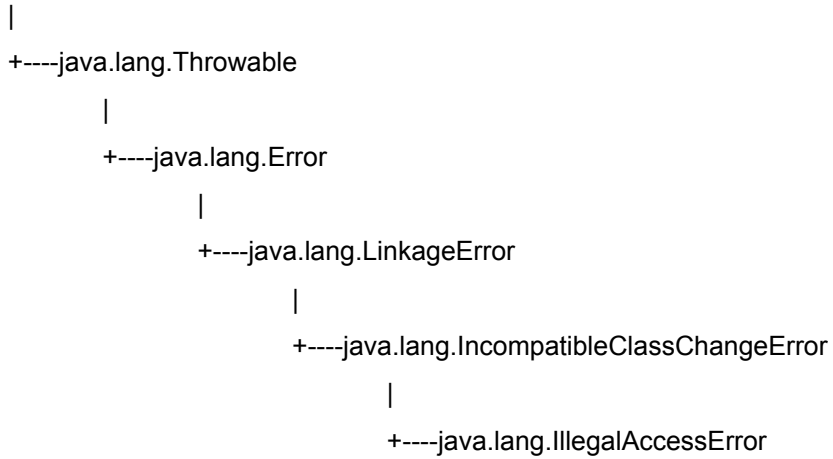
Constructs an `AbstractMethodError` with the specified detail message. A detail message is a `String` that describes this particular exception.

Parameters:

s - the `String` that contains the detail message

Class java.lang.IllegalAccessError

java.lang.Object



Declaration

```
public class IllegalAccessError
    extends IncompatibleClassChangeError
```

Description

Signals that an illegal access exception has occurred.

Constructors

[IllegalAccessError\(\)](#)

[IllegalAccessError\(String\)](#)

IllegalAccessError

Applies to

Class `java.lang.IllegalAccessError`

Declaration

```
public IllegalAccessError()  
public IllegalAccessError(String s)
```

Description

public IllegalAccessError()

Constructs an `IllegalAccessError` with no detail message. A detail message is a `String` that describes this particular exception.

public IllegalAccessError(String s)

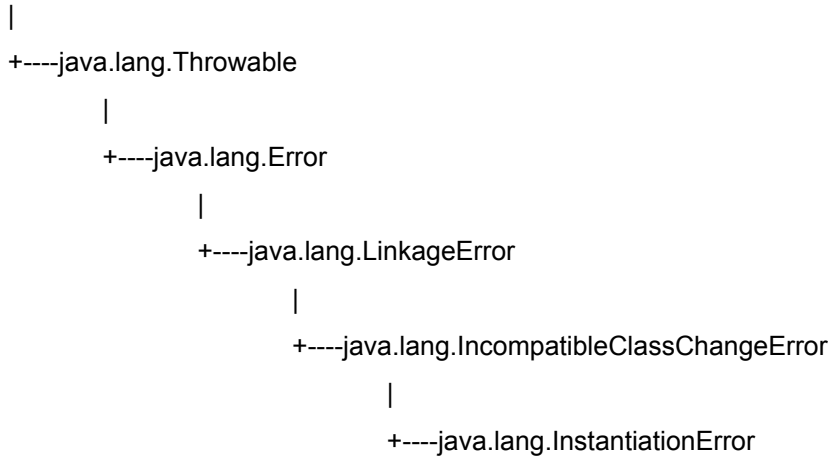
Constructs an `IllegalAccessError` with the specified detail message. A detail message is a `String` that describes this particular exception.

Parameters:

s - the detail message

Class java.lang.InstantiationError

java.lang.Object



Declaration

```
public class InstantiationError
    extends IncompatibleClassChangeError
```

Description

Signals that the interpreter has tried to instantiate an abstract class or an interface.

Constructors

[InstantiationError\(\)](#)

[InstantiationError\(String\)](#)

InstantiationError

Applies to

Class `java.lang.InstantiationError`

Declaration

```
public InstantiationError()  
public InstantiationError(String s)
```

Description

public InstantiationError()

Constructs an `InstantiationError` with no detail message. A detail message is a `String` that describes this particular exception.

public InstantiationError(String s)

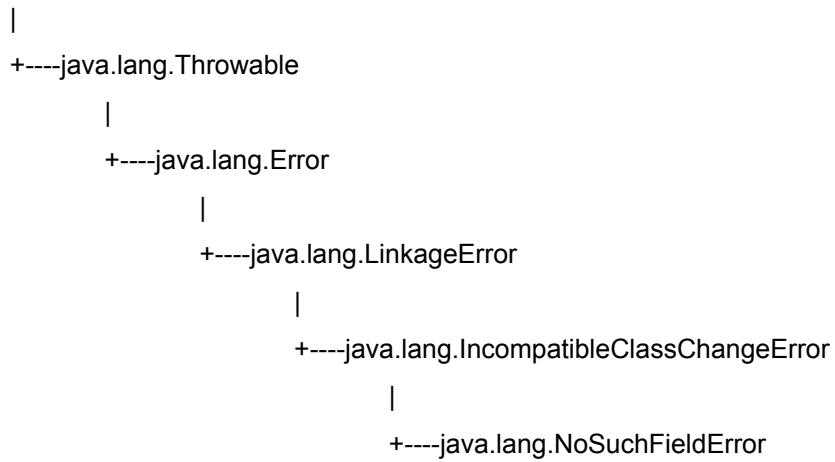
Constructs an `InstantiationError` with the specified detail message. A detail message is a `String` that describes this particular exception.

Parameters:

s - the `String` that contains the detail message

Class java.lang.NoSuchFieldError

java.lang.Object



Declaration

```
public class NoSuchFieldError
    extends IncompatibleClassChangeError
```

Description

Signals that a particular field could not be found.

Constructors

[NoSuchFieldError\(\)](#)

[NoSuchFieldError\(String\)](#)

NoSuchFieldError

Applies to

java.lang.NoSuchFieldError

Declaration

```
public NoSuchFieldError()  
public NoSuchFieldError(String s)
```

Description

```
public NoSuchFieldError()
```

Constructs a NoSuchFieldException without a detail message. A detail message is a String that describes this particular exception.

```
public NoSuchFieldError(String s)
```

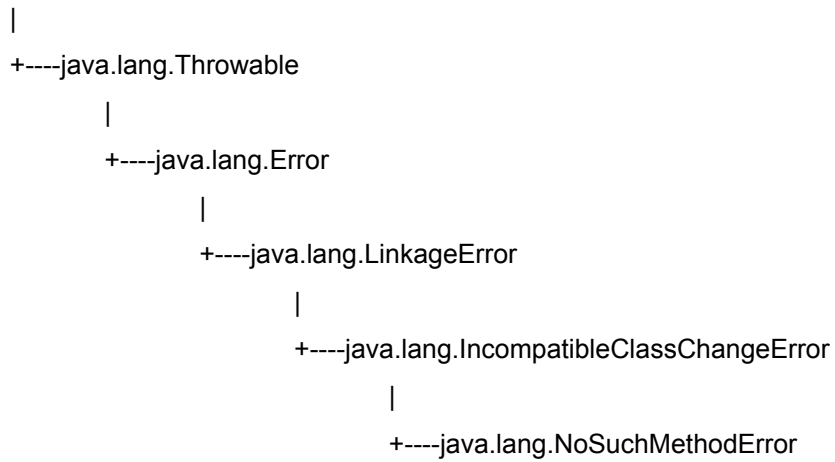
Constructs a NoSuchFieldException with a detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class java.lang.NoSuchMethodError

java.lang.Object



Declaration

```
public class NoSuchMethodError
    extends IncompatibleClassChangeError
```

Description

Signals that a particular method could not be found.

Constructors

[NoSuchMethodError\(\)](#)

[NoSuchMethodError\(String\)](#)

NoSuchMethodError

Applies to

Class `java.lang.NoSuchMethodError`

Declaration

```
public NoSuchMethodError()  
public NoSuchMethodError(String s)
```

Description

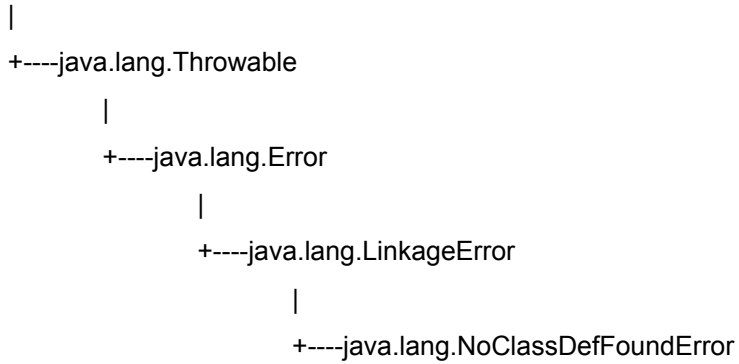
Constructs a `NoSuchMethodException` with a detail message. A detail message is a `String` that describes this particular exception.

Parameters:

s - the detail message

Class java.lang.NoClassDefFoundError

java.lang.Object



Declaration

```
public class NoClassDefFoundError
    extends LinkageError
```

DescriptionDescription

Signals that a class could not be found.

Constructors

[NoClassDefFoundError\(\)](#)

[NoClassDefFoundError\(String\)](#)

NoClassDefFoundError

Applies to

Class `java.lang.NoClassDefFoundError`

Declaration

```
public NoClassDefFoundError()  
public NoClassDefFoundError(String s)
```

Description

public NoClassDefFoundError()

Constructs a NoClassDefFoundError with no detail message. A detail message is a String that describes this particular exception.

public NoClassDefFoundError(String s)

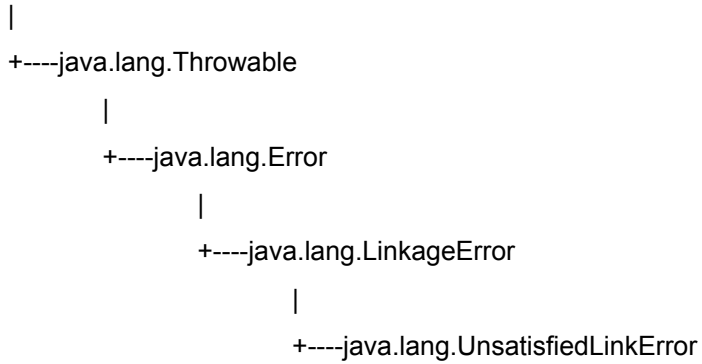
Constructs a NoClassDefFoundError with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class `java.lang.UnsatisfiedLinkError`

`java.lang.Object`



Declaration

```
public class UnsatisfiedLinkError
    extends LinkageError
```

Description

Signals an unsatisfied link.

See Also:

Runtime

Constructors

[UnsatisfiedLinkError\(\)](#)

[UnsatisfiedLinkError\(String\)](#)

UnsatisfiedLinkError

Applies to

Class `java.lang.UnsatisfiedLinkError`

Declaration

```
public UnsatisfiedLinkError()  
public UnsatisfiedLinkError(String s)
```

Description

public UnsatisfiedLinkError()

Constructs an `UnsatisfiedLinkError` with no detail message. A detail message is a `String` that describes this particular exception.

public UnsatisfiedLinkError(String s)

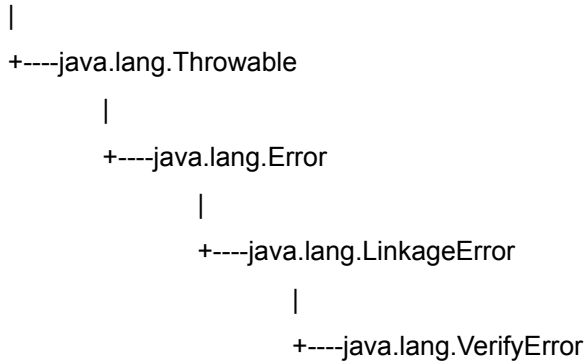
Constructs an `UnsatisfiedLinkError` with the specified detail message. A detail message is a `String` that describes this particular exception.

Parameters:

s - the detail message

Class java.lang.VerifyError

java.lang.Object



Declaration

```
public class VerifyError
    extends LinkageError
```

Description

Signals that a Verification Error occurred.

Constructors

[VerifyError\(\)](#)

[VerifyError\(String\)](#)

VerifyError

Applies to

Class `java.lang.VerifyError`

Declaration

```
public VerifyError()  
public VerifyError(String s)
```

Description

```
public VerifyError()
```

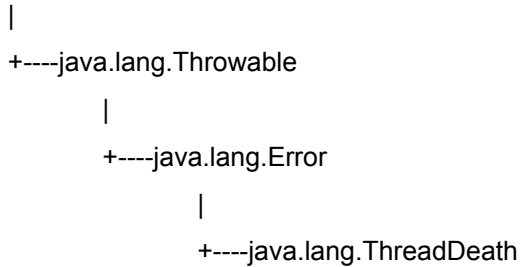
Constructor.

```
public VerifyError(String s)
```

Constructor with a detail message.

Class java.lang.ThreadDeath

java.lang.Object



Declaration

```
public class ThreadDeath
    extends Error
```

Description

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.

Constructors

ThreadDeath()

ThreadDeath

Applies to

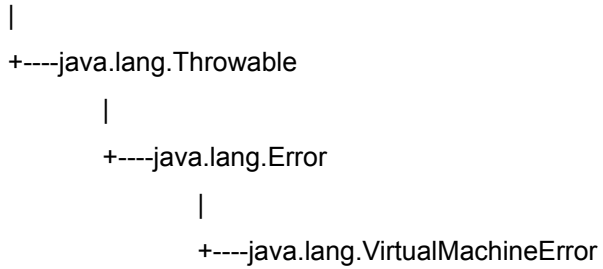
Class `java.lang.ThreadDeath`

Declaration

```
public ThreadDeath()
```


Class java.lang.VirtualMachineError

java.lang.Object



Declaration

```
public class VirtualMachineError
    extends Error
```

Description

A `VirtualMachineError` indicates that the virtual machine is broken or has run out of resources.

Constructors

[VirtualMachineError\(\)](#)

[VirtualMachineError\(String\)](#)

VirtualMachineError

Applies to

Class `java.lang.VirtualMachineError`

Declaration

```
public VirtualMachineError()  
public VirtualMachineError(String s)
```

Description

public VirtualMachineError()

Constructs a VirtualMachineError with no detail message. A detail message is a String that describes this particular exception.

public VirtualMachineError(String s)

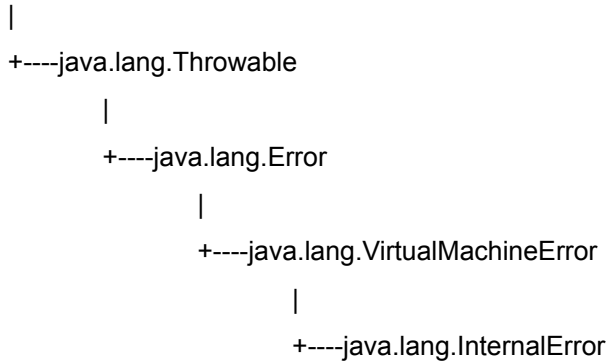
Constructs a VirtualMachineError with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class java.lang.InternalError

java.lang.Object



Declaration

```
public class InternalError
    extends VirtualMachineError
```

Description

Signals that an internal error has occurred.

Constructors

[InternalError\(\)](#)

[InternalError\(String\)](#)

InternalError

Applies to

Class `java.lang.InternalError`

Declaration

```
public InternalError()  
public InternalError(String s)
```

Description

public InternalError()

Constructs an InternalError with no detail message. A detail message is a String that describes this particular exception.

public InternalError(String s)

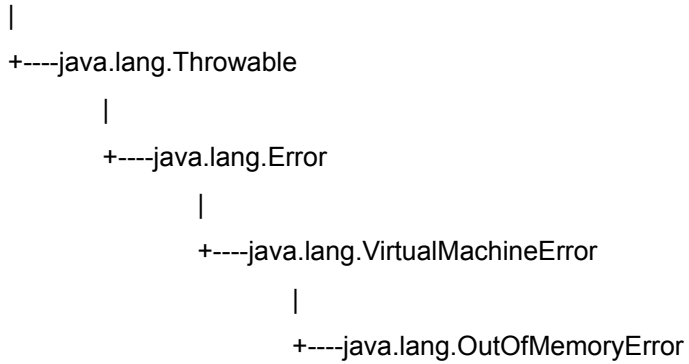
Constructs an InternalError with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class java.lang.OutOfMemoryError

java.lang.Object



Declaration

```
public class OutOfMemoryError
    extends VirtualMachineError
```

Description

Signals that you are out of memory.

Constructors

[OutOfMemoryError\(\)](#)

[OutOfMemoryError\(String\)](#)

OutOfMemoryError

Applies to

Class `java.lang.OutOfMemoryError`

Declaration

```
public OutOfMemoryError()  
public OutOfMemoryError(String s)
```

Description

```
public OutOfMemoryError()
```

Constructs an `OutOfMemoryError` with no detail message. A detail message is a `String` that describes this particular exception.

```
public OutOfMemoryError(String s)
```

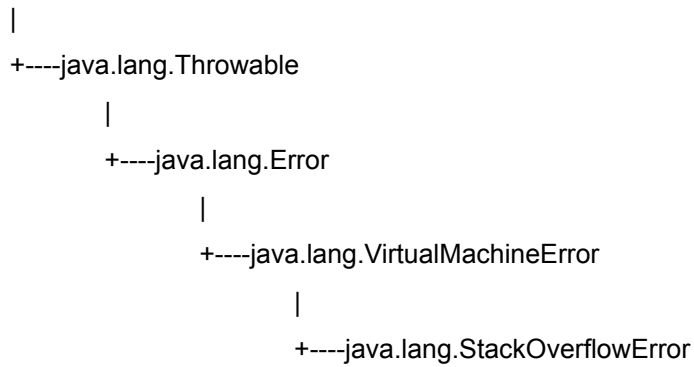
Constructs an `OutOfMemoryError` with the specified detail message. A detail message is a `String` that describes this particular exception.

Parameters:

s - the detail message

Class java.lang.StackOverflowError

java.lang.Object



Declaration

```
public class StackOverflowError
    extends VirtualMachineError
```

Description

Signals that a stack overflow has occurred.

Constructors

[StackOverflowError\(\)](#)

[StackOverflowError\(String\)](#)

StackOverflowError

Applies to

Class `java.lang.StackOverflowError`

Declaration

```
public StackOverflowError()  
public StackOverflowError(String s)
```

Description

```
public StackOverflowError()
```

Constructs a `StackOverflowError` with no detail message. A detail message is a `String` that describes this particular exception.

```
public StackOverflowError(String s)
```

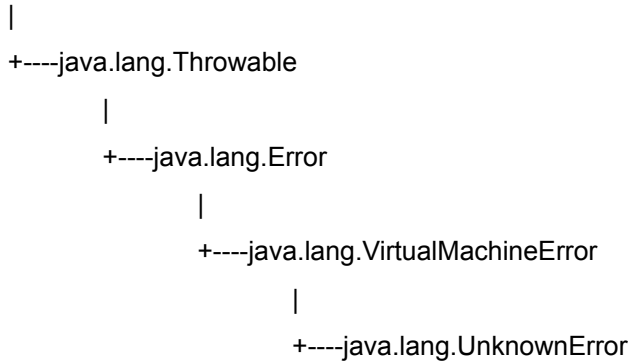
Constructs a `StackOverflowError` with the specified detail message. A detail message is a `String` that describes this particular exception.

Parameters:

s - the detail message

Class java.lang.UnknownError

java.lang.Object



Declaration

```
public class UnknownError
    extends VirtualMachineError
```

Description

Signals that an unknown but serious exception has occurred.

Constructors

[UnknownError\(\)](#)

[UnknownError\(String\)](#)

UnknownError

Applies to

Class `java.lang.UnknownError`

Declaration

```
public UnknownError()  
public UnknownError(String s)
```

Description

public UnknownError()

Constructs an UnknownError with no detail message. A detail message is a String that describes this particular exception.

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.Exception

java.lang.Object

|

+----java.lang.Throwable

|

+----java.lang.Exception

Declaration

```
public class Exception
    extends Throwable
```

Description

Exception are a form of Throwable that normal programs may wish to try and catch.

Constructors

[Exception\(\)](#)

[Exception\(String\)](#)

Exception

Applies to

Class `java.lang.Exception`

Declaration

```
public Exception()  
public Exception(String s)
```

Description

public Exception()

Constructs an Exception with no specified detail message. A detail message is a String that describes this particular exception.

public Exception(String s)

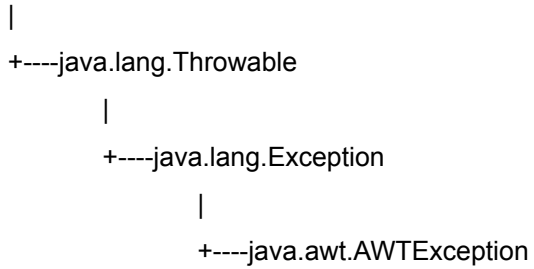
Constructs a Exception with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class java.awt.AWTException

java.lang.Object



Declaration

```
public class AWTException
    extends Exception
```

Description

Signals that an Abstract Window Toolkit exception has occurred.

Constructors

[AWTException\(String\)](#)

AWTException

Applies to

Class java.awt.AWTException

Declaration

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

Description

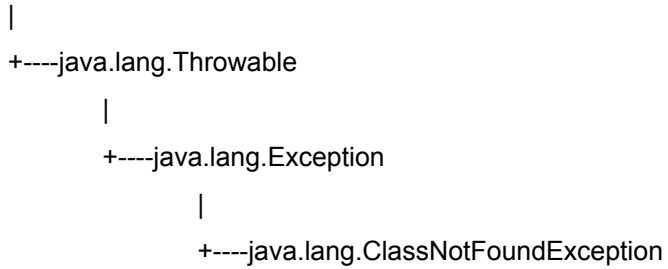
Constructs an AWTException with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

msg - the detail message

Class java.lang.ClassNotFoundException

java.lang.Object



Declaration

```
public class ClassNotFoundException
    extends Exception
```

Description

Signals that a class could not be found.

Constructors

[ClassNotFoundException\(\)](#)

[ClassNotFoundException\(String\)](#)

ClassNotFoundException

Applies to

Class `java.lang.ClassNotFoundException`

Declaration

```
public ClassNotFoundException()  
public ClassNotFoundException(String s)
```

Description

public `ClassNotFoundException()`

Constructs a `ClassNotFoundException` with no detail message. A detail message is a `String` that describes this particular exception.

public `ClassNotFoundException(String s)`

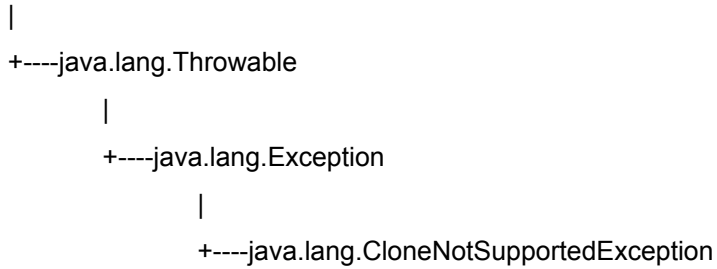
Constructs a `ClassNotFoundException` with the specified detail message. A detail message is a `String` that describes this particular exception.

Parameters:

s - the detail message

Class java.lang.CloneNotSupportedException

java.lang.Object



Declaration

```
public class CloneNotSupportedException
    extends Exception
```

Description

Signals that an attempt has been made to clone an object that does not want to be cloned.

Constructors

[CloneNotSupportedException\(\)](#)

[CloneNotSupportedException\(String\)](#)

CloneNotSupportedException

Applies to

Class `java.lang.CloneNotSupportedException`

Declaration

```
public CloneNotSupportedException()  
public CloneNotSupportedException(String s)
```

Description

```
public CloneNotSupportedException()
```

Constructs an `CloneNotSupportedException` with no detail message. A detail message is a `String` that describes this particular exception.

```
public CloneNotSupportedException(String s)
```

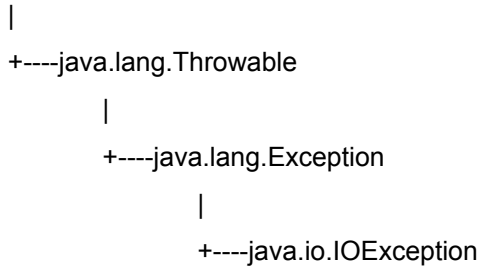
Constructs an `CloneNotSupportedException` with the specified detail message. A detail message is a `String` that describes this particular exception.

Parameters:

s - the detail message

Class java.io.IOException

java.lang.Object



Declaration

```
public class IOException
    extends Exception
```

Description

Signals that an I/O exception has occurred.

See Also:

InputStream, OutputStream

Constructors

[IOException\(\)](#)

[IOException\(String\)](#)

IOException

Applies to

Class java.io.IOException

Declaration

```
public IOException()  
public IOException(String s)
```

Description

```
public IOException()
```

Constructs an IOException with no detail message. A detail message is a String that describes this particular exception.

```
public IOException(String s)
```

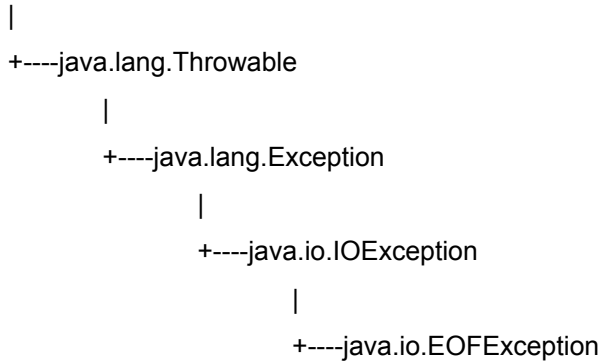
Constructs an IOException with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class java.io.EOFException

java.lang.Object



Declaration

```
public class EOFException
    extends IOException
```

Description

Signals that an EOF has been reached unexpectedly during input.

See Also:

`IOException`, `DataInputStream`

Constructors

[EOFException\(\)](#)

[EOFException\(String\)](#)

EOFException

Applies to

Class `java.io.EOFException`

Declaration

```
public EOFException()  
public EOFException(String s)
```

Description

public EOFException()

Constructs an EOFException with no detail message. A detail message is a String that describes this particular exception.

public EOFException(String s)

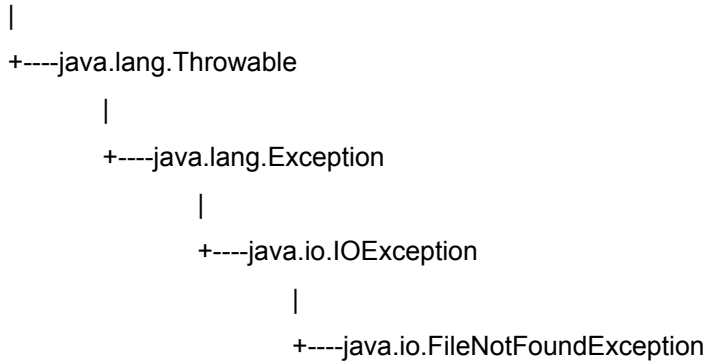
Constructs an EOFException with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class java.io.FileNotFoundException

java.lang.Object



Declaration

```
public class FileNotFoundException
    extends IOException
```

Description

Signals that a file was not found.

Constructors

[FileNotFoundException\(\)](#)

[FileNotFoundException\(String\)](#)

FileNotFoundException

Applies to

Class `java.io.FileNotFoundException`

Declaration

```
public FileNotFoundException()  
public FileNotFoundException(String s)
```

Description

```
public FileNotFoundException()
```

Constructs a `FileNotFoundException` with no detail message. A detail message is a `String` that describes this particular exception.

```
public FileNotFoundException(String s)
```

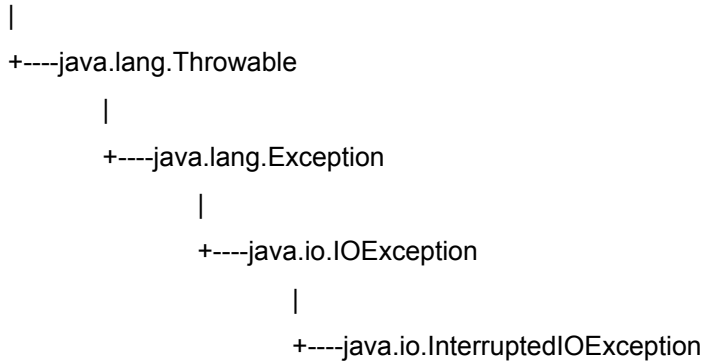
Constructs a `FileNotFoundException` with the specified detail message. A detail message is a `String` that describes this particular exception.

Parameters:

s - the detail message

Class java.io.InterruptedIOException

java.lang.Object



Declaration

```
public class InterruptedIOException
    extends IOException
```

Description

Signals that an I/O operation has been interrupted.

See Also:

`InputStream`, `OutputStream`

Variables

`bytesTransferred`

Constructors

`InterruptedIOException()`

`InterruptedIOException(String)`

bytesTransferred

Applies to

Class `java.io.InterruptedIOException`

Declaration

```
public int bytesTransferred
```

Description

Reports how many bytes had been transferred as part of the IO operation before it was interrupted.

InterruptedException

Applies to

Class `java.io.IOException`

Declaration

```
public IOException()  
public IOException(String s)
```

Description

```
public IOException()
```

Constructs an IOException with no detail message. A detail message is a String that describes this particular exception.

```
public IOException(String s)
```

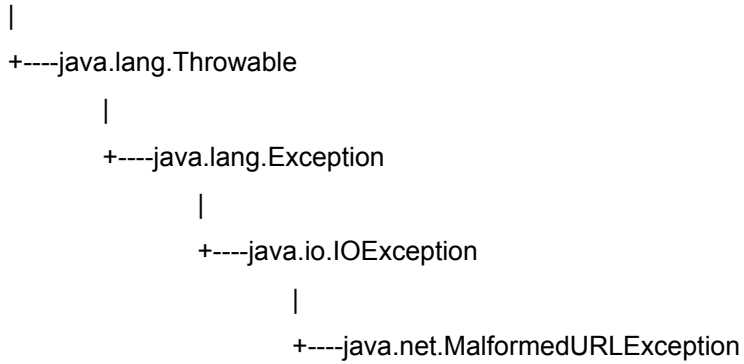
Constructs an IOException with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class `java.net.MalformedURLException`

`java.lang.Object`



Declaration

```
public class MalformedURLException
    extends IOException
```

Description

Signals that a malformed URL has occurred.

Constructors

[MalformedURLException\(\)](#)

[MalformedURLException\(String\)](#)

MalformedURLException

Applies to

Class `java.net.MalformedURLException`

Declaration

```
public MalformedURLException()  
public MalformedURLException(String msg)
```

Description

```
public MalformedURLException()
```

Constructs a `MalformedURLException` with no detail message. A detail message is a `String` that describes this particular exception.

```
public MalformedURLException(String msg)
```

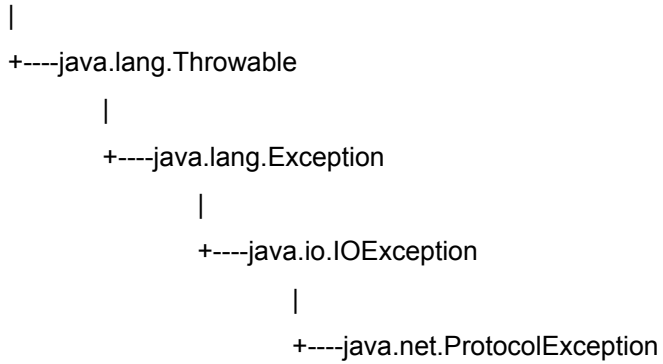
Constructs a `MalformedURLException` with the specified detail message. A detail message is a `String` that describes this particular exception.

Parameters:

`msg` - the detail message

Class java.net.ProtocolException

java.lang.Object



Declaration

```
public class ProtocolException
    extends IOException
```

Description

Signals when connect gets an EPROTO. This exception is specifically caught in class Socket.

Constructors

[ProtocolException\(String\)](#)

[ProtocolException\(\)](#)

ProtocolException

Applies to

Class `java.net.ProtocolException`

Declaration

```
public ProtocolException(String host)
public ProtocolException()
```

Description

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:

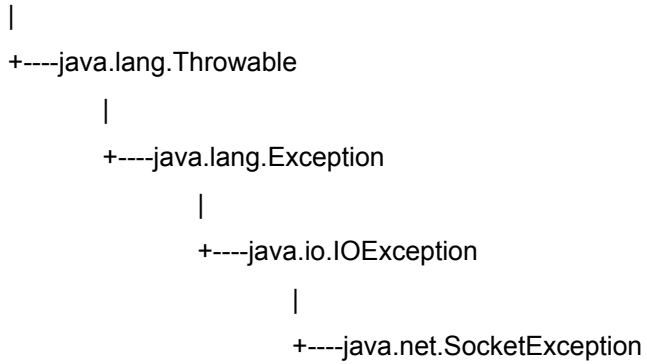
host - the detail message

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.SocketException

java.lang.Object



Declaration

```
public class SocketException
    extends IOException
```

Description

Signals that an error occurred while attempting to use a socket.

Constructors

```
SocketException(String)
SocketException()
```

SocketException

Applies to

Class `java.net.SocketException`

Declaration

```
public SocketException(String msg)
public SocketException()
```

Description

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:

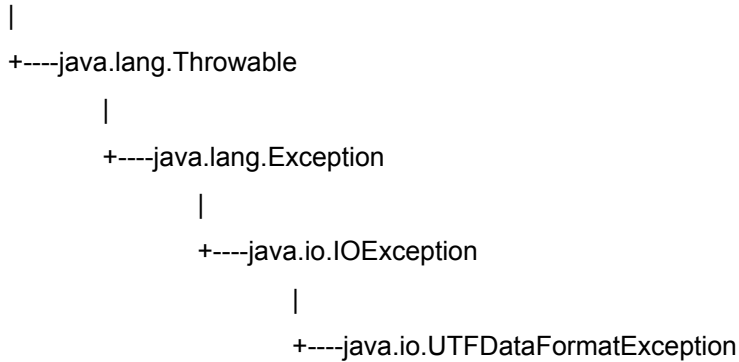
msg - the detail message

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.io.UTFDataFormatException`

`java.lang.Object`



Declaration

```
public class UTFDataFormatException
    extends IOException
```

Description

Signals that a malformed UTF-8 string has been read in a `DataInput` stream.

See Also:

`IOException`, `DataInput`

Constructors

[UTFDataFormatException\(\)](#)

[UTFDataFormatException\(String\)](#)

UTFDataFormatException

Applies to

Class `java.io.UTFDataFormatException`

Declaration

```
public UTFDataFormatException()  
public UTFDataFormatException(String s)
```

Description

public UTFDataFormatException()

Constructs an UTFDataFormatException with no detail message. A detail message is a String that describes this particular exception.

public UTFDataFormatException(String s)

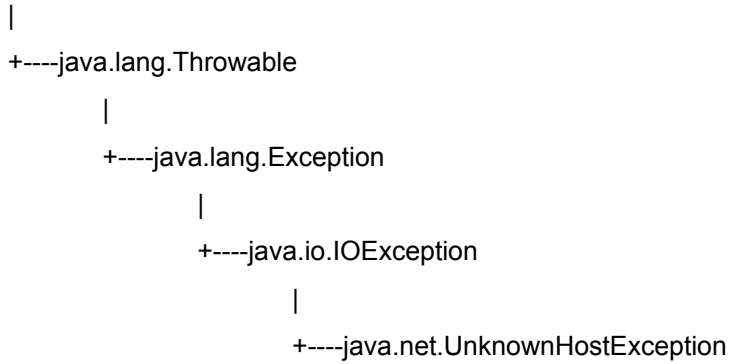
Constructs an UTFDataFormatException with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class `java.net.UnknownHostException`

`java.lang.Object`



Declaration

```
public class UnknownHostException
    extends IOException
```

Description

Signals that the address of the server specified by a network client could not be resolved.

Constructors

[UnknownHostException\(String\)](#)

[UnknownHostException\(\)](#)

UnknownHostException

Applies to

Class `java.net.UnknownHostException`

Declaration

```
public UnknownHostException(String host)
public UnknownHostException()
```

Description

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:

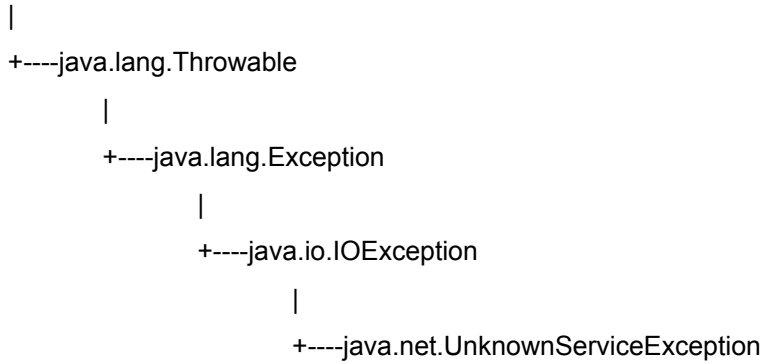
host - the detail message

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`

`java.lang.Object`



Declaration

```
public class UnknownServiceException
    extends IOException
```

Description

Signals that an unknown service exception has occurred.

Constructors

[UnknownServiceException\(\)](#)

[UnknownServiceException\(String\)](#)

UnknownServiceException

Applies to

Class `java.net.UnknownServiceException`

Declaration

```
public UnknownServiceException()  
public UnknownServiceException(String msg)
```

Description

```
public UnknownServiceException()
```

Constructs a new `UnknownServiceException` with no detail message. A detail message is a `String` that gives a specific description of this error.

```
public UnknownServiceException(String msg)
```

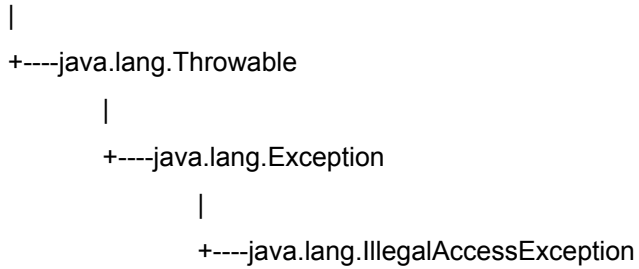
Constructs a new `UnknownServiceException` with the specified detail message. A detail message is a `String` that gives a specific description of this error.

Parameters:

`msg` - the detail message

Class java.lang.IllegalAccessException

java.lang.Object



Declaration

```
public class IllegalAccessException
    extends Exception
```

Description

Signals that a particular method could not be found.

Constructors

[IllegalAccessException\(\)](#)

[IllegalAccessException\(String\)](#)

IllegalAccessException

Applies to

Class `java.lang.IllegalAccessException`

Declaration

```
public IllegalAccessException()  
public IllegalAccessException(String s)
```

Description

public IllegalAccessException()

Constructs a `IllegalAccessException` without a detail message. A detail message is a `String` that describes this particular exception.

public IllegalAccessException(String s)

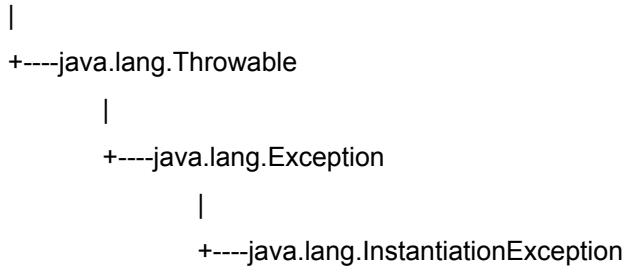
Constructs a `IllegalAccessException` with a detail message. A detail message is a `String` that describes this particular exception.

Parameters:

s - the detail message

Class java.langInstantiationException

java.lang.Object



Declaration

```
public class InstantiationException  
extends Exception
```

Description

Signals that an attempt has been made to instantiate an abstract class or an interface.

Constructors

[InstantiationException\(\)](#)

[InstantiationException\(String\)](#)

InstantiationException

Applies to

Class `java.langInstantiationException`

Declaration

```
public InstantiationException()  
public InstantiationException(String s)
```

Description

Constructs an `InstantiationException` with no detail message. A detail message is a `String` that describes this particular exception.

`public InstantiationException(String s)`

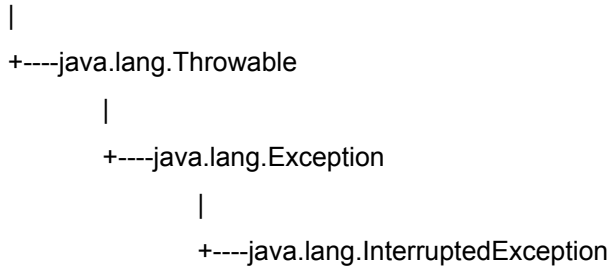
Constructs an `InstantiationException` with the specified detail message. A detail message is a `String` that describes this particular exception.

Parameters:

s - the `String` containing a detail message

Class java.lang.InterruptedIOException

java.lang.Object



Declaration

```
public class InterruptedIOException
    extends Exception
```

Description

An exception indicated that some thread has interrupted this thread.

See Also:

interrupt, interrupted

Constructors

[InterruptedIOException\(\)](#)

[InterruptedIOException\(String\)](#)

InterruptedException

Applies to

Class `java.lang InterruptedException`

Declaration

```
public InterruptedException()  
public InterruptedException(String s)
```

Description

public InterruptedException()

Constructs an InterruptedException with no detail message. A detail message is a String that describes this particular exception.

public InterruptedException(String s)

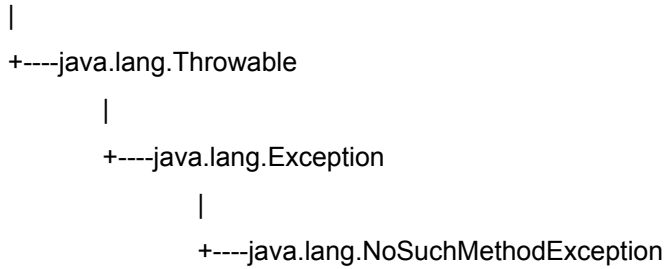
Constructs an InterruptedException with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class java.lang.NoSuchMethodException

java.lang.Object



Declaration

```
public class NoSuchMethodException
    extends Exception
```

Description

Signals that a particular method could not be found.

Constructors

[NoSuchMethodException\(\)](#)

[NoSuchMethodException\(String\)](#)

NoSuchMethodException

Applies to

Class `java.lang.NoSuchMethodException`

Declaration

```
public NoSuchMethodException()  
public NoSuchMethodException(String s)
```

Description

public NoSuchMethodException()

Constructs a `NoSuchMethodException` without a detail message. A detail message is a `String` that describes this particular exception.

public NoSuchMethodException(String s)

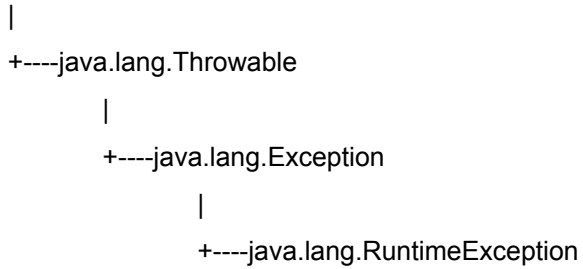
Constructs a `NoSuchMethodException` with a detail message. A detail message is a `String` that describes this particular exception.

Parameters:

s - the detail message

Class java.lang.RuntimeException

java.lang.Object



Declaration

```
public class RuntimeException
    extends Exception
```

Description

An exception that can reasonably occur during the execution of a Java program by the Virtual machine.

Constructors

[RuntimeException\(\)](#)

[RuntimeException\(String\)](#)

RuntimeException

Applies to

Class `java.lang.RuntimeException`

Declaration

```
public RuntimeException()  
public RuntimeException(String s)
```

Description

public RuntimeException()

Constructs a RuntimeException with no detail message. A detail message is a String that describes this particular exception.

public RuntimeException(String s)

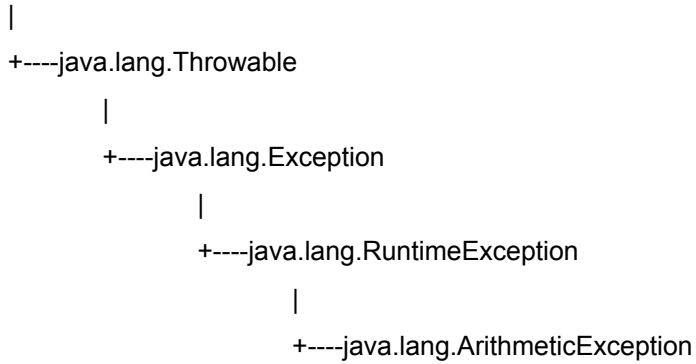
Constructs a RuntimeException with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class java.lang.ArithmeticException

java.lang.Object



Declaration

```
public class ArithmeticException
extends RuntimeException
```

Description

Signals that an exceptional arithmetic condition has occurred. For example, dividing by zero would invoke this class.

Constructors

[ArithmeticException\(\)](#)

[ArithmeticException\(String\)](#)

ArithmeticException

Applies to

Class `java.lang.ArithmeticException`

Declaration

```
public ArithmeticException()  
public ArithmeticException(String s)
```

Description

public ArithmeticException()

Constructs an `ArithmeticException` with no detail message. A detail message is a `String` that describes this particular exception.

public ArithmeticException(String s)

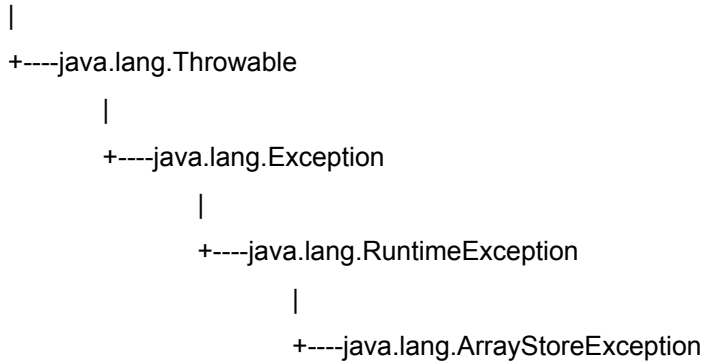
Constructs an `ArithmeticException` with the specified detail message. A detail message is a `String` that describes this particular exception.

Parameters:

s - the `String` that contains a detailed message

Class java.lang.ArrayStoreException

java.lang.Object



Declaration

```
public class ArrayStoreException
    extends RuntimeException
```

Description

An attempt has been made to store the wrong type of Object to an array.

Constructors

[ArrayStoreException\(\)](#)

[ArrayStoreException\(String\)](#)

ArrayStoreException

Applies to

Class `java.lang.ArrayStoreException`

Declaration

```
public ArrayStoreException()  
public ArrayStoreException(String s)
```

Description

public ArrayStoreException()

Constructs a `ArrayStoreException` with no detail message. A detail message is a `String` that describes this particular exception.

public ArrayStoreException(String s)

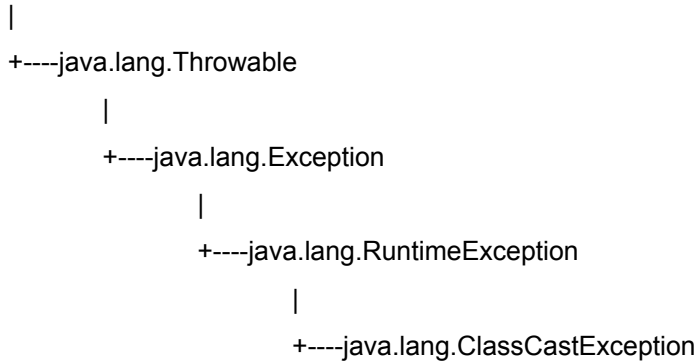
Constructs a `ArrayStoreException` with the specified detail message. A detail message is a `String` that describes this particular exception.

Parameters:

s - the `String` containing a detail message

Class java.lang.ClassCastException

java.lang.Object



Declaration

```
public class ClassCastException
    extends RuntimeException
```

Description

Signals that an invalid cast has occurred.

Constructors

[ClassCastException\(\)](#)

[ClassCastException\(String\)](#)

ClassCastException

Applies to

Class `java.lang.ClassCastException`

Declaration

```
public ClassCastException()  
public ClassCastException(String s)
```

Description

public ClassCastException()

Constructs a `ClassCastException` with no detail message. A detail message is a `String` that describes this particular exception.

public ClassCastException(String s)

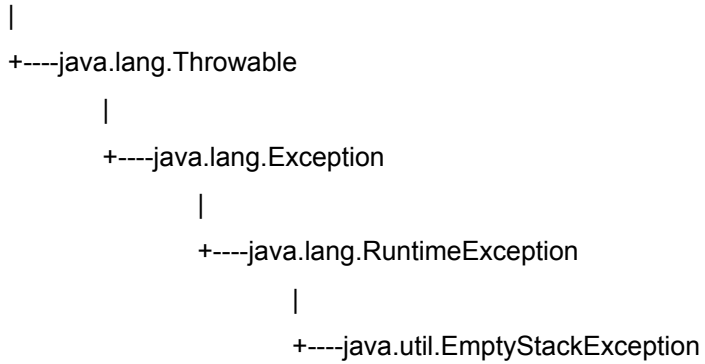
Constructs a `ClassCastException` with the specified detail message. A detail message is a `String` that describes this particular exception.

Parameters:

`s` - the `String` containing a detail message

Class java.util.EmptyStackException

java.lang.Object



Declaration

```
public class EmptyStackException
    extends RuntimeException
```

Description

Signals that the stack is empty.

See Also:

Stack

Constructors

[EmptyStackException\(\)](#)

EmptyStackException

Applies to

Class `java.util.EmptyStackException`

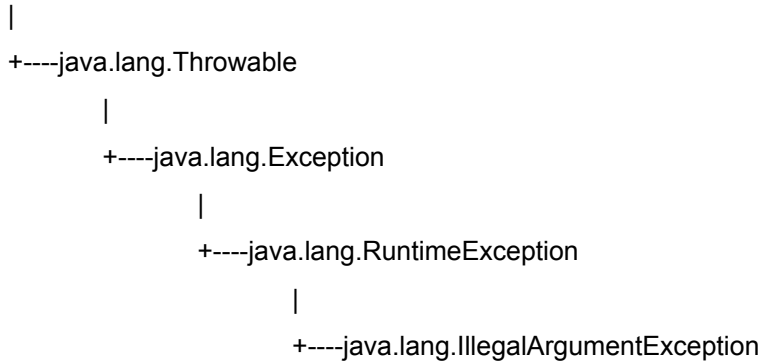
```
public EmptyStackException()
```

Description

Constructs a new `EmptyStackException` with no detail message. A detail message is a `String` that describes the exception.

Class `java.lang.IllegalArgumentException`

`java.lang.Object`



Declaration

```
public class IllegalArgumentException  
extends RuntimeException
```

Description

Signals that an illegal argument exception has occurred.

See Also:

`setPriority`

Constructors

[IllegalArgumentException\(\)](#)

[IllegalArgumentException\(String\)](#)

IllegalArgumentException

Applies to

Class `java.lang.IllegalArgumentException`

Declaration

```
public IllegalArgumentException()  
public IllegalArgumentException(String s)
```

Description

public IllegalArgumentException()

Constructs an IllegalArgumentException with no detail message. A detail message is a String that describes this particular exception.

public IllegalArgumentException(String s)

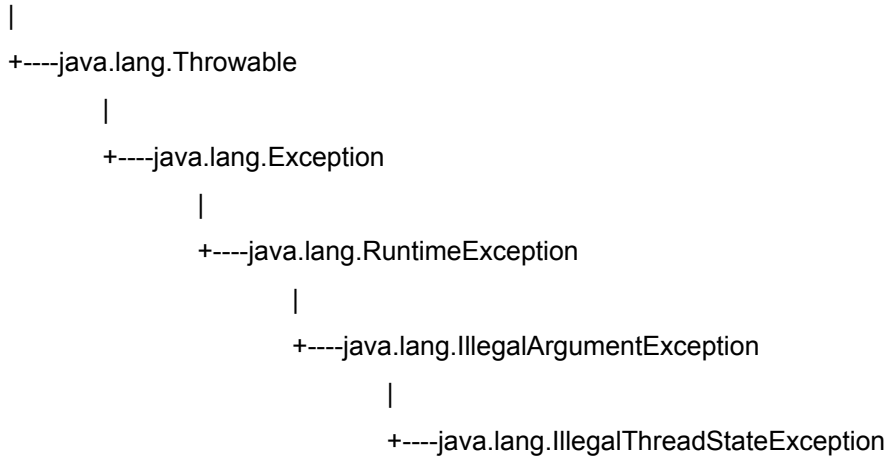
Constructs an IllegalArgumentException with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class java.lang.IllegalThreadStateException

java.lang.Object



Declaration

```
public class IllegalThreadStateException
    extends IllegalArgumentException
```

Description

Exception indicating that a thread is not in the proper state for the requested operation.

See Also:

suspend, resume

Constructors

[IllegalThreadStateException\(\)](#)

[IllegalThreadStateException\(String\)](#)

IllegalThreadStateException

Applies to

Class `java.lang.IllegalThreadStateException`

Declaration

```
public IllegalThreadStateException()  
public IllegalThreadStateException(String s)
```

Description

public IllegalThreadStateException()

Constructs an `IllegalThreadStateException` with no detail message. A detail message is a `String` that describes this particular exception.

public IllegalThreadStateException(String s)

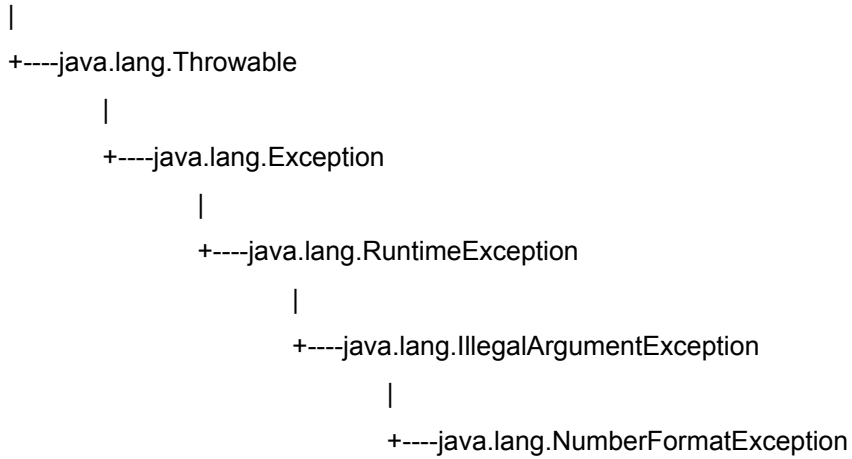
Constructs an `IllegalThreadStateException` with the specified detail message. A detail message is a `String` that describes this particular exception.

Parameters:

s - the detail message

Class java.lang.NumberFormatException

java.lang.Object



Declaration

```
public class NumberFormatException
    extends IllegalArgumentException
```

Description

Signals that an invalid number format has occurred.

See Also:

`toString`

Constructors

[NumberFormatException\(\)](#)

[NumberFormatException\(String\)](#)

NumberFormatException

Applies to

Class `java.lang.NumberFormatException`

Declaration

```
public NumberFormatException()  
public NumberFormatException(String s)
```

Description

public NumberFormatException()

Constructs a `NumberFormatException` with no detail message. A detail message is a `String` that describes this particular exception.

public NumberFormatException(String s)

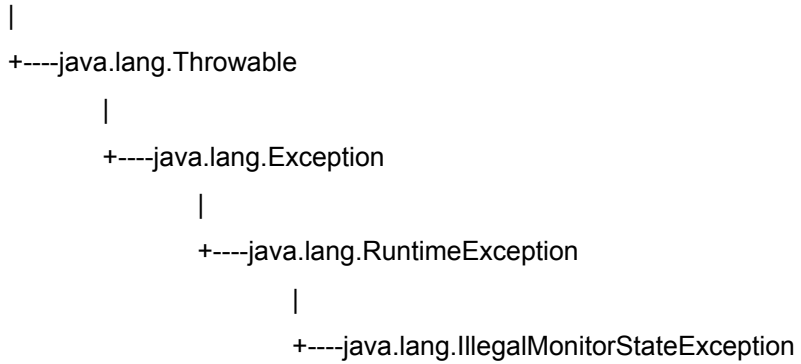
Constructs a `NumberFormatException` with the specified detail message. A detail message is a `String` that describes this particular exception.

Parameters:

s - the detail message

Class java.lang.IllegalMonitorStateException

java.lang.Object



Declaration

```
public class IllegalMonitorStateException
extends RuntimeException
```

Description

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.

Constructors

[IllegalMonitorStateException\(\)](#)

[IllegalMonitorStateException\(String\)](#)

IllegalMonitorStateException

Applies to

Class `java.lang.IllegalMonitorStateException`

Declaration

```
public IllegalMonitorStateException()  
public IllegalMonitorStateException(String s)
```

Description

public IllegalMonitorStateException()

Constructs an `IllegalMonitorStateException` with no detail message. A detail message is a `String` that describes this particular exception.

public IllegalMonitorStateException(String s)

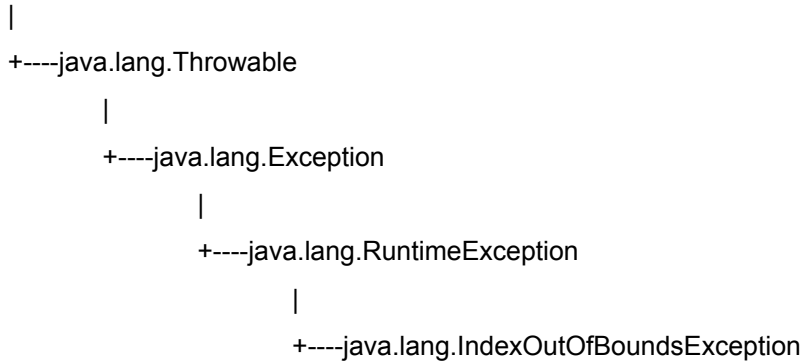
Constructs an `IllegalMonitorStateException` with the specified detail message. A detail message is a `String` that describes this particular exception.

Parameters:

s - the `String` that contains a detailed message

Class java.lang.IndexOutOfBoundsException

java.lang.Object



Declaration

```
public class IndexOutOfBoundsException
    extends RuntimeException
```

Description

Signals that an index of some sort is out of bounds.

Constructors

[IndexOutOfBoundsException\(\)](#)

[IndexOutOfBoundsException\(String\)](#)

IndexOutOfBoundsException

Applies to

Class `java.lang.IndexOutOfBoundsException`

Declaration

```
public IndexOutOfBoundsException()  
public IndexOutOfBoundsException(String s)
```

Description

public IndexOutOfBoundsException()

Constructs an `IndexOutOfBoundsException` with no detail message. A detail message is a `String` that describes this particular exception.

public IndexOutOfBoundsException(String s)

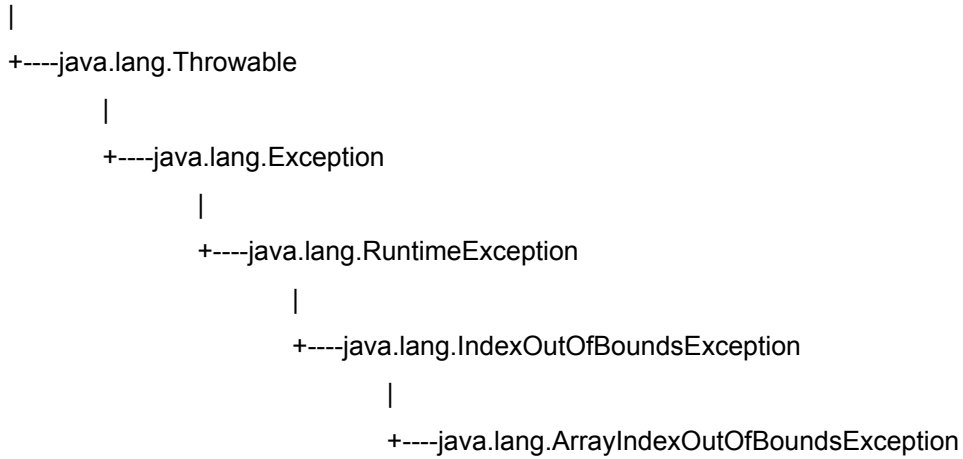
Constructs a `IndexOutOfBoundsException` with the specified detail message. A detail message is a `String` that describes this particular exception.

Parameters:

s - the detail message

Class java.lang.ArrayIndexOutOfBoundsException

java.lang.Object



Declaration

```
public class ArrayIndexOutOfBoundsException
    extends IndexOutOfBoundsException
```

Description

Signals that an invalid array index has been used.

Constructors

[ArrayIndexOutOfBoundsException\(\)](#)

[ArrayIndexOutOfBoundsException\(int\)](#)

[ArrayIndexOutOfBoundsException\(String\)](#)

ArrayIndexOutOfBoundsException

Applies to

Class `java.lang.ArrayIndexOutOfBoundsException`

Declaration

```
public ArrayIndexOutOfBoundsException()  
public ArrayIndexOutOfBoundsException(int index)  
public ArrayIndexOutOfBoundsException(String s)
```

Description

public ArrayIndexOutOfBoundsException()

Constructs an `ArrayIndexOutOfBoundsException` with no detail message. A detail message is a `String` that describes this particular exception.

public ArrayIndexOutOfBoundsException(int index)

Constructs a new `ArrayIndexOutOfBoundsException` class initialized to the specific index.

Parameters:

index - the index where the error occurred

public ArrayIndexOutOfBoundsException(String s)

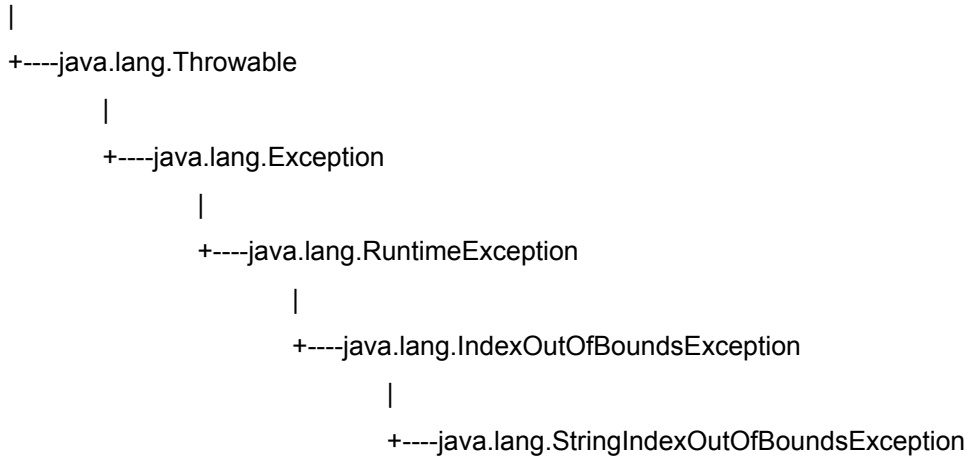
Constructs an `ArrayIndexOutOfBoundsException` class with the specified detail message. A detail message is a `String` that describes this particular exception.

Parameters:

s - the `String` containing a detail message

Class java.lang.StringIndexOutOfBoundsException

java.lang.Object



Declaration

```
public class StringIndexOutOfBoundsException
    extends IndexOutOfBoundsException
```

Description

Signals that a String index is out of range.

See Also:

`charAt`

Constructors

```
StringIndexOutOfBoundsException()
StringIndexOutOfBoundsException(String)
StringIndexOutOfBoundsException(int)
```

StringIndexOutOfBoundsException

Applies to

Class `java.lang.StringIndexOutOfBoundsException`

```
public StringIndexOutOfBoundsException()  
public StringIndexOutOfBoundsException(String s)  
public StringIndexOutOfBoundsException(int index)
```

Description

public StringIndexOutOfBoundsException()

Constructs a `StringIndexOutOfBoundsException` with no detail message. A detail message is a `String` that describes this particular exception.

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

public StringIndexOutOfBoundsException(int index)

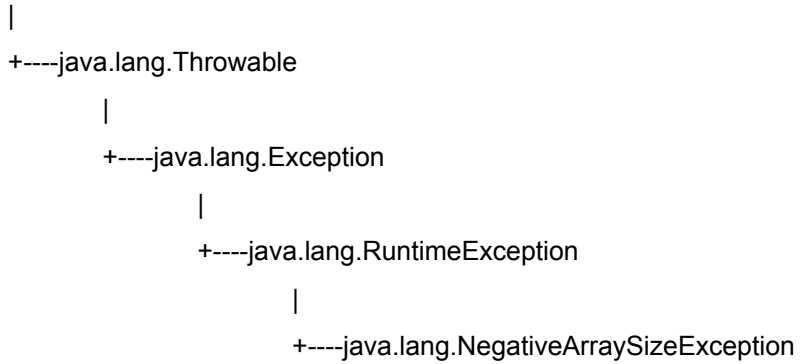
Constructs a `StringIndexOutOfBoundsException` initialized with the specified index.

Parameters:

index - the offending index

Class java.lang.NegativeArraySizeException

java.lang.Object



Declaration

```
public class NegativeArraySizeException
extends RuntimeException
```

Description

Signals that an attempt has been made to create an array with negative size.

Constructors

[NegativeArraySizeException\(\)](#)

[NegativeArraySizeException\(String\)](#)

NegativeArraySizeException

Applies to

Class `java.lang.NegativeArraySizeException`

Declaration

```
public NegativeArraySizeException()  
public NegativeArraySizeException(String s)
```

Description

public NegativeArraySizeException()

Constructs a `NegativeArraySizeException` with no detail message. A detail message is a `String` that describes this particular exception.

public NegativeArraySizeException(String s)

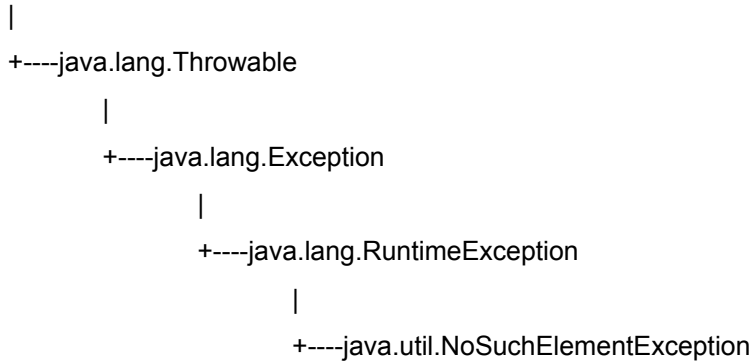
Constructs a `NegativeArraySizeException` with the specified detail message. A detail message is a `String` that describes this particular exception.

Parameters:

s - the detail message

Class `java.util.NoSuchElementException`

`java.lang.Object`



Declaration

```
public class NoSuchElementException  
extends RuntimeException
```

Description

Signals that an enumeration is empty.

See Also:

Enumeration

Constructors

[`NoSuchElementException\(\)`](#)

[`NoSuchElementException\(String\)`](#)

NoSuchElementException

Applies to

Class `java.util.NoSuchElementException`

Declaration

```
public NoSuchElementException()  
public NoSuchElementException(String s)
```

Description

public NoSuchElementException()

Constructs a NoSuchElementException with no detail message. A detail message is a String that describes this particular exception.

public NoSuchElementException(String s)

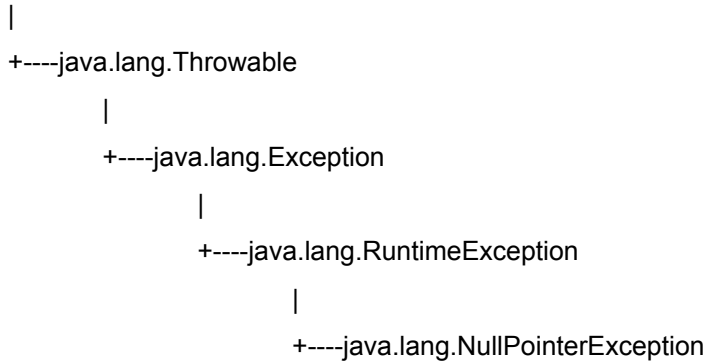
Constructs a NoSuchElementException with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s - the detail message

Class java.lang.NullPointerException

java.lang.Object



Declaration

```
public class NullPointerException
extends RuntimeException
```

Description

Signals the illegal use of a null pointer.

Constructors

[NullPointerException\(\)](#)

[NullPointerException\(String\)](#)

NullPointerException

Applies to

Class `java.lang.NullPointerException`

Declaration

```
public NullPointerException()  
public NullPointerException(String s)
```

Description

public NullPointerException()

Constructs a `NullPointerException` with no detail message. A detail message is a `String` that describes this particular exception.

public NullPointerException(String s)

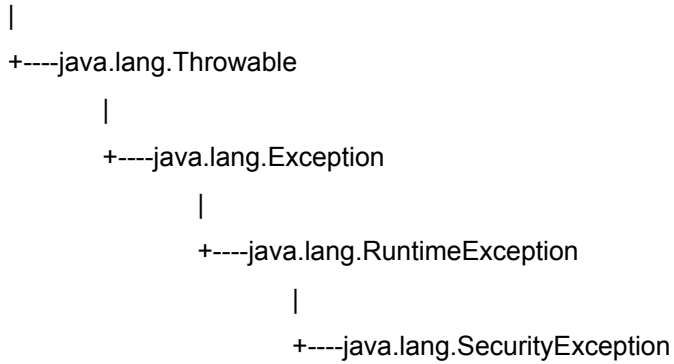
Constructs a `NullPointerException` with the specified detail message. A detail message is a `String` that describes this particular exception.

Parameters:

s - the detail message

Class java.lang.SecurityException

java.lang.Object



Declaration

```
public class SecurityException
    extends RuntimeException
```

Description

Signals that a security exception has occurred.

Constructors

[SecurityException\(\)](#)

[SecurityException\(String\)](#)

SecurityException

Applies to

Class `java.lang.SecurityException`

Declaration

```
public SecurityException()  
public SecurityException(String s)
```

Description

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.awt.Toolkit

java.lang.Object

|

+----java.awt.Toolkit

Declaration

public class Toolkit

extends Object

Description

An AWT toolkit. It is used to bind the abstract AWT classes to a particular native toolkit implementation.

Constructors

[Toolkit\(\)](#)

Methods

[checkImage\(Image, int, int, ImageObserver\)](#)

[createButton\(Button\)](#)

[createCanvas\(Canvas\)](#)

[createCheckbox\(Checkbox\)](#)

[createCheckboxMenuItem\(CheckboxMenuItem\)](#)

[createChoice\(Choice\)](#)

[createDialog\(Dialog\)](#)

[createFileDialog\(FileDialog\)](#)

[createFrame\(Frame\)](#)

[createImage\(ImageProducer\)](#)

[createLabel\(Label\)](#)

[createList\(List\)](#)

[createMenu\(Menu\)](#)

[createMenuBar\(MenuBar\)](#)

[createMenuItem\(MenuItem\)](#)

createPanel(Panel)
createScrollbar(Scrollbar)
createTextArea(TextArea)
createTextField(TextField)
createWindow(Window)
getColorModel()
getDefaultToolkit()
getFontList()
getFontMetrics(Font)
getImage(String)
getImage(URL)
getScreenResolution()
getScreenSize()
prepareImage(Image, int, int, ImageObserver)
sync()

Toolkit

Applies to

Class java.awt.Toolkit

Declaration

```
public Toolkit()
```

createButton

Applies to

Class `java.awt.Toolkit`

Declaration

```
protected abstract ButtonPeer createButton(Button target)
```

Description

Uses the specified Peer interface to create a new Button.

Parameters:

target - the Button to be created

createTextField

Applies to

Class java.awt.Toolkit

Declaration

```
protected abstract TextFieldPeer createTextField(TextField target)
```

Description

Uses the specified Peer interface to create a new TextField.

Parameters:

target - the TextField to be created

createLabel

Applies to

Class `java.awt.Toolkit`

Declaration

```
protected abstract LabelPeer createLabel(Label target)
```

Description

Uses the specified Peer interface to create a new Label.

Parameters:

target - the Label to be created

createList

Applies to

Class `java.awt.Toolkit`

Declaration

```
protected abstract ListPeer createList(List target)
```

Description

Uses the specified Peer interface to create a new List.

Parameters:

target - the List to be created

createCheckbox

Applies to

Class `java.awt.Toolkit`

Declaration

```
protected abstract CheckboxPeer createCheckbox(Checkbox target)
```

Description

Uses the specified Peer interface to create a new Checkbox.

Parameters:

target - the Checkbox to be created

createScrollbar

Applies to

Class java.awt.Toolkit

Declaration

```
protected abstract ScrollbarPeer createScrollbar(Scrollbar target)
```

Description

Uses the specified Peer interface to create a new Scrollbar.

Parameters:

target - the Scrollbar to be created

createTextArea

Applies to

Class `java.awt.Toolkit`

Declaration

```
protected abstract TextAreaPeer createTextArea(TextArea target)
```

Description

Uses the specified Peer interface to create a new TextArea.

Parameters:

target - the TextArea to be created

createChoice

Applies to

Class java.awt.Toolkit

Declaration

```
protected abstract ChoicePeer createChoice(Choice target)
```

Description

Uses the specified Peer interface to create a new Choice.

Parameters:

target - the Choice to be created

createFrame

Applies to

Class java.awt.Toolkit

Declaration

```
protected abstract FramePeer createFrame(Frame target)
```

Description

Uses the specified Peer interface to create a new Frame.

Parameters:

target - the Frame to be created

createCanvas

Applies to

Class `java.awt.Toolkit`

Declaration

```
protected abstract CanvasPeer createCanvas(Canvas target)
```

Description

Uses the specified Peer interface to create a new Canvas.

Parameters:

target - the Canvas to be created

createPanel

Applies to

Class java.awt.Toolkit

Declaration

```
protected abstract PanelPeer createPanel(Panel target)
```

Description

Uses the specified Peer interface to create a new Panel.

Parameters:

target - the Panel to be created

createWindow

Applies to

Class `java.awt.Toolkit`

Declaration

```
protected abstract WindowPeer createWindow(Window target)
```

Description

Uses the specified Peer interface to create a new Window.

Parameters:

target - the Window to be created

createDialog

Applies to

Class `java.awt.Toolkit`

Declaration

```
protected abstract DialogPeer createDialog(Dialog target)
```

Description

Uses the specified Peer interface to create a new Dialog.

Parameters:

target - the Dialog to be created

createMenuBar

Applies to

Class java.awt.Toolkit

Declaration

```
protected abstract MenuBarPeer createMenuBar(MenuBar target)
```

Description

Uses the specified Peer interface to create a new MenuBar.

Parameters:

target - the MenuBar to be created

createMenu

Applies to

Class java.awt.Toolkit

Declaration

```
protected abstract MenuPeer createMenu(Menu target)
```

Description

Uses the specified Peer interface to create a new Menu.

Parameters:

target - the Menu to be created

createMenuItem

Applies to

Class java.awt.Toolkit

Declaration

```
protected abstract MenuItemPeer createMenuItem(MenuItem target)
```

Description

Uses the specified Peer interface to create a new MenuItem.

Parameters:

target - the MenuItem to be created

createFileDialog

Applies to

Class `java.awt.Toolkit`

Declaration

```
protected abstract FileDialogPeer createFileDialog(FileDialog target)
```

Description

Uses the specified Peer interface to create a new FileDialog.

Parameters:

target - the FileDialog to be created

createCheckboxMenuItem

Applies to

Class java.awt.Toolkit

Declaration

```
protected abstract CheckboxMenuItemPeer createCheckboxMenuItem(  
CheckboxMenuItem target)
```

Description

Uses the specified Peer interface to create a new CheckboxMenuItem.

Parameters:

target - the CheckboxMenuItem to be created

getScreenSize

Applies to

Class java.awt.Toolkit

Declaration

```
public abstract Dimension getScreenSize()
```

Description

Gets the size of the screen.

getScreenResolution

Applies to

Class `java.awt.Toolkit`

Declaration

```
public abstract int getScreenResolution()
```

Description

Returns the screen resolution in dots-per-inch.

getColorModel

Applies to

Class `java.awt.Toolkit`

Declaration

```
public abstract ColorModel getColorModel()
```

Description

Returns the ColorModel of the screen.

getFontList

Applies to

Class `java.awt.Toolkit`

Declaration

```
public abstract String[] getFontList()
```

Description

Returns the names of the available fonts.

getFontMetrics

Applies to

Class `java.awt.Toolkit`

Declaration

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

Description

Returns the screen metrics of the font.

sync

Applies to

Class `java.awt.Toolkit`

Declaration

```
public abstract void sync()
```

Description

Syncs the graphics state; useful when doing animation.

getDefaultToolkit

Applies to

Class `java.awt.Toolkit`

Declaration

```
public static synchronized Toolkit getDefaultToolkit()
```

Description

Returns the default toolkit. This is controlled by the "awt.toolkit" property.

Throws: `AWTError`

Toolkit not found or could not be instantiated.

getImage

Applies to

Class java.awt.Toolkit

Declaration

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

Description

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

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

Applies to

Class `java.awt.Toolkit`

Declaration

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

Description

Prepares an image for rendering on the default screen at the specified width and height.

checkImage

Applies to

Class java.awt.Toolkit

Declaration

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

Description

Returns the status of the construction of the indicated method at the indicated width and height for the default screen.

createImage

Applies to

Class java.awt.Toolkit

Declaration

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

Description

Creates an image with the specified image producer.

Parameters:

producer - the image producer to be used

Class java.net.URL

java.lang.Object

|

+----java.net.URL

Declaration

public final class URL

extends Object

Description

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.

Constructors

[URL\(String, String, int, String\)](#)

[URL\(String, String, String\)](#)

[URL\(String\)](#)

[URL\(URL, String\)](#)

Methods

[equals\(Object\)](#)

[getContent\(\)](#)

[getFile\(\)](#)

[getHost\(\)](#)

[getPort\(\)](#)

[getProtocol\(\)](#)

[getRef\(\)](#)

[hashCode\(\)](#)

[openConnection\(\)](#)

[openStream\(\)](#)

[sameFile\(URL\)](#)

set(String, String, int, String, String)

setURLStreamHandlerFactory(URLStreamHandlerFactory)

toExternalForm()

toString()

URL

Applies to

Class java.net.URL

Declaration

```
public URL(String protocol,  
            String host,  
            int port,  
            String file) throws MalformedURLException  
public URL(String protocol,  
            String host,  
            String file) throws MalformedURLException  
public URL(String spec) throws MalformedURLException  
public URL(URL context,  
            String spec) throws MalformedURLException
```

Description

```
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.

**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.

public URL(String spec) throws MalformedURLException

Creates a URL from the unparsed absolute URL.

Parameters:

spec - the URL String to parse

**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 is parsed 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

Applies to

Class java.net.URL

Declaration

```
protected void set(String protocol,  
                   String host,  
                   int port,  
                   String file,  
                   String ref)
```

Description

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 connect to

port - the protocol port to connect to

file - the specified file name on that host

ref - the reference

getPort

Applies to

Class java.net.URL

Declaration

```
public int getPort()
```

Description

Gets the port number. Returns -1 if the port is not set.

getProtocol

Applies to

Class java.net.URL

Declaration

```
public String getProtocol()
```

Description

Gets the protocol name.

getHost

Applies to

Class java.net.URL

Declaration

```
public String getHost()
```

Description

Gets the host name.

getFile

Applies to

Class java.net.URL

Declaration

```
public String getFile()
```

Description

Gets the file name.

getRef

Applies to

Class java.net.URL

Declaration

```
public String getRef()
```

Description

Gets the ref.

equals

Applies to

Class java.net.URL

Declaration

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

Description

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

hashCode

Applies to

Class java.net.URL

Declaration

```
public int hashCode()
```

Description

Creates an integer suitable for hash table indexing.

Overrides:

hashCode in class Object

sameFile

Applies to

Class java.net.URL

Declaration

```
public boolean sameFile(URL other)
```

Description

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

Applies to

Class java.net.URL

Declaration

```
public String toString()
```

Description

Converts to a human-readable form.

Returns:

the textual representation.

Overrides:

toString in class Object

toExternalForm

Applies to

Class java.net.URL

Declaration

```
public String toExternalForm()
```

Description

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

Applies to

Class java.net.URL

Declaration

```
public URLConnection openConnection() throws IOException
```

Description

Creates (if not already in existence) 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

Applies to

Class java.net.URL

Declaration

```
public final InputStream openStream() throws IOException
```

Description

Opens an input stream.

Throws: IOException

If an I/O exception has occurred.

getContent

Applies to

Class java.net.URL

Declaration

```
public final Object getContent() throws IOException
```

Description

Gets the contents from this opened connection.

Throws: IOException

If an I/O exception has occurred.

setURLStreamHandlerFactory

Applies to

Class java.net.URL

Declaration

```
public static synchronized void setURLStreamHandlerFactory(  
URLStreamHandlerFactory fac)
```

Description

Sets the URLStreamHandler factory.

Parameters:

fac - the desired factory

Throws: Error

If the factory has already been defined.

Class java.net.URLConnection

java.lang.Object

|

+----java.net.URLConnection

Declaration

public class URLConnection

extends Object

Description

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.

Variables

allowUserInteraction

connected

doInput

doOutput

ifModifiedSince

url

useCaches

Constructors

URLConnection(URL)

Methods

connect()

getAllowUserInteraction()

getContent()

getContentEncoding()

getContentLength()

getContentType()
getDate()
getDefaultAllowUserInteraction()
getDefaultRequestProperty (String)
getDefaultUseCaches()
getDoInput()
getDoOutput()
getExpiration()
getHeaderField(String)
getHeaderField(int)
getHeaderFieldDate(String, long)
getHeaderFieldInt(String, int)
getHeaderFieldKey(int)
getIfModifiedSince()
getInputStream()
getLastModified()
getOutputStream()
getRequestProperty(String)
getURL()
getUseCaches()
guessContentTypeFromName (String)
guessContentTypeFromStream(InputStream)
setAllowUserInteraction(boolean)
setContentHandlerFactory(ContentHandlerFactory)
setDefaultAllowUserInteraction(boolean)
setDefaultRequestProperty(String, String)
setDefaultUseCaches(boolean)
setDoInput(boolean)
setDoOutput(boolean)
setIfModifiedSince(long)
setRequestProperty(String, String)
setUseCaches(boolean)
toString()

url

Applies to

Class java.net.URLConnection

Declaration

protected URL url

doInput

Applies to

Declaration

protected boolean doInput

doOutput

Applies to

Class `java.net.URLConnection`

Declaration

```
protected boolean doOutput
```

allowUserInteraction

Applies to

Class `java.net.URLConnection`

Declaration

```
protected boolean allowUserInteraction
```

useCaches

Applies to

Class java.net.URLConnection

Declaration

protected boolean useCaches

ifModifiedSince

Applies to

Class java.net.URLConnection

Declaration

protected long ifModifiedSince

connected

Applies to

Class java.net.URLConnection

Declaration

protected boolean connected

URLConnection

Applies to

Class java.net.URLConnection

Declaration

```
protected URLConnection(URL url)
```

Description

Constructs a URL connection to the specified URL.

Parameters:

url - the specified URL

connect

Applies to

Class `java.net.URLConnection`

Declaration

```
public abstract void connect() throws IOException
```

Description

`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 `getLength`, will implicitly perform the connection if necessary. Connecting when already connected does nothing.

getURL

Applies to

Class java.net.URLConnection

Declaration

```
public URL getURL()
```

Description

Gets the URL for this connection.

getLength

Applies to

Class java.net.URLConnection

Declaration

```
public int getLength()
```

Description

Gets the content length. Returns -1 if not known.

getContentType

Applies to

Class java.net.URLConnection

Declaration

```
public String getContentType()
```

Description

Gets the content type. Returns null if not known.

getContentEncoding

Applies to

Class java.net.URLConnection

Declaration

```
public String getContentEncoding()
```

Description

Gets the content encoding. Returns null if not known.

getExpiration

Applies to

Class java.net.URLConnection

Declaration

```
public long getExpiration()
```

Description

Gets the expiration date of the object. Returns 0 if not known.

getDate

Applies to

Class java.net.URLConnection

Declaration

```
public long getDate()
```

Description

Gets the sending date of the object. Returns 0 if not known.

getLastModified

Applies to

Class java.net.URLConnection

Declaration

```
public long getLastModified()
```

Description

Gets the last modified date of the object. Returns 0 if not known.

getHeaderField

Applies to

Class java.net.URLConnection

Declaration

```
public String getHeaderField(String name)  
public String getHeaderField(int n)
```

Description

public String getHeaderField(String name)

Gets a header field by name. Returns null if not known.

Parameters:

name - the name of the header field

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.

getHeaderFieldInt

Applies to

Class java.net.URLConnection

Declaration

```
public int getHeaderFieldInt(String name,  
                             int Default)
```

Description

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.

getHeaderFieldDate

Applies to

Class java.net.URLConnection

Declaration

```
public long getHeaderFieldDate(String name,  
                               long Default)
```

Description

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

Applies to

Class `java.net.URLConnection`

Declaration

```
public String getHeaderFieldKey(int n)
```

Description

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.

getContent

Applies to

Class java.net.URLConnection

Declaration

```
public Object getContent() throws IOException
```

Description

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

Applies to

Class `java.net.URLConnection`

Declaration

```
public InputStream getInputStream() throws IOException
```

Description

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

Applies to

Class java.net.URLConnection

Declaration

```
public OutputStream getOutputStream() throws IOException
```

Description

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

Applies to

Class java.net.URLConnection

Declaration

```
public String toString()
```

Description

Returns the String representation of the URL connection.

Overrides:

toString in class Object

setDoInput

Applies to

Class `java.net.URLConnection`

Declaration

```
public void setDoInput(boolean doinput)
```

Description

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

Applies to

Class java.net.URLConnection

Declaration

```
public boolean getDoInput()
```


setDoOutput

Applies to

Class `java.net.URLConnection`

Declaration

```
public void setDoOutput(boolean dooutput)
```

Description

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

Applies to

Class java.net.URLConnection

Declaration

```
public boolean getDoOutput()
```

setAllowUserInteraction

Applies to

Class `java.net.URLConnection`

Declaration

```
public void setAllowUserInteraction(boolean allowuserinteraction)
```

Description

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

Applies to

Class `java.net.URLConnection`

Declaration

```
public boolean getAllowUserInteraction()
```

setDefaultAllowUserInteraction

Applies to

Class `java.net.URLConnection`

Declaration

```
public static void setDefaultAllowUserInteraction(boolean defaultallowuserinteraction)
```

Description

Sets/gets the default value of the `allowUserInteraction` flag. This default is "sticky", being a part of the static state of all `URLConnection`s. This flag applies to the next, and all following `URLConnection`s that are created.

getDefaultAllowUserInteraction

Applies to

Class `java.net.URLConnection`

Declaration

```
public static boolean getDefaultAllowUserInteraction()
```

setUseCaches

Applies to

Class java.net.URLConnection

Declaration

```
public void setUseCaches(boolean usecaches)
```

Description

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

Applies to

Class java.net.URLConnection

Declaration

```
public boolean getUseCaches()
```


setIfModifiedSince

Applies to

Class `java.net.URLConnection`

Declaration

```
public void setIfModifiedSince(long ifmodifiedsince)
```

Description

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

Applies to

Class `java.net.URLConnection`

Declaration

```
public long getIfModifiedSince()
```

getDefaultUseCaches

Applies to

Class `java.net.URLConnection`

Declaration

```
public boolean getDefaultUseCaches()
```

Description

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

Applies to

Class java.net.URLConnection

Declaration

```
public void setDefaultUseCaches(boolean defaultusecaches)
```

setRequestProperty

Applies to

Class java.net.URLConnection

Declaration

```
public void setRequestProperty(String key,  
                               String value)
```

Description

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

Applies to

Class java.net.URLConnection

Declaration

```
public String getRequestProperty(String key)
```

setDefaultRequestProperty

Applies to

Class java.net.URLConnection

Declaration

```
public static void setDefaultRequestProperty(String key,  
                                             String value)
```

Description

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

Applies to

Class `java.net.URLConnection`

Declaration

```
public static String getDefaultRequestProperty(String key)
```


setContentHandlerFactory

Applies to

Class java.net.URLConnection

Declaration

```
public static synchronized void setContentHandlerFactory(  
ContentHandlerFactory fac)
```

Description

Sets the ContentHandler factory.

Parameters:

fac - the desired factory

Throws: Error

If the factory has already been defined.

guessContentTypeFromName

Applies to

Class java.net.URLConnection

Declaration

```
protected static String guessContentTypeFromName(String fname)
```

Description

A useful utility routine that tries to guess the content-type of an object based upon its extension.

guessContentTypeFromStream

Applies to

Class `java.net.URLConnection`

Declaration

protected static String guessContentTypeFromStream(InputStream is) throws IOException

Description

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

java.lang.Object

|

+----java.net.URLEncoder

Declaration

```
public class URLEncoder
```

```
extends Object
```

Description

Turns Strings of text into x-www-form-urlencoded format.

Methods

[encode\(String\)](#)

encode

Applies to

Declaration

```
public static String encode(String s)
```

Description

Translates String into x-www-form-urlencoded format.

Parameters:

s - String to be translated

Returns:

the translated String.

Class java.net.URLStreamHandler

java.lang.Object

|

+----java.net.URLStreamHandler

Declaration

```
public class URLStreamHandler
```

```
extends Object
```

Description

Abstract class for URL stream openers. Subclasses of this class know how to create streams for particular protocol types.

Constructors

[URLStreamHandler\(\)](#)

Methods

[openConnection\(URL\)](#)

[parseURL\(URL, String, int, int\)](#)

[setURL\(URL, String, String, int, String, String\)](#)

[toExternalForm\(URL\)](#)

URLStreamHandler

Applies to

Class `java.net.URLStreamHandler`

Declaration

```
public URLStreamHandler()
```

openConnection

Applies to

Class `java.net.URLStreamHandler`

Declaration

protected abstract `URLConnection` openConnection(`URL` u) throws `IOException`

Description

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

Applies to

Class java.net.URLStreamHandler

Declaration

```
protected void parseURL(URL u,  
                        String spec,  
                        int start,  
                        int limit)
```

Description

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

Applies to

Class `java.net.URLStreamHandler`

Declaration

```
protected String toExternalForm(URL u)
```

Description

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).

setURL

Applies to

Class `java.net.URLStreamHandler`

Declaration

```
protected void setURL(URL u,  
                        String protocol,  
                        String host,  
                        int port,  
                        String file,  
                        String ref)
```

Description

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`

Declaration

```
public interface URLStreamHandlerFactory  
extends Object
```

Description

This interface defines a factory for `URLStreamHandler` instances. It is used by the `URL` class to create `URLStreamHandlers` for various streams.

Methods

`createURLStreamHandler(String)`

createURLStreamHandler

Applies to

Interface `java.net.URLStreamHandlerFactory`

Declaration

```
public abstract URLStreamHandler createURLStreamHandler(String protocol)
```

Description

Creates a new `URLStreamHandler` instance with the specified protocol.

Parameters:

protocol - the protocol to use (ftp, http, nntp, etc.)

Class java.util.Vector

java.lang.Object

|

+----java.util.Vector

Declaration

public class Vector

extends Object

implements Cloneable

Description

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.

Variables

capacityIncrement

elementCount

elementData

Constructors

Vector(int, int)

Vector(int)

Vector()

Methods

addElement(Object)

capacity()
clone()
contains(Object)
copyInto(Object[])
elementAt(int)
elements()
ensureCapacity(int)
firstElement()
indexOf(Object)
indexOf(Object, int)
insertElementAt(Object, int)
isEmpty()
lastElement()
lastIndexOf(Object)
lastIndexOf(Object, int)
removeAllElements()
removeElement(Object)
removeElementAt(int)
setElementAt(Object, int)
setSize(int)
size()
toString()
trimToSize()

elementData

Applies to

Class `java.util.Vector`

Declaration

```
protected Object elementData[]
```

Description

The buffer where elements are stored.

elementCount

Applies to

Declaration

protected int elementCount

Description

The number of elements in the buffer.

capacityIncrement

Applies to

Class `java.util.Vector`

Declaration

```
protected int capacityIncrement
```

Description

The size of the increment. If it is 0 the size of the the buffer is doubled everytime it needs to grow.

Vector

Applies to

Class `java.util.Vector`

Declaration

```
public Vector(int initialCapacity, int capacityIncrement)
public Vector(int initialCapacity)
public Vector()
```

Description

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.

public Vector(int initialCapacity)

Constructs an empty vector with the specified storage capacity.

Parameters:

initialCapacity - the initial storage capacity of the vector

public Vector()

Constructs an empty vector.

copyInto

Applies to

Class `java.util.Vector`

Declaration

```
public final synchronized void copyInto(Object anArray[])
```

Description

Copies the elements of this vector into the specified array.

Parameters:

`anArray` - the array where elements get copied into

trimToSize

Applies to

Class `java.util.Vector`

Declaration

```
public final synchronized void trimToSize()
```

Description

Trims the vector's capacity down to size. Use this operation to minimize the storage of a vector. Subsequent insertions will cause reallocation.

ensureCapacity

Applies to

Class `java.util.Vector`

Declaration

```
public final synchronized void ensureCapacity(int minCapacity)
```

Description

Ensures that the vector has at least the specified capacity.

Parameters:

minCapacity - the desired minimum capacity

setSize

Applies to

Class `java.util.Vector`

Declaration

```
public final synchronized void setSize(int newSize)
```

Description

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

Applies to

Class `java.util.Vector`

Declaration

```
public final int capacity()
```

Description

Returns the current capacity of the vector.

size

Applies to

Class `java.util.Vector`

Declaration

```
public final int size()
```

Description

Returns the number of elements in the vector. Note that this is not the same as the vector's capacity.

isEmpty

Applies to

Class `java.util.Vector`

Declaration

```
public final boolean isEmpty()
```

Description

Returns true if the collection contains no values.

elements

Applies to

Class `java.util.Vector`

Declaration

```
public final synchronized Enumeration elements()
```

Description

Returns an enumeration of the elements. Use the Enumeration methods on the returned object to fetch the elements sequentially.

contains

Applies to

Class `java.util.Vector`

Declaration

```
public final boolean contains(Object elem)
```

Description

Returns true if the specified object is a value of the collection.

Parameters:

elem - the desired element

indexOf

Applies to

Class `java.util.Vector`

Declaration

```
public final int indexOf(Object elem)  
public final synchronized int indexOf(Object elem, int index)
```

Description

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.

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.

lastIndexOf

Applies to

Class `java.util.Vector`

Declaration

```
public final int lastIndexOf(Object elem)  
public final synchronized int lastIndexOf(Object elem, int index)
```

Description

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.

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

Applies to

Class `java.util.Vector`

Declaration

```
public final synchronized Object elementAt(int index)
```

Description

Returns the element at the specified index.

Parameters:

index - the index of the desired element

Throws: `ArrayIndexOutOfBoundsException`

If an invalid index was given.

firstElement

Applies to

Class `java.util.Vector`

Declaration

```
public final synchronized Object firstElement()
```

Description

Returns the first element of the sequence.

Throws: `NoSuchElementException`

If the sequence is empty.

lastElement

Applies to

Class `java.util.Vector`

Declaration

```
public final synchronized Object lastElement()
```

Description

Returns the last element of the sequence.

Throws: `NoSuchElementException`

If the sequence is empty.

setElementAt

Applies to

Class `java.util.Vector`

Declaration

```
public final synchronized void setElementAt(Object obj, int index)
```

Description

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

Applies to

Class `java.util.Vector`

Declaration

```
public final synchronized void removeElementAt(int index)
```

Description

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

Applies to

Class `java.util.Vector`

Declaration

```
public final synchronized void insertElementAt(Object obj,  
                                              int index)
```

Description

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

Applies to

Class `java.util.Vector`

Declaration

```
public final synchronized void addElement(Object obj)
```

Description

Adds the specified object as the last element of the vector.

Parameters:

obj - the element to be added

removeElement

Applies to

Class `java.util.Vector`

Declaration

```
public final synchronized boolean removeElement(Object obj)
```

Description

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

Applies to

Class `java.util.Vector`

Declaration

```
public final synchronized void removeAllElements()
```

Description

Removes all elements of the vector. The vector becomes empty.

clone

Applies to

Class `java.util.Vector`

Declaration

```
public synchronized Object clone()
```

Description

Clones this vector. The elements are not cloned.

Overrides:

clone in class `Object`

toString

Applies to

Class `java.util.Vector`

Declaration

```
public final synchronized String toString()
```

Description

Converts the vector to a string. Useful for debugging.

Overrides:

toString in class Object

Class java.util.Stack

java.lang.Object

|

+----java.util.Vector

|

+----java.util.Stack

Declaration

```
public class Stack
```

```
extends Vector
```

Description

A Last-In-First-Out(LIFO) stack of objects.

Constructors

[Stack\(\)](#)

Methods

[empty\(\)](#)

[peek\(\)](#)

[pop\(\)](#)

[push\(Object\)](#)

[search\(Object\)](#)

Stack

Applies to

Class `java.util.Stack`

Declaration

```
public Stack()
```

push

Applies to

Class `java.util.Stack`

Declaration

```
public Object push(Object item)
```

Description

Pushes an item onto the stack.

Parameters:

item - the item to be pushed on.

pop

Applies to

Class `java.util.Stack`

Declaration

```
public Object pop()
```

Description

Pops an item off the stack.

Throws: `EmptyStackException`

If the stack is empty.

peek

Applies to

Class `java.util.Stack`

Declaration

```
public Object peek()
```

Description

Peeks at the top of the stack.

Throws: `EmptyStackException`

If the stack is empty.

empty

Applies to

Class `java.util.Stack`

Declaration

```
public boolean empty()
```

Description

Returns true if the stack is empty.

search

Applies to

Class `java.util.Stack`

Declaration

```
public int search(Object o)
```

Description

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.

Interface `java.awt.peer.WindowPeer`

Declaration

```
public interface WindowPeer  
    extends Object  
    extends ContainerPeer
```

Methods

```
toBack()  
ToFront()
```

toFront

Applies to

Interface `java.awt.peer.WindowPeer`

Declaration

```
public abstract void toFront()
```

[toBack](#)

Applies to

Interface `java.awt.peer.WindowPeer`

Declaration

```
public abstract void toBack()
```


Packages

Applet API Packages

Java interfaces and classes are grouped into packages . The following Package Index lists all available packages, from which you can access interfaces and classes.

[java.applet](#)

[java.awt](#)

[java.awt.image](#)

[java.awt.peer](#)

[java.io](#)

[java.lang](#)

[java.net](#)

[java.util](#)

Other Packages

[sun.tools.debug](#)

package java.applet

Interfaces

Classes

Package that enables construction of applets. It also provides information about an applet's parent document, about other applets in that document, and enables an applet to play audio.

Interfaces

AppletContext

AppletStub

AudioClip

Classes

Applet

package java.awt

Interfaces

Classes

Exceptions

Errors

Package that provides user interface features such as windows, dialog boxes, buttons, checkboxes, lists, menus, scrollbars and text fields. (Abstract Window Toolkit)

Interfaces

LayoutManager

MenuContainer

Classes

<u>BorderLayout</u>	<u>Button</u>	<u>Canvas</u>
<u>CardLayout</u>	<u>Checkbox</u>	<u>CheckboxGroup</u>
<u>CheckboxMenuItem</u>	<u>Choice</u>	<u>Color</u>
<u>Component</u>	<u>Container</u>	<u>Dialog</u>
<u>Dimension</u>	<u>Event</u>	<u>FileDialog</u>
<u>FlowLayout</u>	<u>Font</u>	<u>FontMetrics</u>
<u>Frame</u>	<u>Graphics</u>	<u>GridBagConstraints</u>
<u>GridBagLayout</u>	<u>GridLayout</u>	<u>Image</u>
<u>Insets</u>	<u>Label</u>	<u>List</u>
<u>MediaTracker</u>	<u>Menu</u>	<u>MenuBar</u>
<u>MenuItem</u>	<u>MenuItem</u>	<u>Panel</u>
<u>Point</u>	<u>Polygon</u>	<u>Rectangle</u>
<u>Scrollbar</u>	<u>TextArea</u>	<u>TextComponent</u>
<u>TextField</u>	<u>Toolkit</u>	<u>Window</u>

Exceptions

AWTException

Errors

AWTError

package java.awt.image

Interfaces

Classes

Package for managing image data, such as the setting the color model, cropping, color filtering, setting pixel values and grabbing snapshots.

Interfaces

ImageConsumer

ImageObserver

ImageProducer

Classes

ColorModel

CropImageFilter

DirectColorModel

FilteredImageSource

ImageFilter

IndexColorModel

MemoryImageSource

PixelGrabber

RGBImageFilter

`package java.awt.peer`

Interfaces

Package that connects AWT components to their platform-specific implementation (such as Motif widgets or Microsoft Windows controls).

Interfaces

<u>ButtonPeer</u>	<u>CanvasPeer</u>	<u>CheckboxMenuItemPeer</u>
<u>CheckboxPeer</u>	<u>ChoicePeer</u>	<u>ComponentPeer</u>
<u>ContainerPeer</u>	<u>DialogPeer</u>	<u>FileDialogPeer</u>
<u>FramePeer</u>	<u>LabelPeer</u>	<u>ListPeer</u>
<u>MenuBarPeer</u>	<u>MenuComponentPeer</u>	<u>MenuItemPeer</u>
<u>MenuPeer</u>	<u>PanelPeer</u>	<u>ScrollbarPeer</u>
<u>TextAreaPeer</u>	<u>TextComponentPeer</u>	<u>TextFieldPeer</u>
<u>WindowPeer</u>		

package java.io

Interfaces

Classes

Exceptions

Package that provides a set of input and output streams to read and write data to files, strings, and other sources.

Interfaces

DataInput

DataOutput

FilenameFilter

Classes

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

Exceptions

EOFException

FileNotFoundException

IOException

InterruptedIOException

UTFDataFormatException

package java.lang

Interfaces

Classes

Exceptions

Errors

Package that contains essential Java classes, including numerics, strings, objects, compiler, runtime, security and threads. Unlike other packages, java.lang is automatically imported into every Java program.

Interfaces

Cloneable

Runnable

Classes

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

Exceptions

ArithmeticException

ArrayIndexOutOfBoundsException

ArrayStoreException

ClassCastException

ClassNotFoundException

CloneNotSupportedException

Exception

IllegalAccessException

IllegalArgumentException

IllegalMonitorStateException

IllegalThreadStateException

IndexOutOfBoundsException

InstantiationException

InterruptedException

NegativeArraySizeException

NoSuchMethodException

NullPointerException

NumberFormatException

RuntimeException

SecurityException

StringIndexOutOfBoundsException

Errors

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

package java.net

[Interfaces](#)

[Classes](#)

[Exceptions](#)

Package for network support, including URLs, TCP sockets, UDP sockets, IP addresses and a binary-to-text converter.

Interfaces

ContentHandlerFactory

SocketImplFactory

URLStreamHandlerFactory

Classes

ContentHandler

DatagramPacket

DatagramSocket

InetAddress

ServerSocket

Socket

SocketImpl

URL

URLConnection

URLEncoder

URLStreamHandler

Exceptions

MalformedURLException

ProtocolException

SocketException

UnknownHostException

UnknownServiceException

package java.util

Interfaces

Classes

Exceptions

Package containing miscellaneous utility classes, including generic data structures, settable bits class, time, date, string manipulation, random number generation, system properties, notification, and enumeration of data structures.

Interfaces

Enumeration

Observer

Classes

BitSet

Date

Dictionary

Hashtable

Observable

Properties

Random

Stack

StringTokenizer

Vector

Exceptions

EmptyStackException

NoSuchElementException

`package sun.tools.debug`

[Interfaces](#)

[Classes](#)

Package to support Java debugging and object inspection tools.

Interfaces

DebuggerCallback

Classes

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

Align property

Align in Label

[Label\(String text \[, int alignment\]\)](#)

[setAlignment\(int alignment\)](#)

[getAlignment\(\)](#)

Background property

Background in Component

[setBackground\(Color color\)](#)

[getBackground\(\)](#)

Caption property

Caption in Frame

[setTitle\(String title\)](#)

[getTitle\(\)](#)

Caption in Button

[Button\(\[String label\]\)](#)

[setLabel\(String label\)](#)

[getLabel\(\)](#)

Caption in CheckBox

[CheckBox\(\[String label\]\)](#)

[CheckBox\(String label, CheckBoxGroup group, boolean state\)](#)

[setLabel\(String label\)](#)

[getLabel\(\)](#)

Caption in Label

[Label\(String text \[, int alignment\]\)](#)

[setText\(String text\)](#)

[getText\(\)](#)

Caption in MenuItem

[MenuItem\(String label\)](#)

[setLabel\(String label\)](#)

[getLabel\(\)](#)

Checked property

Checked in CheckBox

[CheckBox\(String label, CheckBoxGroup group, boolean state\)](#)

[setState\(boolean checked\)](#)

[getState\(\)](#)

Checked in CheckBoxMenuItem

[setState\(boolean checked\)](#)

[getState\(\)](#)

Cursor property

Cursor in Frame

[setCursor\(int cursorType\)](#)

[getCursorType\(\)](#)

Enabled property

Enabled in Component

[enable\(\)](#)

[enable\(boolean enable\)](#)

[disable\(\)](#)

[isEnabled\(\)](#)

Enabled in MenuItem

[enable\(\)](#)

[enable\(boolean enable\)](#)

[disable\(\)](#)

[isEnabled\(\)](#)

Font property

Font in Component

[getFont\(\)](#)

[getFontMetrics\(\)](#)

[setFont\(Font font\)](#)

Font in FontMetrics

[FontMetrics\(Font font\)](#)

[getFont\(\)](#)

Font in Font

[Font\(String name, int style, int pointSize\)](#)

[getFont\(String propertyName \[, Font font\]\)](#)

FontName in Font

[getName\(\)](#)

FontSize in Font

[getSize\(\)](#)

FontStyle in Font

[isBold\(\)](#)

[isItalic\(\)](#)

[isPlain\(\)](#)

Foreground property

Foreground in Component

[setForeground\(Color color\)](#)

[getForeground\(\)](#)

Height property

Height in Component

[minimunSize\(\)](#)

[preferredSize\(\)](#)

[reshape\(int x, int y, int height, int width\)](#)

[resize\(int height, int width\)](#)

[size\(\)](#)

Height in Dimension

[Dimension\(int width, int height\)](#)

[height](#) variable

Items property

Items in Choice

[addItem\(String item\)](#)
[countItems\(\)](#)
[getItem\(\)](#)
[getSelectedIndex\(\)](#)
[getSelectedItem\(\)](#)
[select\(String item\)](#)
[select\(int position\)](#)

Items in List

[addItem\(String item, int index\)](#)
[clear\(\)](#)
[countItems\(\)](#)
[getItem\(\)](#)
[delItem\(int index\)](#)
[delItems\(int start, int end\)](#)
[getSelectedIndex\(\)](#)
[getSelectedIndexes\(\)](#)
[getSelectedItem\(\)](#)
[getSelectedItems\(\)](#)
[isSelected\(int index\)](#)
[replaceItem\(String newItem, int index\)](#)
[select\(int index\)](#)

Items in TextArea

[TextArea\(String text \[, int columns, int rows\]\)](#)
[appendText\(String text\)](#)
[insertText\(String text, int position\)](#)
[replaceText\(String newText, int start, int end\)](#)

[getText\(\)](#)

[setText\(String text\)](#)

Items in MenuBar

Menu

[Menu\(String title \[, boolean tearOff\]\)](#)

[add\(menuItem item\)](#)

[add\(String label\)](#)

[addSeparator\(\)](#)

[countItems\(\)](#)

[getItem\(int index\)](#)

[remove\(int index\)](#)

[remove\(MenuComponent item\)](#)

MenuItem

[MenuItem\(String label\)](#)

[getLabel\(\)](#)

[setLabel\(String label\)](#)

Name property

Name a component.

Left property

left in Component

[location\(\)](#)

[move\(int x, int y\)](#)

[reshape\(int x, int y, int height, int width\)](#)

LineInc property

LineInc in Scrollbar

[getLineIncrement\(\)](#)

[setLineIncrement\(int inc\)](#)

Max property

Max in Scrollbar

[Scrollbar\(int orientation, int value, int min, int max\)](#)

[getMaximum\(\)](#)

[setValues\(int value, int visible, int min, int max\)](#)

Min property

Min in Scrollbar

[Scrollbar\(int orientation, int value, int min, int max\)](#)

[getMinimum\(\)](#)

[setValues\(int value, int visible, int min, int max\)](#)

MultiSelect property

MultiSelect in List

[List\(int rows, boolean multiSelections\)](#)

[allowMultiSelections\(\)](#)

[setMultiSelections\(boolean multiSelections\)](#)

Orientation property

Orientation in Scrollbar

[ScrollBar\(int orientation\)](#)

[ScrollBar\(int orientation, int value, int min, int max\)](#)

[getOrientation\(\)](#)

PageInc property

PageInc in Scrollbar

[getPageIncrement\(\)](#)

[setPageIncrement\(int inc\)](#)

PasswdChar property

PasswdChar in TextField

[EchoCharIsSet\(\)](#)

[getEchoChar\(\)](#)

[setEchoCharacter\(char ch\)](#)

Resizable property

Resizable in Frame

[isResizable\(\)](#)

[setResizable\(boolean resizable\)](#)

Text property

Text in TextField

[TextField\(String initialText \[, int column\]\)](#)

[getText\(\)](#)

[setText\(String text\)](#)

Handling of selected text is done by methods of parent class, TextComponent.

[getSelectedText\(\)](#)

[getSelectionEnd\(\)](#)

[getSelectionStart\(\)](#)

[select\(int start, int end\)](#)

[selectAll\(\)](#)

Top property

Top in Component

[location\(\)](#)

[move\(int x, int y\)](#)

[reshape\(int x, int y, int height, int width\)](#)

Value property

Value in ScrollBar

[ScrollBar\(int orientation, int value, int min, int max\)](#)

[getValue\(\)](#)

[setValue\(int value\)](#)

[setValues\(int value, int visible int in, int max\)](#)

Visible property

Visible in Component

[hide\(\)](#)

[isVisible\(\)](#)

[isShowing\(\)](#)

[show\(\)](#)

[show\(boolean show\)](#)

Width property

Width in Component

[minimunSize\(\)](#)

[preferredSize\(\)](#)

[reshape\(int x, int y, int height, int width\)](#)

[resize\(int height, int width\)](#)

[size\(\)](#)

Width in Dimension

[Dimension\(int width, int height\)](#)

[width](#) [variable](#)

FileName property

FileName in Image

If you change [Picture](#) property FileName property is changed automatically.

And that FileName property does not include path like URL, so if you want to change the relative path of image file you have to put it in the FileName property editor in Object Inspector.

I designed this feature to be default for both Applets and Applications.

awt.Applet

[getImage\(URL url\)](#)

[getImage\(URL url, String name\)](#)

awt.Toolkit

[getImage\(String filename\)](#)

[getImage\(URL url\)](#)

FileName in OpenFileDialog

(under construction)

Picture property

Picture in Image

see [FileName property](#)

Program Structure

The source code for an Java program consists of one or more compilation units. Each compilation unit can contain only the following (in addition to white space and comments):

- a package statement (see Packages)
- import statements (see Packages)
- class declarations (see Classes)
- interface declarations (see Interfaces)

Although each Java compilation unit can contain multiple classes or interfaces, at most one class or interface per compilation unit can be public (see Classes).

When Java source code is compiled, the result is Java bytecode. Java bytecode consists of machine-independent instructions that can be interpreted efficiently by the Java runtime system. The Java runtime system operates like a virtual machine, for information see The Java Virtual Machine Specification.

In the current Java implementation, each compilation unit is a file with a ".java" suffix.

Comments

The Java language has three styles of comments:

```
// text
```

All characters from // to the end of the line are ignored.

```
/* text */
```

All characters from /* to */ are ignored.

```
/** text */
```

These comments are treated specially when they occur immediately before any declaration. They should not be used any other place in the code. These comments indicate that the enclosed text should be included in automatically generated documentation as a description of the declared item.

Identifiers

Identifiers must start with a letter, underscore (" _"), or dollar sign (" \$"); subsequent characters can also contain digits (0-9). Java uses the Unicode character set. For the purposes of determining what is a legal identifier the following are considered "letters:"

- The characters "A" through "Z"
- The characters "a" through "z"
- All Unicode characters with a character number above hex 00C0

Other characters valid after the first letter of an identifier include every character except those in the segment of Unicode reserved for special characters.

Thus, "garç;on" and "Mjø;lner" are legal identifiers, but strings containing characters such as "xa6 " are not.

For more information on the Unicode standard, see The Unicode Standard, Worldwide Character Encoding, Version 1.0, Volumes 1&2. The FTP address for Unicode, Inc. (formerly the Unicode Consortium) is unicode.org.

Keywords

The following identifiers are reserved for use as keywords. They cannot be used in any other way.

Integer Literals

Integers can be expressed in decimal (base 10), hexadecimal (base 16), or octal (base 8) format. A decimal integer literal consists of a sequence of digits (optionally suffixed as described below) without a leading 0 (zero). An integer can be expressed in octal or hexadecimal rather than decimal. A leading 0 (zero) on an integer literal means it is in octal; a leading 0x (or 0X) means hexadecimal. Hexadecimal integers can include digits (0-9) and the letters a-f and A-F. Octal integers can include only the digits 0-7.

Integer literals are of type `int` unless they are larger than 32-bits, in which case they are of type `long` (see Integer Types). A literal can be forced to be long by appending an `L` or `l` to its value.

The following are all legal integer literals:

```
2, 2L 0777 0xDeadBeef
```

Floating Point Literals

A floating point literal can have the following parts: a decimal integer, a decimal point ((".")), a fraction (another decimal number), an exponent, and a type suffix. The exponent part is an e or E followed by an integer, which can be signed. A floating point literal must have at least one digit, plus either a decimal point or e (or E). Some examples of floating point literals are:

3.1415 3.1E12 .1e12 2E12

As described in Floating Point Types , the Java language has two floating point types: float (IEEE 754 single precision) and double (IEEE 754 double precision). You specify the type of a floating point literal as follows:

2.0d or 2.0D double

2.0f or 2.0F or 2.0 float

Boolean Literals

The boolean type has two literal values: true and false. See [Boolean Types](#) for more information on boolean values.

Character Literals

A character literal is a character (or group of characters representing a single character) enclosed in single quotes. Characters have type `char` and are drawn from the Unicode character set (see [Character Types](#)). The following escape sequences allow for the representation of some non-graphic characters as well as the single quote, `"` and the backslash `\`, in Java code:

String Literals

A string literal is zero or more characters enclosed in double quotes. Each string literal is implemented as a String object (not as an array of characters). For example, "abc" creates a new instance of class String. The following are all legal string literals:

```
"" \ the empty string
```

```
"\"
```

```
"This is a string"
```

```
"This is a \  
two-line string"
```

Operators and Miscellaneous Separators

The following characters are used in source code as operators or separators:

+ - ! % ^ & * | ~ / > < ;

() { } [] ; ? : , . =

In addition, the following character combinations are used as operators:

++ -- == <= >= != << >>;

>>>; += -= *= /= &|=

^= %= <<= >>= >>>= || &&;

Types

Every variable and every expression has a type. Type determines the allowable range of values a variable can hold, allowable operations on those values, and the meanings of the operations. Built-in types are provided by the Java language. Programmers can compose new types using the class and interface mechanisms (see [Classes](#) and [Interfaces](#)).

The Java language has two kinds of types: simple and composite. Simple types are those that cannot be broken down; they are atomic. The integer, floating point, boolean, and character types are all simple types. Composite types are built on simple types. The language has three kinds of composite types: arrays, classes, and interfaces. Simple types and arrays are discussed in this section.

Integer Types

Integers are similar to those in C and C++, with two exceptions: all integer types are machine independent, and some of the traditional definitions have been changed to reflect changes in the world since C was introduced. The four integer types have widths of 8, 16, 32, and 64 bits and are signed.

A variable's type does not directly affect its storage allocation. Type only determines a variable's arithmetic properties and legal range of values. If a value is assigned to a variable that is outside the legal range of the variable, the value is reduced modulo the range.

Floating Point Types

The float keyword denotes single precision (32 bit); double denotes double precision (64 bit). The result of a binary operator on two float operands is a float. If either operand is a double, the result is a double.

Floating point arithmetic and data formats are defined by IEEE 754. See Appendix: Floating Point for details on the floating point implementation.

Character Types

The language uses the Unicode character set throughout. Consequently the char data type is defined as a 16-bit unsigned integer.

Boolean Types

The boolean type is used for variables that can be either true or false, and for methods that return true and false values. It's also the type that is returned by the relational operators (e.g., ">=").

Boolean values are not numbers and cannot be converted into numbers by casting.

Arrays

Arrays in the language are first class objects. They replace pointer arithmetic. All objects (including arrays) are referred to by pointers that cannot be damaged by being manipulated as numbers. Arrays are created using the new operator:

```
char s[] = new char[30];
```

The first element of an array is at index 0 (zero). Specifying dimensions in the declarations is not allowed. Every allocation of an array must be explicit--use new every time:

```
int i[] = new int[3];
```

The language does not support multi-dimensional arrays. Instead, programmers can create arrays of arrays:

```
int i[][] = new int[3][4];
```

At least one dimension must be specified but other dimensions can be explicitly allocated by a program at a later time. For example:

```
int i[][] = new int[3][];
```

is a legal declaration.

In addition to the C-style array declaration, where brackets follow the name of the variable or method, Java allows brackets following the array element type. The following two lines are equivalent:

```
int iarray[];  
int[] iarray;
```

as are the following method declarations:

```
byte    f( int n )[];  
byte[]  f( int n );
```

Subscripts are checked to make sure they're valid:

```
int a[] = new int[10];
a[5] = 1;
a[1] = a[0] + a[2];
a[-1]      = 4;      // Throws an ArrayIndexOutOfBoundsException
                        // at runtime
a[10] = 2;           // Throws an ArrayIndexOutOfBoundsException
                        // at runtime
```

Array dimensions must be integer expressions:

```
int n;
...
float arr[] = new float[n + 1];
```

The length of any array can be found by using `.length`:

```
int a[][] = new int[10][3];
println(a.length);           // prints 10
println(a[0].length);        // prints 3
```

Arrays are instances of subclasses of class `Object`. In the class hierarchy there is a class named `Array`, which has one instance variable, `"length"`. For each primitive type there is a corresponding subclass of `Array`. Similarly, for all classes a corresponding subclass of `Array` implicitly exists. For example:

```
new Thread[n]
```

creates an instance of `Thread[]`. If class `A` is a superclass of class `B` (i.e., `B` extends `A`) then `A[]` is a superclass of `B[]` (see the diagram below).

Hence, you can assign an array to an `Object`:

```
Object o;
```

```
int a[] = new int[10];  
o = a;
```

and you can cast an Object to an array:

```
a = (int[])o;
```

Array classes cannot be explicitly subclassed.

Classes

Classes represent the classical object oriented programming model. They support data abstraction and implementations tied to data. In Java, each new class creates a new type.

To make a new class, the programmer must base it on an existing class. The new class is said to be derived from the existing class. The derived class is also called a subclass of the other, which is known as a superclass. Class derivation is transitive: if B is a subclass of A, and C is a subclass of B, then C is a subclass of A.

The immediate superclass of a class and the interfaces (see Interfaces) that the class implements (if any) are indicated in the class declaration by the keywords `extends` and `implements`, respectively:

```
[Doc comment] [Modifiers] class Classname

    extends Superclassname]

    implements Interface[, Interface]] {

        ClassBody

    }
```

For example:

```
/** 2 dimensional point */
public class Point {
    float x, y;
    ...
}

/** Printable point */
class PrintablePoint extends Points implements Printable {
    ...
    public void print() {
        ...
    }
}
```

```
}
```

All classes are derived from a single root class: Object. Every class except Object has exactly one immediate superclass. If a class is declared without specifying an immediate superclass, Object is assumed. For example, the following:

```
class Point {  
    float x, y;  
}
```

is the same as:

```
class Point extends Object {  
    float x, y;  
}
```

The language supports only single inheritance. Through a feature known as interfaces, it supports some features that in other languages are supported through multiple inheritance (see Interfaces).

Casting Between Class Types

The language supports casting between types and because each class is a new type, Java supports casting between class types. If B is a subclass of A, then an instance of B can be used as an instance of A. No explicit cast is required, but an explicit cast is legal--this is called widening. If an instance of A needs to be used as if it were an instance of B, the programmer can write a type conversion or cast--this is called narrowing. Casts from a class to a subclass are always checked at runtime to make sure that the object is actually an instance of the subclass (or one of its subclasses). Casting between sibling classes is a compile-time error. The syntax of a class cast is:

```
(classname)ref
```

where (classname) is the object being cast to and ref is the object being cast.

Casting affects only the reference to the object, not the object itself. However, access to instance variables is affected by the type of the object reference. Casting an object from one type to another may result in a different instance variable being accessed even though the same variable name is used.

```
class ClassA {
    String name = "ClassA";
}

class ClassB extends ClassA { // ClassB is a subclass of ClassA
    String name = "ClassB";
}

class AccessTest {
    void test() {
        ClassB b = new ClassB();
        println(b.name);      // print: ClassB

        ClassA a;
        a = (ClassA)b;
        println(a.name);      // print: ClassA
    }
}
```


Methods

Methods are the operations that can be performed on an object or class. They can be declared in either classes or interfaces, but they can be implemented only in classes. (All user-defined operations in the language are implemented with methods.)

A method declaration in a class has the following form (native and abstract methods have no method body):

```
[Doc comment] [Modifiers] returnType methodName ( parameterList ) {  
  
    [methodBody]  
  
}
```

Methods:

- Have a return type unless they're constructors, in which case they have no return type. If a non-constructor method does not return any value, it must have a void return type.
- Have a parameter list consisting of comma-separated pairs of types and parameter names. The parameter list should be empty if the method has no parameters.

Variables declared in methods (local variables) can't hide other local variables or parameters in the same method. For example, if a method is implemented with a parameter named `i`, it's a compile-time error for the method to declare a local variable named `i`. In the following example:

```
class Rectangle {  
    void vertex(int i, int j) {  
        for (int i = 0; i <= 100; i++) {    // ERROR  
            ...  
        }  
    }  
}
```

the declaration of "i" in the for loop of the method body of "vertex" is a compile-time error.

The language allows polymorphic method naming--declaring a method with a name that has already been used in the class or its superclass--for overriding and overloading methods. Overriding means providing a different implementation of an inherited method. Overloading means declaring a method that has the same name as another method, but a different parameter list.

Note: Return types are not used to distinguish methods. Within a class scope, methods that have the same name and parameter list, i.e., the same number, position, and types of parameters, must return the same type. It is a compile-time error to declare such a method with a different return type.

Instance Variables

All variables in a class declared outside the scope of a method and not marked static (see Static Methods, Variables, and Initializers) are instance variables. (Variables declared inside the scope of a method are considered local variables.) Instance variables can have modifiers (see Modifiers).

Instance variables can be of any type and can have initializers. If an instance variable does not have an initializer, it is initialized to zero; boolean variables are initialized to false; and objects are initialized to null. An example of an initializer for an instance variable named `j` is:

```
class A {  
    int j = 23;  
    ...  
}
```

The this and super Variables

Inside the scope of a non-static method, the name `this` represents the current object. For example, an object may need to pass itself as an argument to another object's method:

```
class MyClass {  
    void aMethod(OtherClass obj) {  
        ...  
        obj.Method(this);  
        ...  
    }  
}
```

Any time a method refers to its own instance variables or methods, an implicit `"this."` is in front of each reference:

```
class Foo {  
    int a, b, c;  
    ...  
    void myPrint(){  
        print(a + "\n");    // a == "this.a"  
    }  
    ...  
}
```

The `super` variable is similar to the `this` variable. The `this` variable contains a reference to the current object; its type is the class containing the currently executing method. The `super` variable contains a reference which has the type of the superclass.

Setting Local Variables

Methods are rigorously checked to be sure that all local variables (variables declared inside a method) are set before they are referenced. Using a local variable before it is initialized is a compile-time error.

Overriding Methods

To override a method, a subclass of the class that originally declared the method must declare a method with the same name, return type (or a subclass), and parameter list. When the method is invoked on an instance of the subclass, the new method is called rather than the original method. The overridden method can be invoked using the super variable such that:

```
setThermostat(...)    // refers to the overriding method  
super.setThermostat(...) // refers to the overridden method
```


Overload Resolution

Overloaded methods have the same name as an existing method, but differ in the number and/or the types of arguments. Overload resolution involves determining which overloaded method to invoke. The return type is not considered when resolving overloaded methods. Methods may be overloaded within the same class. The order of method declaration within a class is not significant.

Methods may be overloaded by varying both the number and the type of arguments. The compiler determines which matching method has the lowest type conversion cost. Only methods with the same name and number of arguments are considered for matching. The cost of matching a method is the maximum cost of converting any one of its arguments. There are two types of arguments to consider: object types and base types.

The cost of converting among object types is the number of links in the class tree between the actual parameter's class and the prototype parameter's class. Only widening conversions are considered. (See [Casting Between Class Types](#) for more information on object conversion.) No conversion is necessary for argument types that match exactly, making their cost 0.

The cost of converting base types is calculated from the table below. Exact matches cost 0.

Cost ≥ 10 causes data loss.

Once a conversion cost is assigned to each matching method, the compiler chooses the method which has the lowest conversion cost. If there is more than one potential method with the same lowest cost the match is ambiguous and a compile-time error occurs.

For example:

```
class A {  
    int method(Object o, Thread t);  
    int method(Thread t, Object o);  
  
    void g(Object o, Thread t) {  
        method(o, t);    // calls the first method.  
        method(t, o);    // calls the second method.  
    }  
}
```

```
        method(t, t);    // ambiguous - compile-time error
    }
}
```

Note: The names of parameters are not significant. Only the number, type, and order are.

Constructors

Constructors are special methods provided for initialization. They are distinguished by having the same name as their class and by not having any return type. Constructors are automatically called upon the creation of an object. They cannot be called explicitly through an object. If you want to be able to call the constructor outside the package, make the constructor public (see Access Specifiers for more information).

Constructors can be overloaded by varying the number and types of parameters, just as any other method can be overloaded.

```
class Foo {
    int x;
    float y;
    Foo() {
        x = 0;
        y = 0.0;
    }
    Foo(int a) {
        x = a;
        y = 0.0;
    }
    Foo(float a) {
        x = 0;
        y = a;
    }
    Foo(int a, float b) {
        x = a;
        y = b;
    }
    static void myFoo() {
        Foo obj1 = new Foo();      //calls Foo();
        Foo obj2 = new Foo(4);     //calls Foo(int a);
        Foo obj3 = new Foo(4.0);   //calls Foo(float a);
        Foo obj4 = new Foo(4, 4.0); //calls Foo(int a, float b);
    }
}
```

```
}
```

The instance variables of superclasses are initialized by calling either a constructor for the immediate superclass or a constructor for the current class. If neither is specified in the code, the superclass constructor that has no parameters is invoked. If a constructor calls another constructor in this class or a constructor in the immediate super class, that call must be the first thing in the constructor body. Instance variables can't be referenced before calling the constructor.

Invoking a constructor of the immediate superclass is done as follows:

```
class MyClass extends OtherClass {  
    MyClass(someParameters) {  
        /* Call immediate superclass constructor */  
        super(otherParameters);  
        ...  
    }  
    ...  
}
```

Invoking a constructor in the current class is done as follows:

```
class MyClass extends OtherClass {  
    MyClass(someParameters) {  
        ...  
    }  
    MyClass(otherParameters) {  
        /* Call the constructor in this class that has the  
           specified parameter list. */  
        this(someParameters);  
        ...  
    }  
    ...  
}
```

The Foo and FooSub methods below are examples of constructors.

```

class Foo extends Bar {
    int a;
    Foo(int a) {
        // implicit call to Bar()
        this.a = a;
    }
    Foo() {
        this(42);    // calls Foo(42) instead of Bar()
    }
}

```

```

class FooSub extends Foo {
    int b;
    FooSub(int b) {
        super(13);    // calls Foo(13); without this line,
                     // would have called Foo()
        this.b = b;
    }
}

```

If a class declares no constructors, the compiler automatically generates one of the following form:

```

class MyClass extends OtherClass {
    MyClass() {    // automatically generated
        super();
    }
}

```

Object Creation--the new Operator

A class is a template used to define the state and behavior of an object. An object is an instance of a class. All instances of classes are allocated in a garbage collected heap. Declaring a reference to an object does not allocate any storage for that object. The programmer must explicitly allocate the storage for objects, but no explicit deallocation is required; the garbage collector automatically reclaims the memory when it is no longer needed.

To allocate storage for an object, use the new operator. In addition to allocating storage, new initializes the instance variables and then calls the instance's constructor. The constructor is a method that initializes an object (see Constructors). The following syntax allocates and initializes a new instance of a class named ClassA:

```
a = new ClassA();
```

This constructor syntax provides arguments to the constructor:

```
b = new ClassA(3,2);
```

A third form of allocator allows the class name to be provided as a String expression. The String is evaluated at runtime, and new returns an object of type Object, which must be cast to the desired type.

```
b = new ( "Class"+"A" );
```

In this case, the constructor without arguments is called.

Garbage Collection

The garbage collector makes most aspects of storage management simple and robust. Programs never need to explicitly free storage: it is done for them automatically. The garbage collector never frees pieces of memory that are still referenced, and it always frees pieces that are not. This makes both dangling pointer bugs and storage leaks impossible. It also frees designers from having to figure out which parts of a system have to be responsible for managing storage.

Finalization

The Java language includes the concept of object finalization. Java finalization is generalization of garbage collection that allows a program to free arbitrary resources (e.g., file descriptors or graphics contexts) owned by objects that cannot be accessed by any Java program. Reclaiming an object's memory by garbage collection does not guarantee that these resources will be reclaimed*1.

The null Reference

The keyword `null` is a predefined constant that represents "no instance." `null` can be used anywhere an instance is expected and can be cast to any class type.

Static Methods, Variables, and Initializers

Variables and methods declared in a class can be declared static, which makes them apply to the class itself, rather than to an instance of the class. In addition, a block of code within a class definition can be declared static. Such a block of code is called a static initializer.

Static variables can have initializers, just as instance variables can. See [Order of Initialization](#) for more information. A static variable exists only once per class, no matter how many instances of the class exist. Both static variables and static methods are accessed using the class name. For convenience, they can also be accessed using an instance of the class.

```
class Ahem {
    int i;                // Instance variable
    static int j;         // Static variable
    static int arr[] = new int[12];
    static {              // static initializer:
                          // initialize the array
        for (int i = 0; i < arr.length; i++) {
            arr[i] = i;
        }
    }

    void seti(int i) {    // Instance method
        this.i = i;
    }

    static void setj(int j) { // Static method
        Ahem.j = j;
    }
}

static void clearThroat() {
    Ahem a = new Ahem();
    Ahem.j = 2;          // valid; static var via class
    a.j = 3;             // valid; static var via instance
    Ahem.setj(2);        // valid; static method via class
    a.setj(3);           // valid; static method via instance
    a.i = 4;             // valid; instance var via instance
    Ahem.i = 5;          // ERROR; instance var via class
}
```

```
    a.seti(4);      // valid; instance method via instance
    Ahem.seti(5);   // ERROR; instance method via class
  }
}
```

Order of Declarations

The order of declaration of classes and the methods and instance variables within them is irrelevant. However, it is possible for cycles to exist during initialization. For information on cycles during initialization see [Order of Initialization](#) . Methods are free to make forward references to other methods and instance variables. The following is legal:

```
class A {
    void a() {
        f.set(42);
    }
    B f;
}
class B {
    void set(long n) {
        this.n = n; }
    long n;
}
```

Order of Initialization

When a class is loaded, all of its static initialization code is executed. Static initializers are executed at the same time that static variables are initialized. The initializations occur in lexical order. For example, a class C is declared as follows:

```
class C {  
    static int a = 1;  
    static {  
        a++;  
        b = 7;  
    }  
    static int b = 2;  
}
```

When class C is loaded, the following occurs in order:

- a is set to 1
- the static initializer is executed, setting a to 2 and b to 7
- b is set to 2

If any static initialization code has a reference to some other, unloaded class, that class is loaded and its static initialization code is executed first. Each unloaded class referenced during static initialization is loaded and initialized before the class that referenced it. If at any time during this initialization sequence a reference is made to an uninitialized class that is earlier in the sequence, a cycle is created. A cycle causes a `NoClassDefFoundException` to be thrown.

For example, if ClassA is loaded, its static initialization code is executed. However, ClassA's static initialization code can have a reference to another unloaded class, for example, ClassB. In that case, ClassB is loaded and its static initialization occurs before ClassA's. Then, ClassA's static initializations are executed. A cycle is created if ClassB has a reference to ClassA in its static initialization code.

It is a compile-time error for instance or static variable initializations to have a forward dependency. For example, the following code:

```
int i = j + 2;  
int j = 4;
```

results in a compile-time error.

An instance variable's initialization can have an apparent forward dependency on a static variable. For example in the following code fragment:

```
int i = j + 2;           // Instance variable  
static int j = 4;       // Static variable
```

it appears that `i` has a forward dependency on `j`. However, `i` is initialized to 6 and `j` is initialized to 4. This initialization occurs because `j` is a static variable and is initialized before the instance variable. Thus, `j` is initialized to 4 before `i` is initialized.

Static methods cannot refer to instance variables; they can only use static variables and static methods.

Access Specifiers

Access specifiers are modifiers that allow programmers to control access to methods and variables. The keywords used to control access are `public`, `private`, and `protected`. Methods marked as `public` can be accessed from anywhere by anyone. Methods marked as `private` can be accessed only from within the class in which they are declared. Since `private` methods are not visible outside the class, they are effectively final and cannot be overridden (see [Final Classes, Methods, and Variables](#) for more information). Moreover, you cannot override a non-private method and give it `private` access. The `protected` access specifier makes a variable or method accessible to subclasses, but not to any other classes.

`Public` access can be applied to classes, methods, and variables. Classes, methods, and variables marked as `public` can be accessed from anywhere by any other class or method. The access of a `public` method cannot be changed by overriding it.

Classes, methods, and variables that do not have either `private` or `public` access specified can be accessed only from within the package where they are declared (see [Packages](#)).

Variable Scoping Rules

Within a package, when a class is defined as a subclass of another, declarations made in the superclass are visible in the subclass. When a variable is referenced inside a method definition, the following scoping rules are used:

1. The current block is searched first, and then all enclosing blocks, up to and including the current method. This is considered the local scope.

After the local scope, the search continues in the class scope:

1. The variables of the current class are searched.
2. If the variable is not found, variables of all superclasses are searched, starting with the immediate superclass, and continuing up through class Object until the variable is found. If the variable is not found, imported classes and package names are searched. If it is not found, it is a compile-time error.

Multiple variables with the same name within the same class are not allowed and result in a compile-time error.

Threadsafe Variables

An instance or static variable can be marked threadsafe to indicate that the variable will never be changed by some other thread while one thread is using it, i.e., the variable never changes asynchronously. The purpose of marking a variable as threadsafe is to allow the compiler to perform some optimizations that may mask the occurrence of asynchronous changes. The primary optimization enabled by the use of threadsafe is the caching of instance variables in registers.

Transient Variables

The transient flag is available to the interpreter and is intended to be used for persistent objects. Variables marked transient are treated specially when instances of the class are written out as persistent objects.

Final Classes, Methods, and Variables

The final keyword is a modifier that marks a class as never having subclasses, a method as never being overridden, or a variable as having a constant value. It is a compile-time error to override a final method, subclass a final class, or change the value of a final variable. Variables marked as final behave like constants.

Using final lets the compiler perform a variety of optimizations. One such optimization is inline expansion of method bodies, which may be done for small, final methods (where the meaning of small is implementation dependent).

Examples of the various final declarations are:

```
class Foo {  
    final int value = 3;           // final variable  
    final int foo(int a, int b) {  // final method  
        ...  
    }  
}
```

Native Methods

Methods marked as native are implemented in a platform-dependent language, e.g., C, not Java. Native methods do not have a method body, instead the declaration is terminated with a semicolon. Constructors cannot be marked as native. Though implemented in a platform-dependent language, native methods behave exactly as non-native methods do, for example, it is possible to override them. An example of a native method declaration is:

```
native long timeOfDay();
```

Abstract Methods

Abstract methods provide the means for a superclass or interface to define a protocol that subclasses must implement. Methods marked as abstract must be defined in a subclass of the class in which they are declared. An abstract method does not have a method body; instead the declaration is terminated with a semi-colon.

The following rules apply to the use of the abstract keyword:

- Constructors cannot be marked as abstract.
- Static methods cannot be abstract.
- Private methods cannot be abstract.
- Abstract methods must be defined in some subclass of the class in which they are declared.
- A method that overrides a superclass method cannot be abstract.
- Classes that contains abstract methods and classes that inherit abstract methods without overriding them are considered abstract classes.
- It is a compile-time error to instantiate an abstract class or attempt to call an abstract method directly.

Synchronized Methods and Blocks

The `synchronized` keyword is a modifier that marks a method or block of code as being required to acquire a lock. The lock is necessary so that the synchronized code does not run at the same time as other code that needs access to the same resource. Each object has exactly one lock associated with it; each class also has exactly one lock. Synchronized methods are reentrant.

When a synchronized method is invoked, it waits until it can acquire the lock for the current instance (or class, if it's a static method). After acquiring the lock, it executes its code and then releases the lock.

Synchronized blocks of code behave similarly to synchronized methods. The difference is that instead of using the lock for the current instance or class, they use the lock associated with the object or class specified in the block's `synchronized` statement.

Synchronized blocks are declared as follows:

```
/* ...preceding code in the method... */
synchronized(<object or class name>;) {    //sync. block
    /* code that requires synchronized access */
}
/* ...remaining code in the method... */
```

An example of the declaration of a synchronized method is:

```
class Point {
    float x, y;
    synchronized void scale(float f) {
        x *= f;
        y *= f;
    }
}
```

An example of a synchronized block is:

```
class Rectangle {
    Point topLeft;
```

```
...  
void print() {  
    synchronized (topLeft) {  
        println("topLeft.x = " + topLeft.x);  
        println("topLeft.y = " + topLeft.y);  
    }  
    ...  
}  
}
```

Interfaces

An interface specifies a collection of methods without implementing their bodies. Interfaces provide encapsulation of method protocols without restricting the implementation to one inheritance tree. When a class implements an interface, it generally must implement the bodies of all the methods described in the interface. (If the implementing class is abstract--never implemented--it can leave the implementation of some or all of the interface methods to its subclasses.)

Interfaces solve some of the same problems that multiple inheritance does without as much overhead at runtime. However, because interfaces involve dynamic method binding, there is a small performance penalty to using them.

Using interfaces allows several classes to share a programming interface without having to be fully aware of each other's implementation. The following example shows an interface declaration (with the interface keyword) and a class that implements the interface:

```
public interface Storing {
    void freezeDry(Stream s);
    void reconstitute(Stream s);
}
public class Image implements Storing, Painting {
    ...
    void freezeDry(Stream s) {
        // JPEG compress image before storing
        ...
    }
    void reconstitute (Stream s) {
        // JPEG decompress image before reading
        ...
    }
}
```

Like classes, interfaces are either private (the default) or public. The scope of public and private interfaces is the same as that of public and private classes, respectively. Methods in an interface are always public. Variables are public, static, and final.

Interfaces as Types

The declaration syntax `interfaceName variableName` declares a variable or parameter to be an instance of some class that implements `interfaceName`. Interfaces behave exactly as classes when used as a type. This lets the programmer specify that an object must implement a given interface, without having to know the exact type or inheritance of that object. Using interfaces makes it unnecessary to force related classes to share a common abstract superclass or to add methods to `Object`.

The following pseudocode illustrates the `interfaceName variableName` syntax:

```
class StorageManager {  
    Stream stream;  
    ...  
    // Storing is the interface name  
    void pickle(Storing obj) {  
        obj.freezeDry(stream);  
    }  
}
```

Methods in Interfaces

Methods in interfaces are declared as follows:

```
returnType methodName ( parameterList );
```

The declaration contains no modifiers. All methods specified in an interface are public and abstract and no other modifiers may be applied.

See Abstract Methods for more information on abstract methods.

Variables in Interfaces

Variables declared in interfaces are final, public, and static. No modifiers can be applied. Variables in interfaces must be initialized.

Combining Interfaces

Interfaces can incorporate one or more other interfaces, using the extends keyword as follows:

```
interface DoesItAll extends Storing, Painting {  
    void doesSomethingElse();  
}
```

Packages

Packages are groups of classes and interfaces. They are a tool for managing a large namespace and avoiding conflicts. Every class and interface name is contained in some package. By convention, package names consist of period-separated words, with the first name representing the organization that developed the package.

Specifying a Compilation Unit's Package

The package that a compilation unit is in is specified by a package statement. When this statement is present, it must be the first non-comment, non-white space line in the compilation unit. It has the following format:

```
package packageName;
```

When a compilation unit has no package statement, the unit is placed in a default package, which has no name.

Using Classes and Interfaces from Other Packages

The language provides a mechanism for making the definitions and implementations of classes and interfaces available across packages. The `import` keyword is used to mark classes as being imported into the current package. A compilation unit automatically imports every class and interface in its own package.

Code in one package can specify classes or interfaces from another package in one of two ways:

- By prefacing each reference to the class or interface name with the name of its package:

// prefacing with a package

```
acme.project.FooBar obj = new acme.project.FooBar();
```

- By importing the class or interface or the package that contains it, using an import statement. Importing a class or interface makes the name of the class or interface available in the current namespace. Importing a package makes the names of all of its public classes and interfaces available. The construct:

```
// import all classes from acme.project
```

```
import acme.project.*;
```

means that every public class from `acme.project` is imported.

The following construct imports a single class, `Employee_List`, from the `acme.project` package:

```
// import Employee_List from acme.project
```

```
import acme.project.Employee_List;
```

```
Employee_List obj = new Employee_List();
```

It is illegal to specify an ambiguous class name and doing so always generates a compile-time error.

Class names may be disambiguated through the use of a fully qualified class name, i.e., one that includes the name of the class's package.

Operators

The operators, from highest to lowest precedence, are:

. [] ()

++ -- ! ~ instanceof

* / %

+ -

<;<; >;>; >;>;>;

<; >; <;= >;=

== !=

&;

^

|

&;&;

||

?:

= op=

,

Operators on Integers

For operators with integer results, if any operand is long, the result type is long. Otherwise the result type is int--never byte, short, or char. Thus, if a variable `i` is declared a short or a byte, `i+1` would be an int. When a result outside an operator's range would be produced, the result is reduced modulo the range of the result type.

The unary integer operators are:

The `++` operator is used to express incrementing directly. Incrementing can also be expressed indirectly using addition and assignment. `++lvalue` means `lvalue+=1`. `++lvalue` also means `lvalue=lvalue+1` (as long as `lvalue` has no side effects). The `--` operator is used to express decrementing. The `++` and `--` operators can be used as both prefix and postfix operators.

The binary integer operators are:

Integer division rounds toward zero. Division and modulus obey the identity $(a/b)*b + (a\%b) == a$.

The only exceptions for integer arithmetic are caused by a divide or modulus by zero, which throw the `ArithmeticException`. An underflow generates zero. An overflow leads to wrap-around, i.e., adding 1 to the maximum integer wraps around to the minimum integer.

An `op=` assignment operator corresponds to each of the binary operators in the above table.

The integer relational operators `<`, `>`, `<=`, `>=`, `==`, and `!=` produce boolean results.

Operators on Boolean Values

Variables or expressions that are boolean can be combined to yield other boolean values. The unary operator `!` is boolean negation. The binary operators `&`, `|`, and `^` are the logical AND, OR, and XOR operators; they force evaluation of both operands. To avoid evaluation of right-hand operands, you can use the short-cut evaluation operators `&&` and `||`. You can also use `==` and `!=`. The assignment operators also work: `&=`, `|=`, `^=`. The ternary conditional operator `?:` works as it does in C.

Operators on Floating Point Values

Floating point values can be combined using the usual operators: unary -, binary +, -, *, and /; and the assignment operators +=, -=, *=, and /=. The ++ and -- operators also work on floating point values (they add or subtract 1.0). In addition, % and %= work on floating point values, i.e.,

$a \% b$

is the same as:

$a - ((\text{int})(a / b) * b)$

This means that $a \% b$ is the floating point equivalent of the remainder after division.

Floating point expressions involving only single-precision operands are evaluated using single-precision operations and produce single-precision results. Floating point expressions that involve at least one double-precision operand are evaluated using double-precision operations and produce double-precision results.

The language has no arithmetic exceptions for floating point arithmetic. Following the IEEE 754 floating point specification, the distinguished values Inf and NaN are used instead. Overflow generates Inf. Underflow generates 0. Divide by zero generate s Inf.

The usual relational operators are also available and produce boolean results: >, <, >=, <=, ==, !=. Because of the properties of NaN, floating point values are not fully ordered, so care must be taken in comparison. For instance, if $a < b$ is not true, it does not follow that $a > b$. Likewise, $a != b$ does not imply that $a > b \parallel a < b$. In fact, there may no ordering at all.

Floating point arithmetic and data formats are defined by IEEE 754, "Standard for Floating Point Arithmetic." See Appendix: Floating Point for details on the language's floating point implementation.

Operators on Arrays

The following:

```
<expression>[<expression>:]
```

gets the value of an element of an array. Legal ranges for the expression are from 0 to the length of the array minus 1. The range is checked only at runtime.

Operators on Strings

Strings are implemented as String objects (see [String Literals](#) for more information). The operator + concatenates Strings, automatically converting operands into Strings if necessary. If the operand is an object it can define a method call toString() that returns a String in the class of the object.

```
// Examples of the + operator used with strings
```

```
float a = 1.0;
```

```
print("The value of a is " + a + "\n");
```

```
String s = "a = " + a;
```

The += operator works on Strings. Note, that the left hand side (s1 in the following example) is evaluated only once.

```
s1 += a; //s1 = s1 + a; // a is converted to String if necessary
```

Operators on Objects

The binary operator `instanceof` tests whether the specified object is an instance of the specified class or one of its subclasses. For example:

```
if (thermostat instanceof MeasuringDevice) {  
    MeasuringDevice dev = (MeasuringDevice)thermostat;  
    ...  
}
```

determines whether `thermostat` is a `MeasuringDevice` object (an instance of `MeasuringDevice` or one of its subclasses).

Casts and Conversions

The Java language and runtime system restrict casts and conversions to help prevent the possibility of corrupting the system. Integers and floating point numbers can be cast back and forth, but integers cannot be cast to arrays or objects. Objects cannot be cast to base types. An instance can be cast to a superclass with no penalty, but casting to a subclass generates a runtime check. If the object being cast to a subclass is not an instance of the subclass (or one of its subclasses), the runtime system throws a `ClassCastException`.

Declarations

Declarations can appear anywhere that a statement is allowed. The scope of the declaration ends at the end of the enclosing block.

In addition, declarations are allowed at the head of for statements, as shown below:

```
for (int i = 0; i < 10; i++) {  
    ...  
}
```

Items declared in this way are valid only within the scope of the for statement. For example, the preceding code sample is equivalent to the following:

```
{  
    int i = 0;  
    for (; i < 10; i++) {  
        ...  
    }  
}
```

Expressions

Expressions are statements:

```
a = 3;  
  print(23);  
  foo.bar();
```

Control Flow

The following is a summary of control flow:

```
if(boolean) statement
else statement
switch(e1) {
    case e2: statements
    default: statements
}
break [label];
continue [label];
return e1;
for([e1]; [e2]; [e3]) statement
while(boolean) statement
do statement while(boolean);
label:statement
```

The language supports labeled loops and labeled breaks, for example:

```
outer: // the label
    for (int i = 0; i < 10; i++) {
        for (int j = 0; j < 10; j++) {
            if (...) {
                break outer;
            }
            if (...) {
            }
        }
    }
}
```

The use of labels in loops and breaks has the following rules:

- Any statement can have a label.
- If a break statement has a label it must be the label of an enclosing statement.

- If a continue statement has a label it must be the label of an enclosing loop.

Exceptions

When an error occurs in an Java program—for example, when an argument has an invalid value—the code that detects the error can throw an exception^{*1}. By default, exceptions result in the thread terminating after printing an error message. However, programs can have exception handlers that catch the exception and recover from the error.

Some exceptions are thrown by the Java runtime system. However, any class can define its own exceptions and cause them to occur using throw statements. A throw statement consists of the throw keyword followed by an object. By convention, the object should be an instance of Exception or one of its subclasses. The throw statement causes execution to switch to the appropriate exception handler. When a throw statement is executed, any code following it is not executed, and no value is returned by its enclosing method. The following example shows how to create a subclass of Exception and throw an exception.

```
class MyException extends Exception {  
}
```

```
class MyClass {  
    void oops() {  
        if (/* no error occurred */) {  
            ...  
        } else { /* error occurred */  
            throw new MyException();  
        }  
    }  
}
```

To define an exception handler, the program must first surround the code that can cause the exception with a try statement. After the try statement come one or more catch statements—one per exception class that the program can handle at that point. In each catch statement is exception handling code. For example:

```
try {  
    p.a = 10;  
} catch (NullPointerException e) {
```

```

        println("p was null");
    } catch (Exception e) {
        println("other error occurred");
    } catch (Object obj) {
        println("Who threw that object?");
    }
}

```

A catch statement is like a method definition with exactly one parameter and no return type. The parameter can be either a class or an interface. When an exception occurs, the nested try/catch statements are searched for a parameter that matches the exception class. The parameter is said to match the exception if it:

- is the same class as the exception; or
- is a superclass of the exception; or
- if the parameter is an interface, the exception class implements the interface.

The first try/catch statement that has a parameter that matches the exception has its catch statement executed. After the catch statement executes, execution resumes after the try/catch statement. It is not possible for an exception handler to resume execution at the point that the exception occurred. For example, this code fragment:

```

print("now ");
    try {
        print("is ");
        throw new MyException();
        print("a ");
    } catch(MyException e) {
        print("the ");
    }
print("time\n");

```

prints "now is the time". As this example shows, exceptions don't have to be used only for error handling, but any other use is likely to result in code that's hard to understand.

Exception handlers can be nested, allowing exception handling to happen in more than one place. Nested exception handling is often used when the first handler can't recover completely from the error, yet needs to execute some cleanup code (as shown in the following code example). To pass exception handling up to the next higher handler, use the throw keyword using the same object that was caught. Note that the method that rethrows the exception stops executing after the throw statement; it never returns.

```
try {  
    f.open();  
} catch(Exception e) {  
    f.close();  
    throw e;  
}
```

The finally Statement

The following example shows the use of a finally statement that is useful for guaranteeing that some code gets executed whether or not an exception occurs. For example, the following code example:

```
try {  
    // do something  
} finally {  
    // clean up after it  
}
```

is similar to:

```
try {  
    // do something  
} catch(Object e){  
    // clean up after it  
    throw e;  
}  
// clean up after it
```

The finally statement is executed even if the try block contains a return, break, continue, or throw statement. For example, the following code example always results in "finally" being printed, but "after try" is printed only if a != 10.

```
try {  
    if (a == 10) {  
        return;  
    }  
} finally {  
    print("finally\n");  
}  
print("after try\n");
```


Runtime Exceptions

This section contains a list of the exceptions that the Java runtime throws when it encounters various errors.

ArithmeticException

Attempting to divide an integer by zero or take a modulus by zero throw the ArithmeticException--no other arithmetic operation in Java throws an exception. For information on how Java handles other arithmetic errors see [Operators on Integers](#) and [Operators on Floating Point Values](#).

For example, the following code causes an ArithmeticException to be thrown:

```
class Arith {  
    public static void main(String args[]) {  
        int j = 0;  
        j = j / j;  
    }  
}
```

NullPointerException

An attempt to access a variable or method in a null object or a element in a null array throws a NullPointerException. For example, the accesses o.length and a[0] in the following class declaration throws a NullPointerException at runtime.

```
class Null {  
    public static void main(String args[]) {  
        String o = null;  
        int a[] = null;  
        o.length();  
        a[0] = 0;  
    }  
}
```

It is interesting to note that if you throw a null object you actually throw a NullPointerException.

IncompatibleClassChangeException

In general the `IncompatibleClassChangeException` is thrown whenever one class's definition changes but other classes that reference the first class aren't recompiled. Four specific changes that throw a `IncompatibleClassChangeException` at runtime are:

- A variable's declaration is changed from static to non-static in one class but other classes that access the changed variable aren't recompiled.
- A variable's declaration is changed from non-static to static in one class but other classes that access the changed variable aren't recompiled.
- A field that is declared in one class is deleted but other classes that access the field aren't recompiled.
- A method that is declared in one class is deleted but other classes that access the method aren't recompiled.

ClassCastException

A `ClassCastException` is thrown if an attempt is made to cast an object `O` into a class `C` and `O` is neither `C` nor a subclass of `C`. For more information on casting see [Casting Between Class Types](#).

The following class declaration results in a `ClassCastException` at runtime:

```
class ClassCast {  
    public static void main(String args[]) {  
        Object o = new Object();  
        String s = (String)o;           // the cast attempt  
        s.length();  
    }  
}
```

NegativeArraySizeException

A `NegativeArraySizeException` is thrown if an array is created with a negative size. For example, the following class definition throws a `NegativeArraySizeException` at runtime:

```
class NegArray {
    public static void main(String args[]) {
        int a[] = new int[-1];
        a[0] = 0;
    }
}
```

`OutOfMemoryException`

An `OutOfMemoryException` is thrown when the system can no longer supply the application with memory. The `OutOfMemoryException` can only occur during the creation of an object, i.e., when `new` is called. For example, the following code results in an `OutOfMemoryException` at runtime:

```
class Link {
    int a[] = new int[1000000];
    Link l;
}

class OutOfMem {
    public static void main(String args[]) {
        Link root = new Link();
        Link cur = root;
        while(true) {
            cur.l = new Link();
            cur = cur.l;
        }
    }
}
```

`NoClassDefFoundException`

A `NoClassDefFoundException` is thrown if a class is referenced but the runtime system cannot find the referenced class.

For example, class NoClass is declared:

```
class NoClass {  
    public static void main(String args[]) {  
        C c = new C();  
    }  
}
```

When NoClass is run, if the runtime system can't find C.class it throws the NoClassDefFoundException.

C.class must have existed at the time NoClass is compiled.

IncompatibleTypeException

An IncompatibleTypeException is thrown if an attempt is made to instantiate an interface. For example, the following code causes an IncompatibleTypeException to be thrown.

```
interface I {  
    }  
  
class IncompType {  
    public static void main(String args[]) {  
        I r = (I)new("I");  
    }  
}
```

ArrayIndexOutOfBoundsException

An attempt to access an invalid element in an array throws an ArrayIndexOutOfBoundsException. For example:

```
class ArrayOut {  
    public static void main(String args[]) {  
        int a[] = new int[0];  
        a[0] = 0;  
    }  
}
```

```
}
```

UnsatisfiedLinkException

An `UnsatisfiedLinkException` is thrown if a method is declared native and the method cannot be linked to a routine in the runtime.

```
class NoLink {  
    static native void foo();  
  
    public static void main(String args[]) {  
        foo();  
    }  
}
```

InternalException

An `InternalException` should never be thrown. It's only thrown if some consistency check in the runtime fails. Please send mail to java@java.Sun.COM if you have a reproducible case that throws this exception.

