---

package java.awt

Interface Index

- LayoutManager
- MenuContainer

Class Index

- BorderLayout
- Button
- Canvas
- CardLayout
- Checkbox
- CheckboxGroup
- CheckboxMenuItem
- Choice
- Color
- Component
- Container
- Dialog
- Dimension
- Event
- FileDialog
- FlowLayout
- Font
- FontMetrics
- Frame
- Graphics
- GridBagConstraints
- GridBagLayout
- GridLayout
- Image
- Insets
- Label
- List
- MediaTracker
- Menu
- MenuBar
- MenuComponent

- MenuItem
- Panel
- Point
- Polygon
- Rectangle
- Scrollbar
- TextArea
- TextComponent
- TextField
- Toolkit
- Window

Exception Index

- AWTException

Error Index

- AWTError

Interface java.awt.LayoutManager

public interface **LayoutManager**
extends Object

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

See Also:
Container

Method Index

Methods

addLayoutComponent

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

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

removeLayoutComponent

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

Removes the specified component from the layout.

Parameters:

comp – the component ot be removed

preferredLayoutSize

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

Calculates the preferred size dimensions for the specified panel given the components in the specified parent container.

Parameters:

parent – the component to be laid out

See Also:

minimumLayoutSize

minimumLayoutSize

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

Calculates the minimum size dimensions for the specified panel given the components in the specified parent container.

Parameters:

parent – the component to be laid out

See Also:

preferredLayoutSize

layoutContainer

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

Lays out the container in the specified panel.

Parameters:

parent – the component which needs to be laid out

Interface java.awt.MenuContainer

public interface **MenuContainer**
extends Object

The super class of all menu related containers.

Method Index

Methods

getFont

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

postEvent

```
public abstract boolean postEvent(Event evt)
```

remove

```
public abstract void remove(MenuComponent comp)
```

Class java.awt.BorderLayout

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

---

public class **BorderLayout**
extends Object
implements LayoutManager

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

Constructor Index

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

Method Index

- **addLayoutComponent**(String, Component)
  Adds the specified named component to the layout.
- **layoutContainer**(Container)
  Lays out the specified container.
- **minimumLayoutSize**(Container)
  Returns the minimum dimensions needed to layout the components contained in the specified target container.
- **preferredLayoutSize**(Container)
  Returns the preferred dimensions for this layout given the components in the specified target container.
- **removeLayoutComponent**(Component)
  Removes the specified component from the layout.
- **toString**()
  Returns the String representation of this BorderLayout's values.

Constructors

BorderLayout

```
public BorderLayout()
```

Constructs a new BorderLayout.

BorderLayout

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

Constructs a BorderLayout with the specified gaps.
Parameters:
hgap – the horizontal gap
vgap – the vertical gap

Methods

addLayoutComponent

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

Adds the specified named component to the layout.
Parameters:
name – the String name
comp – the component to be added

removeLayoutComponent

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

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

minimumLayoutSize

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

Returns the minimum dimensions needed to layout the components contained in the specified target container.
Parameters:
target – the Container on which to do the layout

See Also:
Container, preferredLayoutSize

preferredLayoutSize

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

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

Parameters:
target – the component which needs to be laid out

See Also:
Container, minimumLayoutSize

layoutContainer

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

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

Parameters:
target – the component being laid out

See Also:
Container

toString

```
public String toString()
```

Returns the String representation of this BorderLayout's values.

Overrides:
toString in class Object

Class java.awt.Button

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

---

public class **Button**
extends Component

A class that produces a labeled button component.

Constructor Index

- **Button**()
  Constructs a Button with no label.
- **Button**(String)
  Constructs a Button with a string label.

Method Index

- **addNotify**()
  Creates the peer of the button.
- **getLabel**()
  Gets the label of the button.
- **paramString**()
  Returns the parameter String of this button.
- **setLabel**(String)
  Sets the button with the specified label.

Constructors

Button

```
public Button()
```

Constructs a Button with no label.

Button

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

Constructs a Button with a string label.
Parameters:
label – the button label

Methods

addNotify

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

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

getLabel

```
public String getLabel()
```

Gets the label of the button.
See Also:
setLabel

setLabel

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

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

paramString

```
protected String paramString()
```

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

Class java.awt.Canvas

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

---

public class **Canvas**
extends Component

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

Constructor Index

- **Canvas**()

Method Index

- **addNotify**()
  Creates the peer of the canvas.
- **paint**(Graphics)
  Paints the canvas in the default background color.

Constructors

Canvas

```
public Canvas()
```

Methods

addNotify

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

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

Overrides:

addNotify in class Component

paint

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

Paints the canvas in the default background color.

Parameters:

g – the specified Graphics window

Overrides:

paint in class Component

Class java.awt.CardLayout

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

---

public class **CardLayout**
extends Object
implements LayoutManager

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

Constructor Index

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

Method Index

- **addLayoutComponent**(String, Component)
  Adds the specified component with the specified name to the layout.
- **first**(Container)
  Flip to the first card.
- **last**(Container)
  Flips to the last card of the specified container.
- **layoutContainer**(Container)
  Performs a layout in the specified panel.
- **minimumLayoutSize**(Container)
  Calculates the minimum size for the specified panel.
- **next**(Container)
  Flips to the next card of the specified container.
- **preferredLayoutSize**(Container)
  Calculates the preferred size for the specified panel.
- **previous**(Container)
  Flips to the previous card of the specified container.

- **removeLayoutComponent**(Component)
  Removes the specified component from the layout.
- **show**(Container, String)
  Flips to the specified component name in the specified container.
- **toString**()
  Returns the String representation of this CardLayout's values.

Constructors

CardLayout

```
public CardLayout()
```

Creates a new card layout.

CardLayout

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

Creates a card layout with the specified gaps.
Parameters:
hgap – the horizontal gap
vgap – the vertical gap

Methods

addLayoutComponent

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

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

removeLayoutComponent

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

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

preferredLayoutSize

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

Calculates the preferred size for the specified panel.

Parameters:

parent – the name of the parent container

Returns:

the dimensions of this panel.

See Also:

minimumLayoutSize

minimumLayoutSize

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

Calculates the minimum size for the specified panel.

Parameters:

parent – the name of the parent container

Returns:

the dimensions of this panel.

See Also:

preferredLayoutSize

layoutContainer

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

Performs a layout in the specified panel.

Parameters:

parent – the name of the parent container

first

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

Flip to the first card.

Parameters:

parent – the name of the parent container

next

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

Flips to the next card of the specified container.

Parameters:

parent – the name of the container

previous

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

Flips to the previous card of the specified container.

Parameters:

parent – the name of the parent container

last

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

Flips to the last card of the specified container.

Parameters:

parent – the name of the parent container

show

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

Flips to the specified component name in the specified container.

Parameters:

parent – the name of the parent container
name – the component name

toString

```
public String toString()
```

Returns the String representation of this CardLayout's values.

Overrides:

toString in class Object

Class java.awt.Checkbox

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

---

public class **Checkbox**
extends Component

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

Constructor Index

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

Method Index

- **addNotify**()
  Creates the peer of the Checkbox.
- **getCheckboxGroup**()
  Returns the checkbox group.
- **getLabel**()
  Gets the label of the button.
- **getState**()
  Returns the boolean state of the Checkbox.
- **paramString**()
  Returns the parameter String of this Checkbox.
- **setCheckboxGroup**(CheckboxGroup)
  Sets the CheckboxGroup to the specified group.

- **setLabel**(String)
  Sets the button with the specified label.
- **setState**(boolean)
  Sets the Checkbox to the specifed boolean state.

Constructors

Checkbox

```
public Checkbox()
```

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

Checkbox

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

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

Parameters:

label – the label on the Checkbox

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

Methods

addNotify

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

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

Overrides:
addNotify in class Component

getLabel

```
public String getLabel()
```

Gets the label of the button.
See Also:
setLabel

setLabel

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

Sets the button with the specified label.
Parameters:
label – the label of the button
See Also:
getLabel

getState

```
public boolean getState()
```

Returns the boolean state of the Checkbox.
See Also:
setState

setState

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

Sets the Checkbox to the specifed boolean state.
Parameters:
state – the boolean state
See Also:
getState

getCheckboxGroup

```
public CheckboxGroup getCheckboxGroup()
```

Returns the checkbox group.
See Also:
setCheckboxGroup

setCheckboxGroup

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

Sets the CheckboxGroup to the specified group.

Parameters:

g – the new CheckboxGroup

See Also:

getCheckboxGroup

paramString

```
protected String paramString()
```

Returns the parameter String of this Checkbox.

Overrides:

paramString in class Component

Class java.awt.CheckboxGroup

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

---

public class **CheckboxGroup**
extends Object

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

Constructor Index

- **CheckboxGroup**()
  Creates a new CheckboxGroup.

Method Index

- **getCurrent**()
  Gets the current choice.
- **setCurrent**(Checkbox)
  Sets the current choice to the specified Checkbox.
- **toString**()
  Returns the String representation of this CheckboxGroup's values.

Constructors

CheckboxGroup

```
public CheckboxGroup()
```

Creates a new CheckboxGroup.

Methods

getCurrent

```
public Checkbox getCurrent()
```

Gets the current choice.

setCurrent

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

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

Parameters:

box – the current Checkbox choice

toString

```
public String toString()
```

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

Overrides:

toString in class Object

Class java.awt.CheckboxMenuItem

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

---

public class **CheckboxMenuItem**
extends MenuItem

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

Constructor Index

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

Method Index

- **addNotify**()
  Creates the peer of the checkbox item.
- **getState**()
  Returns the state of this MenuItem.
- **paramString**()
  Returns the parameter String of this button.
- **setState**(boolean)
  Sets the state of this MenuItem if it is a Checkbox.

Constructors

CheckboxMenuItem

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

Creates the checkbox item with the specified label.
Parameters:
label – the button label

addNotify

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

Creates the peer of the checkbox item. This peer allows us to change the look of the checkbox item without changing its functionality.
Overrides:
addNotify in class MenuItem

getState

```
public boolean getState()
```

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

setState

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

Sets the state of this MenuItem if it is a Checkbox.
Parameters:
t – the specified state of the checkbox

paramString

```
public String paramString()
```

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

Class java.awt.Choice

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

---

public class **Choice**
extends Component

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

Constructor Index

- **Choice**()
  Constructs a new Choice.

Method Index

- **addItem**(String)
  Adds an item to this Choice.
- **addNotify**()
  Creates the Choice's peer.
- **countItems**()
  Returns the number of items in this Choice.
- **getItem**(int)
  Returns the String at the specified index in the Choice.
- **getSelectedIndex**()
  Returns the index of the currently selected item.
- **getSelectedItem**()
  Returns a String representation of the current choice.
- **paramString**()
  Returns the parameter String of this Choice.
- **select**(int)
  Selects the item with the specified postion.
- **select**(String)
  Selects the item with the specified String.

Constructors

Choice

```
public Choice()
```

Constructs a new Choice.

Methods

addNotify

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

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

countItems

```
public int countItems()
```

Returns the number of items in this Choice.
See Also:
getItem

getItem

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

Returns the String at the specified index in the Choice.
Parameters:
index – the index at which to begin
See Also:
countItems

addItem

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

Adds an item to this Choice.
Parameters:
item – the item to be added
Throws: NullPointerException
If the item's value is equal to null.

getSelectedItem

```
public String getSelectedItem()
```

Returns a String representation of the current choice.
See Also:
getSelectedIndex

getSelectedIndex

```
public int getSelectedIndex()
```

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

select

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

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

select

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

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

paramString

```
protected String paramString()
```

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

Class java.awt.Color

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

---

public final class **Color**
extends Object

A class to encapsulate RGB Colors.

Variable Index

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

Constructor Index

- **Color**(int, int, int)
  Creates a color with the specified red, green, and blue values in the range (0 – 255).
- **Color**(int)
  Creates a color with the specified combined RGB value consisting of the red component in bits 16–23, the green component in bits 8–15, and the blue component in bits 0–7.
- **Color**(float, float, float)
  Creates a color with the specified red, green, and blue values in the range (0.0 – 1.0).

Method Index

- **HSBtoRGB**(float, float, float)
  Returns the RGB value defined by the default RGB ColorModel, of the color corresponding to the given HSB color components.
- **RGBtoHSB**(int, int, int, float[])
  Returns the HSB values corresponding to the color defined by the red, green, and blue components.
- **brighter**()
  Returns a brighter version of this color.
- **darker**()
  Returns a darker version of this color.
- **equals**(Object)
  Compares this object against the specified object.
- **getBlue**()
  Gets the blue component.
- **getColor**(String)
  Gets the specified Color property.
- **getColor**(String, Color)
  Gets the specified Color property of the specified Color.
- **getColor**(String, int)
  Gets the specified Color property of the color value.
- **getGreen**()
  Gets the green component.
- **getHSBColor**(float, float, float)
  A static Color factory for generating a Color object from HSB values.
- **getRGB**()
  Gets the RGB value representing the color in the default RGB ColorModel.
- **getRed**()
  Gets the red component.
- **hashCode**()
  Computes the hash code.

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

variables

white

```
public final static Color white
```

The color white.

lightGray

```
public final static Color lightGray
```

The color light gray.

gray

```
public final static Color gray
```

The color gray.

darkGray

```
public final static Color darkGray
```

The color dark gray.

black

```
public final static Color black
```

The color black.

red

```
public final static Color red
```

The color red.

pink

```
public final static Color pink
```

The color pink.

orange

```
public final static Color orange
```

The color orange.

yellow

```
public final static Color yellow
```

The color yellow.

green

```
public final static Color green
```

The color green.

magenta

```
public final static Color magenta
```

The color magneta.

cyan

```
public final static Color cyan
```

The color cyan.

blue

```
public final static Color blue
```

The color blue.

Constructors

Color

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

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

Parameters:

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

See Also:

getRed, getGreen, getBlue, getRGB

Color

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

Creates a color with the specified combined RGB value consisting of the red component in bits 16–23, the green component in bits 8–15, and the blue component in bits 0–7. The actual color used in rendering will depend on finding the best match given the color space available for a given output device.

Parameters:

rgb – the combined RGB components

See Also:

getRGBdefault, getRed, getGreen, getBlue, getRGB

Color

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

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

Parameters:

r – the red component
g – the red component
b – the red component

See Also:

getRed, getGreen, getBlue, getRGB

Methods

getRed

```
public int getRed()
```

Gets the red component.

See Also:

getRGB

getGreen

```
public int getGreen()
```

Gets the green component.
See Also:
getRGB

getBlue

```
public int getBlue()
```

Gets the blue component.
See Also:
getRGB

getRGB

```
public int getRGB()
```

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

brighter

```
public Color brighter()
```

Returns a brighter version of this color.

darker

```
public Color darker()
```

Returns a darker version of this color.

hashCode

```
public int hashCode()
```

Computes the hash code.
Overrides:
hashCode in class Object

equals

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

Compares this object against the specified object.

Parameters:

obj – the object to compare with.

Returns:

true if the objects are the same; false otherwise.

Overrides:

equals in class Object

toString

```
public String toString()
```

Returns the String representation of this Color's values.

Overrides:

toString in class Object

getColor

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

Gets the specified Color property.

Parameters:

nm – the name of the color property

getColor

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

Gets the specified Color property of the specified Color.

Parameters:

nm – the name of the color property

v – the specified color

Returns:

the new color.

getColor

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

Gets the specified Color property of the color value.

Parameters:

nm – the name of the color property

v – the color value

Returns:

the new color.

HSBtoRGB

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

Returns the RGB value defined by the default RGB ColorModel, of the color corresponding to the given HSB color components.

Parameters:

hue – the hue component of the color
saturation – the saturation of the color
brightness – the brightness of the color

See Also:

getRGBdefault, getRGB

RGBtoHSB

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

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

Parameters:

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

Returns:

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

See Also:

getRGBdefault, getRGB

getHSBColor

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

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

Parameters:

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

Returns:

the Color object for the corresponding RGB color

Class java.awt.Component

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

---

public class **Component**
extends Object
implements ImageObserver

A generic Abstract Window Toolkit component.

Method Index

- **action**(Event, Object)
  Called if an action occurs in the Component.
- **addNotify**()
  Notifies the Component to create a peer.
- **bounds**()
  Returns the current bounds of this component.
- **checkImage**(Image, ImageObserver)
  Returns the status of the construction of a screen representation of the specified image.
- **checkImage**(Image, int, int, ImageObserver)
  Returns the status of the construction of a scaled screen representation of the specified image.
- **createImage**(ImageProducer)
  Creates an image from the specified image producer.
- **createImage**(int, int)
  Creates an off–screen drawable Image to be used for double buffering.
- **deliverEvent**(Event)
  Delivers an event to this component or one of its sub components.
- **disable**()
  Disables a component.
- **enable**()
  Enables a component.
- **enable**(boolean)
  Conditionally enables a component.
- **getBackground**()
  Gets the background color.

- **getColorModel**()
  Gets the ColorModel used to display the component on the output device.
- **getFont**()
  Gets the font of the component.
- **getFontMetrics**(Font)
  Gets the font metrics for this component.
- **getForeground**()
  Gets the foreground color.
- **getGraphics**()
  Gets a Graphics context for this component.
- **getParent**()
  Gets the parent of the component.
- **getPeer**()
  Gets the peer of the component.
- **getToolkit**()
  Gets the toolkit of the component.
- **gotFocus**(Event, Object)
  Indicates that this component has received the input focus.
- **handleEvent**(Event)
  Handles the event.
- **hide**()
  Hides the component.
- **imageUpdate**(Image, int, int, int, int, int)
  Repaints the component when the image has changed.
- **inside**(int, int)
  Checks whether a specified x,y location is "inside" this Component.
- **invalidate**()
  Invalidates a component.
- **isEnabled**()
  Checks if this Component is enabled.
- **isShowing**()
  Checks if this Component is showing on screen.
- **isValid**()
  Checks if this Component is valid.
- **isVisible**()
  Checks if this Component is visible.
- **keyDown**(Event, int)
  Called if a character is pressed.
- **keyUp**(Event, int)
  Called if a character is released.
- **layout**()
  Lays out the component.
- **list**()
  Prints a listing to a print stream.
- **list**(PrintStream)
  Prints a listing to the specified print out stream.
- **list**(PrintStream, int)
  Prints out a list, starting at the specified indention, to the specified print stream.
- **locate**(int, int)

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

- **location**()
  Returns the current location of this component.
- **lostFocus**(Event, Object)
  Indicates that this component has lost the input focus.
- **minimumSize**()
  Returns the minimum size of this component.
- **mouseDown**(Event, int, int)
  Called if the mouse is down.
- **mouseDrag**(Event, int, int)
  Called if the mouse is dragged (the mouse button is down).
- **mouseEnter**(Event, int, int)
  Called when the mouse enters the component.
- **mouseExit**(Event, int, int)
  Called when the mouse exits the component.
- **mouseMove**(Event, int, int)
  Called if the mouse moves (the mouse button is up).
- **mouseUp**(Event, int, int)
  Called if the mouse is up.
- **move**(int, int)
  Moves the Component to a new location.
- **nextFocus**()
  Moves the focus to the next component.
- **paint**(Graphics)
  Paints the component.
- **paintAll**(Graphics)
  Paints the component and its subcomponents.
- **paramString**()
  Returns the parameter String of this Component.
- **postEvent**(Event)
  Posts an event to this component.
- **preferredSize**()
  Returns the preferred size of this component.
- **prepareImage**(Image, ImageObserver)
  Prepares an image for rendering on this Component.
- **prepareImage**(Image, int, int, ImageObserver)
  Prepares an image for rendering on this Component at the specified width and height.
- **print**(Graphics)
  Prints this component.
- **printAll**(Graphics)
  Prints the component and its subcomponents.
- **removeNotify**()
  Notifies the Component to destroy the peer.
- **repaint**()
  Repaints the component.
- **repaint**(long)
  Repaints the component.
- **repaint**(int, int, int, int)

Repaints part of the component.

- **repaint**(long, int, int, int, int)
  Repaints part of the component.
- **requestFocus**()
  Requests the input focus.
- **reshape**(int, int, int, int)
  Reshapes the Component to the specified bounding box.
- **resize**(int, int)
  Resizes the Component to the specified width and height.
- **resize**(Dimension)
  Resizes the Component to the specified dimension.
- **setBackground**(Color)
  Sets the background color.
- **setFont**(Font)
  Sets the font of the component.
- **setForeground**(Color)
  Sets the foreground color.
- **show**()
  Shows the component.
- **show**(boolean)
  Conditionally shows the component.
- **size**()
  Returns the current size of this component.
- **toString**()
  Returns the String representation of this Component's values.
- **update**(Graphics)
  Updates the component.
- **validate**()
  Validates a component.

getParent

```
public Container getParent()
```

Gets the parent of the component.

getPeer

```
public ComponentPeer getPeer()
```

Gets the peer of the component.

getToolkit

```
public Toolkit getToolkit()
```

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

isValid

```
public boolean isValid()
```

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

See Also:

validate, invalidate

isVisible

```
public boolean isVisible()
```

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

See Also:

show, hide

isShowing

```
public boolean isShowing()
```

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

See Also:

show, hide

isEnabled

```
public boolean isEnabled()
```

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

See Also:

enable, disable

location

```
public Point location()
```

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

See Also:

move

size

```
public Dimension size()
```

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

bounds

```
public Rectangle bounds()
```

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

enable

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

Enables a component.
See Also:
isEnabled, disable

enable

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

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

disable

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

Disables a component.
See Also:
isEnabled, enable

show

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

Shows the component.
See Also:
isVisible, hide

show

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

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

hide

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

Hides the component.
See Also:
isVisible, hide

getForeground

```
public Color getForeground()
```

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

setForeground

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

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

getBackground

```
public Color getBackground()
```

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

setBackground

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

Sets the background color.
Parameters:
c – the Color
See Also:
getBackground

getFont

```
public Font getFont()
```

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

setFont

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

Sets the font of the component.
Parameters:
f – the font
See Also:
getFont

getColorModel

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

Gets the ColorModel used to display the component on the output device.
See Also:
ColorModel

move

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

Moves the Component to a new location. The x and y coordinates are in the parent's coordinate space.
Parameters:
x – the x coordinate
y – the y coordinate
See Also:
location, reshape

resize

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

Resizes the Component to the specified width and height.

Parameters:

width – the width of the component
height – the height of the component

See Also:

size, reshape

resize

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

Resizes the Component to the specified dimension.

Parameters:

d – the component dimension

See Also:

size, reshape

reshape

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

Reshapes the Component to the specified bounding box.

Parameters:

x – the x coordinate
y – the y coordinate
width – the width of the component
height – the height of the component

See Also:

bounds, move, resize

preferredSize

```
public Dimension preferredSize()
```

Returns the preferred size of this component.

See Also:

minimumSize, LayoutManager

minimumSize

```
public Dimension minimumSize()
```

Returns the minimum size of this component.

See Also:
preferredSize, LayoutManager

layout

```
public void layout()
```

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

validate

```
public void validate()
```

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

invalidate

```
public void invalidate()
```

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

getGraphics

```
public Graphics getGraphics()
```

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

getFontMetrics

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

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

paint

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

Paints the component.
Parameters:
g – the specified Graphics window
See Also:
update

update

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

Updates the component. This method is called in response to a call to repaint. You can assume that the background is not cleared.
Parameters:
g – the specified Graphics window
See Also:
paint, repaint

paintAll

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

Paints the component and its subcomponents.
Parameters:
g – the specified Graphics window
See Also:
paint

repaint

```
public void repaint()
```

Repaints the component. This will result in a call to update as soon as possible.
See Also:
paint

repaint

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

Repaints the component. This will result in a call to update within *tm* milliseconds.
Parameters:
tm – maximum time in milliseconds before update
See Also:
paint

repaint

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

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

Parameters:

x – the x coordinate
y – the y coordinate
width – the width
height – the height

See Also:

repaint

repaint

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

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

Parameters:

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

See Also:

repaint

print

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

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

Parameters:

g – the specified Graphics window

See Also:

paint

printAll

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

Prints the component and its subcomponents.

Parameters:

g – the specified Graphics window

See Also:

print

imageUpdate

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

Repaints the component when the image has changed.

Returns:

true if image has changed; false otherwise.

createImage

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

Creates an image from the specified image producer.

Parameters:

producer – the image producer

createImage

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

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

Parameters:

width – the specified width

height – the specified height

prepareImage

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

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

Parameters:

image – the Image to prepare a screen representation for

observer – the ImageObserver object to be notified as the image is being prepared

Returns:

true if the image has already been fully prepared

See Also:

ImageObserver

prepareImage

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

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

Parameters:

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

Returns:

true if the image has already been fully prepared

See Also:

ImageObserver

checkImage

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

Returns the status of the construction of a screen representation of the specified image. This method does not cause the image to begin loading. Use the prepareImage method to force the loading of an image.

Parameters:

image – the Image to check the status of
observer – the ImageObserver object to be notified as the image is being prepared

Returns:

the boolean OR of the ImageObserver flags for the data that is currently available

See Also:

ImageObserver, prepareImage

checkImage

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

Returns the status of the construction of a scaled screen representation of the specified image. This method does not cause the image to begin loading, use the prepareImage method to force the loading of an image.

Parameters:

image – the Image to check the status of
width – the width of the scaled version to check the status of
height – the height of the scaled version to check the status of
observer – the ImageObserver object to be notified as the image is being prepared

Returns:

the boolean OR of the ImageObserver flags for the data that is currently available

See Also:

ImageObserver, prepareImage

inside

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

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

Parameters:

x – the x coordinate
y – the y coordinate

See Also:

locate

locate

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

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

Parameters:

x – the x coordinate
y – the y coordinate

See Also:

inside

deliverEvent

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

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

Parameters:

e – the event

See Also:

handleEvent, postEvent

postEvent

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

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

Parameters:

e – the event

See Also:

handleEvent, deliverEvent

handleEvent

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

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

Parameters:

evt – the event

See Also:

mouseEnter, mouseExit, mouseMove, mouseDown, mouseDrag, mouseUp, keyDown, action

mouseDown

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

Called if the mouse is down.

Parameters:

evt – the event
x – the x coordinate
y – the y coordinate

See Also:

handleEvent

mouseDrag

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

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

Parameters:

evt – the event

x – the x coordinate
y – the y coordinate

See Also:

handleEvent

mouseUp

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

Called if the mouse is up.

Parameters:

evt – the event
x – the x coordinate
y – the y coordinate

See Also:

handleEvent

mouseMove

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

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

Parameters:

evt – the event
x – the x coordinate
y – the y coordinate

See Also:

handleEvent

mouseEnter

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

Called when the mouse enters the component.

Parameters:

evt – the event
x – the x coordinate
y – the y coordinate

See Also:

handleEvent

mouseExit

```
public boolean mouseExit(Event evt,
```

```
                        int x,
                        int y)
```

Called when the mouse exits the component.

Parameters:

evt – the event
x – the x coordinate
y – the y coordinate

See Also:

handleEvent

keyDown

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

Called if a character is pressed.

Parameters:

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

See Also:

handleEvent

keyUp

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

Called if a character is released.

Parameters:

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

See Also:

handleEvent

action

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

Called if an action occurs in the Component.

Parameters:

evt – the event
what – the action that's occuring

See Also:

handleEvent

addNotify

```
public void addNotify()
```

Notifies the Component to create a peer.
See Also:
getPeer, removeNotify

removeNotify

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

Notifies the Component to destroy the peer.
See Also:
getPeer, addNotify

gotFocus

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

Indicates that this component has received the input focus.
See Also:
requestFocus, lostFocus

lostFocus

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

Indicates that this component has lost the input focus.
See Also:
requestFocus, gotFocus

requestFocus

```
public void requestFocus()
```

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

nextFocus

```
public void nextFocus()
```

Moves the focus to the next component.
See Also:
requestFocus, gotFocus

paramString

```
protected String paramString()
```

Returns the parameter String of this Component.

toString

```
public String toString()
```

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

list

```
public void list()
```

Prints a listing to a print stream.

list

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

Prints a listing to the specified print out stream.
Parameters:
out – the Stream name

list

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

Prints out a list, starting at the specified indention, to the specified print stream.
Parameters:
out – the Stream name
indent – the start of the list

Class java.awt.Container

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

---

public class **Container**
extends Component

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

Method Index

- **add**(Component)
  Adds the specified component to this container.
- **add**(Component, int)
  Adds the specified component to this container at the given position.
- **add**(String, Component)
  Adds the specified component to this container.
- **addNotify**()
  Notifies the container to create a peer.
- **countComponents**()
  Returns the number of components in this panel.
- **deliverEvent**(Event)
  Delivers an event.
- **getComponent**(int)
  Gets the nth component in this container.
- **getComponents**()
  Gets all the components in this container.
- **getLayout**()
  Gets the layout manager for this container.
- **insets**()
  Returns the insets of the container.
- **layout**()
  Does a layout on this Container.
- **list**(PrintStream, int)
  Prints out a list, starting at the specified indention, to the specified out stream.

- **locate**(int, int)
  Locates the component that contains the x,y position.
- **minimumSize**()
  Returns the minimum size of this container.
- **paintComponents**(Graphics)
  Paints the components in this container.
- **paramString**()
  Returns the parameter String of this Container.
- **preferredSize**()
  Returns the preferred size of this container.
- **printComponents**(Graphics)
  Prints the components in this container.
- **remove**(Component)
  Removes the specified component from this container.
- **removeAll**()
  Removes all the components from this container.
- **removeNotify**()
  Notifies the container to remove its peer.
- **setLayout**(LayoutManager)
  Sets the layout manager for this container.
- **validate**()
  Validates this Container and all of the components contained within it.

countComponents

```
public int countComponents()
```

Returns the number of components in this panel.

See Also:

getComponent

getComponent

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

Gets the nth component in this container.

Parameters:

n – the number of the component to get

Throws: ArrayIndexOutOfBoundsException

If the nth value does not exist.

getComponents

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

Gets all the components in this container.

insets

```
public Insets insets()
```

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

See Also:

LayoutManager

add

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

Adds the specified component to this container.

Parameters:

comp – the component to be added

add

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

Adds the specified component to this container at the given position.

Parameters:

comp – the component to be added

pos – the position at which to insert the component. –1 means insert at the end.

See Also:

remove

add

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

Adds the specified component to this container. The component is also added to the layout manager of this container using the name specified .

Parameters:

name – the component name

comp – the component to be added

See Also:

remove, LayoutManager

remove

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

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

removeAll

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

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

getLayout

```
public LayoutManager getLayout()
```

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

setLayout

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

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

layout

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

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

validate

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

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

validate in class Component

See Also:

validate, invalidate

preferredSize

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

Returns the preferred size of this container.

Overrides:

preferredSize in class Component

See Also:

minimumSize

minimumSize

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

Returns the minimum size of this container.

Overrides:

minimumSize in class Component

See Also:

preferredSize

paintComponents

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

Paints the components in this container.

Parameters:

g – the specified Graphics window

See Also:

paint, paintAll

printComponents

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

Prints the components in this container.

Parameters:

g – the specified Graphics window

See Also:

print, printAll

deliverEvent

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

Delivers an event. The appropriate component is located and the event is delivered

to it.

Parameters:

e – the event

Overrides:

deliverEvent in class Component

See Also:

handleEvent, postEvent

locate

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

Locates the component that contains the x,y position.

Parameters:

x – the x coordinate

y – the y coordinate

Returns:

null if the component is not within the x and y coordinates; returns the component otherwise.

Overrides:

locate in class Component

See Also:

inside

addNotify

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

Notifies the container to create a peer. It will also notify the components contained in this container.

Overrides:

addNotify in class Component

See Also:

removeNotify

removeNotify

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

Notifies the container to remove its peer. It will also notify the components contained in this container.

Overrides:

removeNotify in class Component

See Also:

addNotify

paramString

```
protected String paramString()
```

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

list

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

Prints out a list, starting at the specified indention, to the specified out stream.
Parameters:
out – the Stream name
indent – the start of the list
Overrides:
list in class Component

Class java.awt.Dialog

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

---

public class **Dialog**
extends Window

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

Constructor Index

- **Dialog**(Frame, boolean)
  Constructs an initially invisible Dialog.
- **Dialog**(Frame, String, boolean)
  Constructs an initially invisible Dialog with a title.

Method Index

- **addNotify**()
  Creates the frame's peer.
- **getTitle**()
  Gets the title of the Dialog.
- **isModal**()
  Returns true if the Dialog is modal.
- **isResizable**()
  Returns true if the user can resize the frame.
- **paramString**()
  Returns the parameter String of this Dialog.
- **setResizable**(boolean)
  Sets the resizable flag.

- **setTitle**(String)
  Sets the title of the Dialog.

Constructors

Dialog

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

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

Parameters:

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

See Also:

resize, show

Dialog

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

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

Parameters:

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

See Also:

resize, show

Methods

addNotify

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

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

Overrides:

addNotify in class Window

isModal

```
public boolean isModal()
```

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

getTitle

```
public String getTitle()
```

Gets the title of the Dialog.
See Also:
setTitle

setTitle

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

Sets the title of the Dialog.
Parameters:
title – the new title being given to the Dialog
See Also:
getTitle

isResizable

```
public boolean isResizable()
```

Returns true if the user can resize the frame.

setResizable

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

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

paramString

```
protected String paramString()
```

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

Class java.awt.Dimension

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

---

public class **Dimension**
extends Object

A class to encapsulate a width and a height Dimension.

Variable Index

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

Constructor Index

- **Dimension**()
  Constructs a Dimension with a 0 width and 0 height.
- **Dimension**(Dimension)
  Constructs a Dimension and initializes it to the specified value.
- **Dimension**(int, int)
  Constructs a Dimension and initializes it to the specified width and specified height.

Method Index

- **toString**()
  Returns the String representation of this Dimension's values.

Variables

- **width**

```
public int width
```

The width dimension.

- **height**

```
public int height
```

The height dimension.

Constructors

- **Dimension**

```
public Dimension()
```

Constructs a Dimension with a 0 width and 0 height.

- **Dimension**

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

Constructs a Dimension and initializes it to the specified value.
Parameters:
d – the specified dimension for the width and height values

- **Dimension**

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

Constructs a Dimension and initializes it to the specified width and specified height.
Parameters:
width – the specified width dimension
height – the specified height dimension

Methods

toString

```
public String toString()
```

Returns the String representation of this Dimension's values.

Overrides:

toString in class Object

Class java.awt.Event

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

---

public class **Event**
extends Object

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

Variable Index

- **ACTION_EVENT**
  An action event.
- **ALT_MASK**
  The alt modifier constant.
- **CTRL_MASK**
  The control modifier constant.
- **DOWN**
  The down arrow key.
- **END**
  The end key.
- **F1**
  The F1 function key.
- **F10**
  The F10 function key.
- **F11**
  The F11 function key.
- **F12**
  The F12 function key.
- **F2**
  The F2 function key.
- **F3**
  The F3 function key.
- **F4**
  The F4 function key.
- **F5**
  The F5 function key.

- **F6**
  The F6 function key.
- **F7**
  The F7 function key.
- **F8**
  The F8 function key.
- **F9**
  The F9 function key.
- **GOT_FOCUS**
  A component gained the focus.
- **HOME**
  The home key.
- **KEY_ACTION**
  The key action keyboard event.
- **KEY_ACTION_RELEASE**
  The key action keyboard event.
- **KEY_PRESS**
  The key press keyboard event.
- **KEY_RELEASE**
  The key release keyboard event.
- **LEFT**
  The left arrow key.
- **LIST_DESELECT**
- **LIST_SELECT**
- **LOAD_FILE**
  A file loading event.
- **LOST_FOCUS**
  A component lost the focus.
- **META_MASK**
  The meta modifier constant.
- **MOUSE_DOWN**
  The mouse down event.
- **MOUSE_DRAG**
  The mouse drag event.
- **MOUSE_ENTER**
  The mouse enter event.
- **MOUSE_EXIT**
  The mouse exit event.
- **MOUSE_MOVE**
  The mouse move event.
- **MOUSE_UP**
  The mouse up event.
- **PGDN**
  The page down key.
- **PGUP**
  The page up key.
- **RIGHT**
  The right arrow key.
- **SAVE_FILE**

A file saving event.

- **SCROLL_ABSOLUTE**
  The absolute scroll event.
- **SCROLL_LINE_DOWN**
  The line down scroll event.
- **SCROLL_LINE_UP**
  The line up scroll event.
- **SCROLL_PAGE_DOWN**
  The page down scroll event.
- **SCROLL_PAGE_UP**
  The page up scroll event.
- **SHIFT_MASK**
  The shift modifier constant.
- **UP**
  The up arrow key.
- **WINDOW_DEICONIFY**
  The de–iconify window event.
- **WINDOW_DESTROY**
  The destroy window event.
- **WINDOW_EXPOSE**
  The expose window event.
- **WINDOW_ICONIFY**
  The iconify window event.
- **WINDOW_MOVED**
  The move window event.
- **arg**
  An arbitraty argument.
- **clickCount**
  The number of consecutive clicks.
- **evt**
  The next event.
- **id**
  The type of this event.
- **key**
  The key that was pressed in a keyboard event.
- **modifiers**
  The state of the modifier keys.
- **target**
  The target component.
- **when**
  The time stamp.
- **x**
  The x coordinate of the event.
- **y**
  The y coordinate of the event.

Constructor Index

- **Event**(Object, long, int, int, int, int, int, Object)
  Constructs an event with the specified target component, time stamp, event type, x and y coordinates, keyboard key, state of the modifier keys and argument.
- **Event**(Object, long, int, int, int, int, int)
  Constructs an event with the specified target component, time stamp, event type, x and y coordinates, keyboard key, state of the modifier keys and an argument set to null.
- **Event**(Object, int, Object)
  Constructs an event with the specified target component, event type, and argument.

Method Index

- **controlDown**()
  Checks if the control key is down.
- **metaDown**()
  Checks if the meta key is down.
- **paramString**()
  Returns the parameter String of this Event.
- **shiftDown**()
  Checks if the shift key is down.
- **toString**()
  Returns the String representation of this Event's values.
- **translate**(int, int)
  Translates an event relative to the given component.

Variables

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

```
public Object target
```

The target component.

- **when**

```
public long when
```

The time stamp.

- **id**

```
public int id
```

The type of this event.

- **x**

```
public int x
```

The x coordinate of the event.

- **y**

```
public int y
```

The y coordinate of the event.

- **key**

```
public int key
```

The key that was pressed in a keyboard event.

- **modifiers**

```
public int modifiers
```

The state of the modifier keys.

- **clickCount**

```
public int clickCount
```

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

- **arg**

```
public Object arg
```

An arbitraty argument.

evt

```
public Event evt
```

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

Constructors

Event

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

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

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

Event

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

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

Parameters:

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

Event

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

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

Parameters:

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

Methods

translate

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

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

Parameters:

x – the x coordinate
y – the y coordinate

shiftDown

```
public boolean shiftDown()
```

Checks if the shift key is down.

See Also:

modifiers, controlDown, metaDown

controlDown

```
public boolean controlDown()
```

Checks if the control key is down.
See Also:
modifiers, shiftDown, metaDown

metaDown

```
public boolean metaDown()
```

Checks if the meta key is down.
See Also:
modifiers, shiftDown, controlDown

paramString

```
protected String paramString()
```

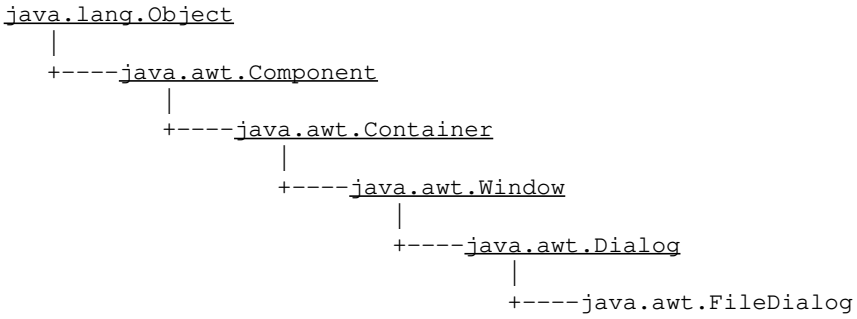
Returns the parameter String of this Event.

toString

```
public String toString()
```

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

Class java.awt.FileDialog



---

public class **FileDialog**
extends Dialog

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

See Also:
show

Variable Index

- **LOAD**
  The file load variable.
- **SAVE**
  The file save variable.

Constructor Index

- **FileDialog**(Frame, String)
  Creates a file dialog for loading a file.
- **FileDialog**(Frame, String, int)
  Creates a file dialog with the specified title and mode.

Method Index

- **addNotify**()
  Creates the frame's peer.
- **getDirectory**()
  Gets the directory of the Dialog.
- **getFile**()
  Gets the file of the Dialog.
- **getFilenameFilter**()
  Gets the filter.
- **getMode**()
  Gets the mode of the file dialog.
- **paramString**()
  Returns the parameter String of this file dialog.
- **setDirectory**(String)
  Set the directory of the Dialog to the specified directory.
- **setFile**(String)
  Sets the file for this dialog to the specified file.
- **setFilenameFilter**(FilenameFilter)
  Sets the filter for this dialog to the specified filter.

Variables

LOAD

```
public final static int LOAD
```

The file load variable.

SAVE

```
public final static int SAVE
```

The file save variable.

Constructors

FileDialog

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

Creates a file dialog for loading a file.
Parameters:

parent – the owner of the dialog
title – the title of the Dialog

FileDialog

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

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

Methods

addNotify

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

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

getMode

```
public int getMode()
```

Gets the mode of the file dialog.

getDirectory

```
public String getDirectory()
```

Gets the directory of the Dialog.

setDirectory

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

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

getFile

```
public String getFile()
```

Gets the file of the Dialog.

setFile

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

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

Parameters:

file – the file being set

getFilenameFilter

```
public FilenameFilter getFilenameFilter()
```

Gets the filter.

setFilenameFilter

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

Sets the filter for this dialog to the specified filter.

Parameters:

filter – the specified filter

paramString

```
protected String paramString()
```

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

Overrides:

paramString in class Dialog

Class java.awt.FlowLayout

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

---

public class **FlowLayout**
extends Object
implements LayoutManager

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

Variable Index

- **CENTER**
  The right alignment variable.
- **LEFT**
  The left alignment variable.
- **RIGHT**
  The right alignment variable.

Constructor Index

- **FlowLayout**()
  Constructs a new Flow Layout with a centered alignment.
- **FlowLayout**(int)
  Constructs a new Flow Layout with the specified alignment.
- **FlowLayout**(int, int, int)
  Constructs a new Flow Layout with the specified alignment and gap values.

Method Index

- **addLayoutComponent**(String, Component)
  Adds the specified component to the layout.
- **layoutContainer**(Container)

Lays out the container.

- **minimumLayoutSize**(Container)
  Returns the minimum dimensions needed to layout the components contained in the specified target container.
- **preferredLayoutSize**(Container)
  Returns the preferred dimensions for this layout given the components in the specified target container.
- **removeLayoutComponent**(Component)
  Removes the specified component from the layout.
- **toString**()
  Returns the String representation of this FlowLayout's values.

Variables

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.

Constructors

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

Methods

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
```

---

public class **Font**
extends Object

A class that produces font objects.

Variable Index

- **BOLD**
  The bold style constant.
- **ITALIC**
  The italicized style constant.
- **PLAIN**
  The plain style constant.
- **name**
  The logical name of this font.
- **size**
  The point size of this font.
- **style**
  The style of the font.

Constructor Index

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

Method Index

- **equals**(Object)
  Compares this object to the specifed object.
- **getFamily**()

Gets the platform specific family name of the font.

- **getFont**(String)
  Gets a font from the system properties list.
- **getFont**(String, Font)
  Gets the specified font from the system properties list.
- **getName**()
  Gets the logical name of the font.
- **getSize**()
  Gets the point size of the font.
- **getStyle**()
  Gets the style of the font.
- **hashCode**()
  Returns a hashcode for this font.
- **isBold**()
  Returns true if the font is bold.
- **isItalic**()
  Returns true if the font is italic.
- **isPlain**()
  Returns true if the font is plain.
- **toString**()
  Converts this object to a String representation.

PLAIN

```
public final static int PLAIN
```

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

BOLD

```
public final static int BOLD
```

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

ITALIC

```
public final static int ITALIC
```

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

name

```
protected String name
```

The logical name of this font.

style

```
protected int style
```

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

size

```
protected int size
```

The point size of the font.

Constructors

Font

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

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

Parameters:

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

See Also:

getFontList

Methods

getFamily

```
public String getFamily()
```

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

See Also:

getName

getName

```
public String getName()
```

Gets the logical name of the font.
See Also:
getFamily

getStyle

```
public int getStyle()
```

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

getSize

```
public int getSize()
```

Gets the point size of the font.

isPlain

```
public boolean isPlain()
```

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

isBold

```
public boolean isBold()
```

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

isItalic

```
public boolean isItalic()
```

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

getFont

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

Gets a font from the system properties list.

Parameters:
nm – the property name

getFont

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

Gets the specified font from the system properties list.
Parameters:
nm – the property name
font – a default font to return if property 'nm' is not defined

hashCode

```
public int hashCode()
```

Returns a hashcode for this font.
Overrides:
hashCode in class Object

equals

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

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

toString

```
public String toString()
```

Converts this object to a String representation.
Overrides:
toString in class Object

Class java.awt.FontMetrics

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

---

public class **FontMetrics**
extends Object

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

Variable Index

- **font**
  The actual font.

Constructor Index

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

Method Index

- **bytesWidth**(byte[], int, int)
  Returns the width of the specified array of bytes in this Font.
- **charWidth**(int)
  Returns the width of the specified character in this Font.
- **charWidth**(char)
  Returns the width of the specified character in this Font.
- **charsWidth**(char[], int, int)
  Returns the width of the specified character array in this Font.
- **getAscent**()
  Gets the font ascent.
- **getDescent**()
  Gets the font descent.

- **getFont**()
  Gets the font.
- **getHeight**()
  Gets the total height of the font.
- **getLeading**()
  Gets the standard leading, or line spacing, for the font.
- **getMaxAdvance**()
  Gets the maximum advance width of any character in this Font.
- **getMaxAscent**()
  Gets the maximum ascent of all characters in this Font.
- **getMaxDecent**()
  For backward compatibility only.
- **getMaxDescent**()
  Gets the maximum descent of all characters.
- **getWidths**()
  Gets the widths of the first 256 characters in the Font.
- **stringWidth**(String)
  Returns the width of the specified String in this Font.
- **toString**()
  Returns the String representation of this FontMetric's values.

Variables

font

```
protected Font font
```

The actual font.
See Also:
getFont

Constructors

FontMetrics

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

Creates a new FontMetrics object with the specified font.
Parameters:
font – the font
See Also:
Font

Methods

getFont

```
public Font getFont()
```

Gets the font.

getLeading

```
public int getLeading()
```

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

getAscent

```
public int getAscent()
```

Gets the font ascent. The font ascent is the distance from the base line to the top of the characters.
See Also:
getMaxAscent

getDescent

```
public int getDescent()
```

Gets the font descent. The font descent is the distance from the base line to the bottom of the characters.
See Also:
getMaxDescent

getHeight

```
public int getHeight()
```

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

getMaxAscent

```
public int getMaxAscent()
```

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

See Also:
getAscent

getMaxDescent

```
public int getMaxDescent()
```

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

getMaxDecent

```
public int getMaxDecent()
```

For backward compatibility only.
See Also:
getMaxDescent

getMaxAdvance

```
public int getMaxAdvance()
```

Gets the maximum advance width of any character in this Font.
Returns:
–1 if the max advance is not known.

charWidth

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

Returns the width of the specified character in this Font.
Parameters:
ch – the specified font
See Also:
stringWidth

charWidth

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

Returns the width of the specified character in this Font.
Parameters:
ch – the specified font
See Also:
stringWidth

stringWidth

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

Returns the width of the specified String in this Font.

Parameters:

str – the String to be checked

See Also:

charsWidth, bytesWidth

charsWidth

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

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

Parameters:

data – the data to be checked
off – the start offset of the data
len – the maximum number of bytes checked

See Also:

stringWidth, bytesWidth

bytesWidth

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

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

Parameters:

data – the data to be checked
off – the start offset of the data
len – the maximum number of bytes checked

See Also:

stringWidth, charsWidth

getWidths

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

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

toString

```
public String toString()
```

Returns the String representation of this FontMetric's values.

Overrides:

toString in class Object

Class java.awt.Frame

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

---

public class **Frame**
extends Window
implements MenuContainer

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

Variable Index

- **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**

Constructor Index

- **Frame**()

Constructs a new Frame that is initially invisible.

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

Method Index

- **addNotify**()
  Creates the Frame's peer.
- **dispose**()
  Disposes of the Frame.
- **getCursorType**()
  Return the cursor type
- **getIconImage**()
  Returns the icon image for this Frame.
- **getMenuBar**()
  Gets the menu bar for this Frame.
- **getTitle**()
  Gets the title of the Frame.
- **isResizable**()
  Returns true if the user can resize the Frame.
- **paramString**()
  Returns the parameter String of this Frame.
- **remove**(MenuComponent)
  Removes the specified menu bar from this Frame.
- **setCursor**(int)
  Set the cursor image to a predefined cursor.
- **setIconImage**(Image)
  Sets the image to display when this Frame is iconized.
- **setMenuBar**(MenuBar)
  Sets the menubar for this Frame to the specified menubar.
- **setResizable**(boolean)
  Sets the resizable flag.
- **setTitle**(String)
  Sets the title for this Frame to the specified title.

Variables

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
```

Constructors

Frame

```
public Frame()
```

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

Frame

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

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

Methods

addNotify

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

Creates the Frame's peer. The peer allows us to change the look of the Frame without changing its functionality.
Overrides:
addNotify in class Window

getTitle

```
public String getTitle()
```

Gets the title of the Frame.
See Also:
setTitle

setTitle

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

Sets the title for this Frame to the specified title.
Parameters:
title – the specified title of this Frame

See Also:
getTitle

getIconImage

```
public Image getIconImage()
```

Returns the icon image for this Frame.

setIconImage

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

Sets the image to display when this Frame is iconized. Note that not all platforms support the concept of iconizing a window.
Parameters:
image – the icon image to be displayed

getMenuBar

```
public MenuBar getMenuBar()
```

Gets the menu bar for this Frame.

setMenuBar

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

Sets the menubar for this Frame to the specified menubar.
Parameters:
mb – the menubar being set

remove

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

Removes the specified menu bar from this Frame.

dispose

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

Disposes of the Frame. This method must be called to release the resources that are used for the frame.
Overrides:
dispose in class Window

isResizable

```
public boolean isResizable()
```

Returns true if the user can resize the Frame.

setResizable

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

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

setCursor

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

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

getCursorType

```
public int getCursorType()
```

Return the cursor type

paramString

```
protected String paramString()
```

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

Class java.awt.Graphics

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

---

public class **Graphics**
extends Object

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

Constructor Index

- **Graphics**()
  Constructs a new Graphics Object.

Method Index

- **clearRect**(int, int, int, int)
  Clears the specified rectangle by filling it with the current background color of the current drawing surface.
- **clipRect**(int, int, int, int)
  Clips to a rectangle.
- **copyArea**(int, int, int, int, int, int)
  Copies an area of the screen.
- **create**()
  Creates a new Graphics Object that is a copy of the original Graphics Object.
- **create**(int, int, int, int)
  Creates a new Graphics Object with the specified parameters, based on the original Graphics Object.
- **dispose**()
  Disposes of this graphics context.
- **draw3DRect**(int, int, int, int, boolean)
  Draws a highlighted 3–D rectangle.
- **drawArc**(int, int, int, int, int, int)
  Draws an arc bounded by the specified rectangle from startAngle to endAngle.
- **drawBytes**(byte[], int, int, int, int)
  Draws the specified bytes using the current font and color.

- **drawChars**(char[], int, int, int, int)
  Draws the specified characters using the current font and color.
- **drawImage**(Image, int, int, ImageObserver)
  Draws the specified image at the specified coordinate (x, y).
- **drawImage**(Image, int, int, int, int, ImageObserver)
  Draws the specified image inside the specified rectangle.
- **drawImage**(Image, int, int, Color, ImageObserver)
  Draws the specified image at the specified coordinate (x, y), with the given solid background Color.
- **drawImage**(Image, int, int, int, int, Color, ImageObserver)
  Draws the specified image inside the specified rectangle, with the given solid background Color.
- **drawLine**(int, int, int, int)
  Draws a line between the coordinates (x1,y1) and (x2,y2).
- **drawOval**(int, int, int, int)
  Draws an oval inside the specified rectangle using the current color.
- **drawPolygon**(int[], int[], int)
  Draws a polygon defined by an array of x points and y points.
- **drawPolygon**(Polygon)
  Draws a polygon defined by the specified point.
- **drawRect**(int, int, int, int)
  Draws the outline of the specified rectangle using the current color.
- **drawRoundRect**(int, int, int, int, int, int)
  Draws an outlined rounded corner rectangle using the current color.
- **drawString**(String, int, int)
  Draws the specified String using the current font and color.
- **fill3DRect**(int, int, int, int, boolean)
  Paints a highlighted 3–D rectangle using the current color.
- **fillArc**(int, int, int, int, int, int)
  Fills an arc using the current color.
- **fillOval**(int, int, int, int)
  Fills an oval inside the specified rectangle using the current color.
- **fillPolygon**(int[], int[], int)
  Fills a polygon with the current color using an even–odd fill rule (otherwise known as an alternating rule).
- **fillPolygon**(Polygon)
  Fills the specified polygon with the current color using an even–odd fill rule (otherwise known as an alternating rule).
- **fillRect**(int, int, int, int)
  Fills the specified rectangle with the current color.
- **fillRoundRect**(int, int, int, int, int, int)
  Draws a rounded rectangle filled in with the current color.
- **finalize**()
  Disposes of this graphics context once it is no longer referenced.
- **getClipRect**()
  Returns the bounding rectangle of the current clipping area.
- **getColor**()
  Gets the current color.
- **getFont**()

Gets the current font.

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

Constructors

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

Methods

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
```

---

public class **GridBagConstraints**
extends Object
implements Cloneable

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

See Also:
GridBagLayout

Variable Index

- **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**

Constructor Index

- **GridBagConstraints**()

Method Index

- **clone**()
  Creates a clone of the object.

Variables

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
```

Constructors

GridBagConstraints

```
public GridBagConstraints()
```

Methods

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
```

---

public class **GridBagLayout**
extends Object
implements LayoutManager

GridBagLayout is a flexible layout manager that aligns components vertically and horizontally, without requiring that the components be the same size. Each GridBagLayout uses a rectangular grid of cells, with each component occupying one or more cells (called its *display area*). Each component managed by a GridBagLayout is associated with a GridBagConstraints instance that specifies how the component is laid out within its display area. How a GridBagLayout places a set of components depends on each component's GridBagConstraints and minimum size, as well as the preferred size of the components' container.

To use a GridBagLayout effectively, you must customize one or more of its components' GridBagConstraints. You customize a GridBagConstraints object by setting one or more of its instance variables:

gridx, gridy
: Specifies the cell at the upper left of the component's display area, where the upper–left–most cell has address gridx=0, gridy=0. Use GridBagConstraints.RELATIVE (the default value) to specify that the component be placed just to the right of (for gridx) or just below (for gridy) the component that was added to the container just before this component was added.

gridwidth, gridheight
: Specifies the number of cells in a row (for gridwidth) or column (for gridheight) in the component's display area. The default value is 1. Use GridBagConstraints.REMAINDER to specify that the component be the last one in its row (for gridwidth) or column (for gridheight). Use GridBagConstraints.RELATIVE to specify that the component be the next to last one in its row (for gridwidth) or column (for gridheight).

fill
: Used when the component's display area is larger than the component's requested size to determine whether (and how) to resize the component. Valid values are GridBagConstraint.NONE (the default), GridBagConstraint.HORIZONTAL (make the component wide enough to fill its display area horizontally, but don't change its height), GridBagConstraint.VERTICAL (make the component tall enough to fill its display area vertically, but don't change its width), and

GridBagConstraint.BOTH (make the component fill its display area entirely).

ipadx, ipady
: Specifies the internal padding: how much to add to the minimum size of the component. The width of the component will be at least its minimum width plus ipadx*2 pixels (since the padding applies to both sides of the component). Similarly, the height of the component will be at least the minimum height plus ipady*2 pixels.

insets
: Specifies the external padding of the component –– the minimum amount of space between the component and the edges of its display area.

anchor
: Used when the component is smaller than its display area to determine where (within the area) to place the component. Valid values are GridBagConstraints.CENTER (the default), GridBagConstraints.NORTH, GridBagConstraints.NORTHEAST, GridBagConstraints.EAST, GridBagConstraints.SOUTHEAST, GridBagConstraints.SOUTH, GridBagConstraints.SOUTHWEST, GridBagConstraints.WEST, and GridBagConstraints.NORTHWEST.

weightx, weighty
: Used to determine how to distribute space; this is important for specifying resizing behavior. Unless you specify a weight for at least one component in a row (weightx) and column (weighty), all the components clump together in the center of their container. This is because when the weight is zero (the default), the GridBagLayout puts any extra space between its grid of cells and the edges of the container.

The following figure shows ten components (all buttons) managed by a GridBagLayout:

All the components have fill=GridBagConstraints.BOTH. In addition, the components have the following non–default constraints:

- Button1, Button2, Button3: weightx=1.0
- Button4: weightx=1.0, gridwidth=GridBagConstraints.REMAINDER
- Button5: gridwidth=GridBagConstraints.REMAINDER
- Button6: gridwidth=GridBagConstraints.RELATIVE
- Button7: gridwidth=GridBagConstraints.REMAINDER
- Button8: gridheight=2, weighty=1.0,
- Button9, Button 10: gridwidth=GridBagConstraints.REMAINDER

Here is the code that implements the example shown above:

```
import java.awt.*;
import java.util.*;
import java.applet.Applet;
public class GridBagEx1 extends Applet {
    protected void makebutton(String name,
```

```
                            GridBagLayout gridbag,
                            GridBagConstraints c) {
        Button button = new Button(name);
        gridbag.setConstraints(button, c);
        add(button);
    }
    public void init() {
        GridBagLayout gridbag = new GridBagLayout();
        GridBagConstraints c = new GridBagConstraints();
        setFont(new Font("Helvetica", Font.PLAIN, 14));
        setLayout(gridbag);
        c.fill = GridBagConstraints.BOTH;
        c.weightx = 1.0;
        makebutton("Button1", gridbag, c);
        makebutton("Button2", gridbag, c);
        makebutton("Button3", gridbag, c);
        c.gridwidth = GridBagConstraints.REMAINDER; //end row
        makebutton("Button4", gridbag, c);
        c.weightx = 0.0;                    //reset to the default
        makebutton("Button5", gridbag, c); //another row
        c.gridwidth = GridBagConstraints.RELATIVE; //next-to-last in row
        makebutton("Button6", gridbag, c);
        c.gridwidth = GridBagConstraints.REMAINDER; //end row
        makebutton("Button7", gridbag, c);
        c.gridwidth = 1;                    //reset to the default
        c.gridheight = 2;
        c.weighty = 1.0;
        makebutton("Button8", gridbag, c);
        c.weighty = 0.0;                    //reset to the default
        c.gridwidth = GridBagConstraints.REMAINDER; //end row
        c.gridheight = 1;                   //reset to the default
        makebutton("Button9", gridbag, c);
        makebutton("Button10", gridbag, c);
        resize(300, 100);
    }
    public static void main(String args[]) {
        Frame f = new Frame("GridBag Layout Example");
        GridBagEx1 ex1 = new GridBagEx1();
        ex1.init();
        f.add("Center", ex1);
        f.pack();
        f.resize(f.preferredSize());
        f.show();
    }
}
```

Variable Index

- **MAXGRIDSIZE**
- **MINSIZE**
- **PREFERREDSIZE**
- **columnWeights**

- **columnWidths**
- **comptable**
- **defaultConstraints**
- **layoutInfo**
- **rowHeights**
- **rowWeights**

Constructor Index

- **GridBagLayout**()
  Creates a gridbag layout.

Method Index

- **AdjustForGravity**(GridBagConstraints, Rectangle)
- **ArrangeGrid**(Container)
- **DumpConstraints**(GridBagConstraints)
  Print the layout constraints.
- **DumpLayoutInfo**(GridBagLayoutInfo)
  Print the layout information.
- **GetLayoutInfo**(Container, int)
- **GetMinSize**(Container, GridBagLayoutInfo)
- **addLayoutComponent**(String, Component)
  Adds the specified component with the specified name to the layout.
- **getConstraints**(Component)
  Retrieves the constraints for the specified component.
- **getLayoutDimensions**()
- **getLayoutOrigin**()
- **getLayoutWeights**()
- **layoutContainer**(Container)
  Lays out the container in the specified panel.
- **location**(int, int)
- **lookupConstraints**(Component)
  Retrieves the constraints for the specified component.
- **minimumLayoutSize**(Container)
  Returns the minimum dimensions needed to layout the components contained in the specified panel.
- **preferredLayoutSize**(Container)
  Returns the preferred dimensions for this layout given the components in the specified panel.
- **removeLayoutComponent**(Component)
  Removes the specified component from the layout.
- **setConstraints**(Component, GridBagConstraints)
  Sets the constraints for the specified component.
- **toString**()

Returns the String representation of this GridLayout's values.

Variables

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[]
```

Constructors

GridBagLayout

```
public GridBagLayout()
```

Creates a gridbag layout.

Methods

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
```

---

public class **GridLayout**
extends Object
implements LayoutManager

A layout manager for a container that lays out grids.

Constructor Index

- **GridLayout**(int, int)
  Creates a grid layout with the specified rows and columns.
- **GridLayout**(int, int, int, int)
  Creates a grid layout with the specified rows, columns, horizontal gap, and vertical gap.

Method Index

- **addLayoutComponent**(String, Component)
  Adds the specified component with the specified name to the layout.
- **layoutContainer**(Container)
  Lays out the container in the specified panel.
- **minimumLayoutSize**(Container)
  Returns the minimum dimensions needed to layout the components contained in the specified panel.
- **preferredLayoutSize**(Container)
  Returns the preferred dimensions for this layout given the components int the specified panel.
- **removeLayoutComponent**(Component)
  Removes the specified component from the layout.
- **toString**()
  Returns the String representation of this GridLayout's values.

Constructors

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.

Methods

addLayoutComponent

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

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

removeLayoutComponent

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

Removes the specified component from the layout. Does not apply.

Parameters:
comp – the component to be removed

preferredLayoutSize

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

Returns the preferred dimensions for this layout given the components int the specified panel.
Parameters:
parent – the component which needs to be laid out
See Also:
minimumLayoutSize

minimumLayoutSize

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

Returns the minimum dimensions needed to layout the components contained in the specified panel.
Parameters:
parent – the component which needs to be laid out
See Also:
preferredLayoutSize

layoutContainer

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

Lays out the container in the specified panel.
Parameters:
parent – the specified component being laid out
See Also:
Container

toString

```
public String toString()
```

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

Class java.awt.Image

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

---

public class **Image**
extends Object

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

Variable Index

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

Constructor Index

- **Image**()

Method Index

- **flush**()
  Flushes all resources being used by this Image object.
- **getGraphics**()
  Gets a graphics object to draw into this image.
- **getHeight**(ImageObserver)
  Gets the actual height of the image.
- **getProperty**(String, ImageObserver)
  Gets a property of the image by name.
- **getSource**()
  Gets the object that produces the pixels for the image.
- **getWidth**(ImageObserver)
  Gets the actual width of the image.

Variables

UndefinedProperty

```
public final static Object UndefinedProperty
```

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

Constructors

Image

```
public Image()
```

Methods

getWidth

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

Gets the actual width of the image. If the width is not known yet then the ImageObserver will be notified later and –1 will be returned.
See Also:
getHeight, ImageObserver

getHeight

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

Gets the actual height of the image. If the height is not known yet then the ImageObserver will be notified later and –1 will be returned.
See Also:
getWidth, ImageObserver

getSource

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

Gets the object that produces the pixels for the image. This is used by the Image filtering classes and by the image conversion and scaling code.
See Also:
ImageProducer

getGraphics

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

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

See Also:

Graphics

getProperty

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

Gets a property of the image by name. Individual property names are defined by the various image formats. If a property is not defined for a particular image, this method will return the UndefinedProperty object. If the properties for this image are not yet known, then this method will return null and the ImageObserver object will be notified later. The property name "comment" should be used to store an optional comment which can be presented to the user as a description of the image, its source, or its author.

See Also:

ImageObserver, UndefinedProperty

flush

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

Flushes all resources being used by this Image object. This includes any pixel data that is being cached for rendering to the screen as well as any system resources that are being used to store data or pixels for the image. The image is reset to a state similar to when it was first created so that if it is again rendered, the image data will have to be recreated or fetched again from its source.

Class java.awt.Insets

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

---

public class **Insets**
extends Object
implements Cloneable

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

See Also:
LayoutManager, Container

Variable Index

- **bottom**
  The inset from the bottom.
- **left**
  The inset from the left.
- **right**
  The inset from the right.
- **top**
  The inset from the top.

Constructor Index

- **Insets**(int, int, int, int)
  Constructs and initializes a new Inset with the specified top, left, bottom, and right insets.

Method Index

- **clone**()

Creates a clone of the object.

- **toString()**

  Returns a String object representing this Inset's values.

Variables

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.

Constructors

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

Methods

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

Class java.awt.Label

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

---

public class **Label**
extends Component

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

Variable Index

- **CENTER**
  The center alignment.
- **LEFT**
  The left alignment.
- **RIGHT**
  The right alignment.

Constructor Index

- **Label**()
  Constructs an empty label.
- **Label**(String)
  Constructs a new label with the specified String of text.
- **Label**(String, int)
  Constructs a new label with the specified String of text and the specified alignment.

Method Index

- **addNotify**()
  Creates the peer for this label.

- **getAlignment**()
  Gets the current alignment of this label.
- **getText**()
  Gets the text of this label.
- **paramString**()
  Returns the parameter String of this label.
- **setAlignment**(int)
  Sets the alignment for this label to the specified alignment.
- **setText**(String)
  Sets the text for this label to the specified text.

Variables

LEFT

```
public final static int LEFT
```

The left alignment.

CENTER

```
public final static int CENTER
```

The center alignment.

RIGHT

```
public final static int RIGHT
```

The right alignment.

Constructors

Label

```
public Label()
```

Constructs an empty label.

Label

```
public Label(String label)
```

Constructs a new label with the specified String of text.
Parameters:

label – the text that makes up the label

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

Methods

addNotify

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

Creates the peer for this label. The peer allows us to modify the appearance of the label without changing its functionality.

Overrides:

addNotify in class Component

getAlignment

```
public int getAlignment()
```

Gets the current alignment of this label.

See Also:

setAlignment

setAlignment

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

Sets the alignment for this label to the specified alignment.

Parameters:

alignment – the alignment value

Throws: IllegalArgumentException

If an improper alignment was given.

See Also:

getAlignment

getText

```
public String getText()
```

Gets the text of this label.
See Also:
setText

setText

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

Sets the text for this label to the specified text.
Parameters:
label – the text that makes up the label
See Also:
getText

paramString

```
protected String paramString()
```

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

Class java.awt.List

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

---

public class **List**
extends Component

A scrolling list of text items.

Constructor Index

- **List**()
  Creates a new scrolling list initialized with no visible Lines or multiple selections.
- **List**(int, boolean)
  Creates a new scrolling list initialized with the specified number of visible lines and a boolean stating whether multiple selections are allowed or not.

Method Index

- **addItem**(String)
  Adds the specified item to the end of scrolling list.
- **addItem**(String, int)
  Adds the specified item to the end of scrolling list.
- **addNotify**()
  Creates the peer for the list.
- **allowsMultipleSelections**()
  Returns true if this list allows multiple selections.
- **clear**()
  Clears the list.
- **countItems**()
  Returns the number of items in the list.
- **delItem**(int)
  Delete an item from the list.
- **delItems**(int, int)
  Delete multiple items from the list.

- **deselect**(int)
  Deselects the item at the specified index.
- **getItem**(int)
  Gets the item associated with the specified index.
- **getRows**()
  Returns the number of visible lines in this list.
- **getSelectedIndex**()
  Get the selected item on the list or –1 if no item is selected.
- **getSelectedIndexes**()
  Returns the selected indexes on the list.
- **getSelectedItem**()
  Returns the selected item on the list or null if no item is selected.
- **getSelectedItems**()
  Returns the selected items on the list.
- **getVisibleIndex**()
  Gets the index of the item that was last made visible by the method makeVisible.
- **isSelected**(int)
  Returns true if the item at the specified index has been selected; false otherwise.
- **makeVisible**(int)
  Forces the item at the specified index to be visible.
- **minimumSize**(int)
  Returns the minimum dimensions needed for the amount of rows in the list.
- **minimumSize**()
  Returns the minimum dimensions needed for the list.
- **paramString**()
  Returns the parameter String of this list.
- **preferredSize**(int)
  Returns the preferred dimensions needed for the list with the specified amount of rows.
- **preferredSize**()
  Returns the preferred dimensions needed for the list.
- **removeNotify**()
  Removes the peer for this list.
- **replaceItem**(String, int)
  Replaces the item at the given index.
- **select**(int)
  Selects the item at the specified index.
- **setMultipleSelections**(boolean)
  Sets whether this list should allow multiple selections or not.

Constructors

List

```
public List()
```

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

List

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

Creates a new scrolling list initialized with the specified number of visible lines and a boolean stating whether multiple selections are allowed or not.

Parameters:

rows – the number of items to show.
multipleSelections – if true then multiple selections are allowed.

Methods

addNotify

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

Creates the peer for the list. The peer allows us to modify the list's appearance without changing its functionality.

Overrides:

addNotify in class Component

removeNotify

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

Removes the peer for this list. The peer allows us to modify the list's appearance without changing its functionality.

Overrides:

removeNotify in class Component

countItems

```
public int countItems()
```

Returns the number of items in the list.

See Also:

getItem

getItem

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

Gets the item associated with the specified index.

Parameters:

index – the position of the item

See Also:

countItems

addItem

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

Adds the specified item to the end of scrolling list.

Parameters:

item – the item to be added

addItem

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

Adds the specified item to the end of scrolling list.

Parameters:

item – the item to be added

index – the position at which to put in the item. The index is zero–based. If index is –1 then the item is added to the end. If index is greater than the number of items in the list, the item gets added at the end.

replaceItem

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

Replaces the item at the given index.

Parameters:

newValue – the new value to replace the existing item

index – the position of the item to replace

clear

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

Clears the list.

See Also:

delItem, delItems

delItem

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

Delete an item from the list.

delItems

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

```
                                    int end)
```

Delete multiple items from the list.

getSelectedIndex

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

Get the selected item on the list or –1 if no item is selected.
See Also:
select, deselect, isSelected

getSelectedIndexes

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

Returns the selected indexes on the list.
See Also:
select, deselect, isSelected

getSelectedItem

```
public synchronized String getSelectedItem()
```

Returns the selected item on the list or null if no item is selected.
See Also:
select, deselect, isSelected

getSelectedItems

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

Returns the selected items on the list.
See Also:
select, deselect, isSelected

select

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

Selects the item at the specified index.
Parameters:
index – the position of the item to select
See Also:
getSelectedItem, deselect, isSelected

deselect

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

Deselects the item at the specified index.

Parameters:

index – the position of the item to deselect

See Also:

select, getSelectedItem, isSelected

isSelected

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

Returns true if the item at the specified index has been selected; false otherwise.

Parameters:

index – the item to be checked

See Also:

select, deselect, isSelected

getRows

```
public int getRows()
```

Returns the number of visible lines in this list.

allowsMultipleSelections

```
public boolean allowsMultipleSelections()
```

Returns true if this list allows multiple selections.

See Also:

setMultipleSelections

setMultipleSelections

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

Sets whether this list should allow multiple selections or not.

Parameters:

v – the boolean to allow multiple selections

See Also:

allowsMultipleSelections

getVisibleIndex

```
public int getVisibleIndex()
```

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

makeVisible

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

Forces the item at the specified index to be visible.

Parameters:

index – the position of the item

See Also:

getVisibleIndex

preferredSize

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

Returns the preferred dimensions needed for the list with the specified amount of rows.

Parameters:

rows – amount of rows in list.

preferredSize

```
public Dimension preferredSize()
```

Returns the preferred dimensions needed for the list.

Returns:

the preferred size with the specified number of rows if the row size is greater than 0.

Overrides:

preferredSize in class Component

minimumSize

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

Returns the minimum dimensions needed for the amount of rows in the list.

Parameters:

rows – minimum amount of rows in the list

minimumSize

```
public Dimension minimumSize()
```

Returns the minimum dimensions needed for the list.

Returns:

the preferred size with the specified number of rows if the row size is greater than zero.

Overrides:

minimumSize in class Component

paramString

```
protected String paramString()
```

Returns the parameter String of this list.

Overrides:

paramString in class Component

Class java.awt.MediaTracker

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

---

public class **MediaTracker**
extends Object

A utility class to track the status of a number of media objects. Media objects could include images as well as audio clips, though currently only images are supported. To use it, simply create an instance and then call addImage() for each image to be tracked. Each image can be assigned a unique ID for indentification purposes. The IDs control the priority order in which the images are fetched as well as identifying unique subsets of the images that can be waited on independently. Here is an example:

```
import java.applet.Applet;
import java.awt.Color;
import java.awt.Image;
import java.awt.Graphics;
import java.awt.MediaTracker;
public class ImageBlaster extends Applet implements Runnable {
        MediaTracker tracker;
        Image bg;
        Image anim[] = new Image[5];
        int index;
        Thread animator;
        // Get the images for the background (id == 0) and the animation
        // frames (id == 1) and add them to the MediaTracker
        public void init() {
            tracker = new MediaTracker(this);
            bg = getImage(getDocumentBase(), "images/background.gif");
            tracker.addImage(bg, 0);
            for (int i = 0; i < 5; i++) {
                anim[i] = getImage(getDocumentBase(), "images/anim"+i+".gif");
                tracker.addImage(anim[i], 1);
            }
        }
        // Start the animation thread.
        public void start() {
            animator = new Thread(this);
            animator.start();
        }
        // Stop the animation thread.
        public void stop() {
            animator.stop();
            animator = null;
        }
        // Run the animation thread.
        // First wait for the background image to fully load and paint.
```

```
        // Then wait for all of the animation frames to finish loading.
        // Finally loop and increment the animation frame index.
        public void run() {
            try {
                tracker.waitForID(0);
                tracker.waitForID(1);
            } catch (InterruptedException e) {
                return;
            }
            Thread me = Thread.currentThread();
            while (animator == me) {
                try {
                    Thread.sleep(100);
                } catch (InterruptedException e) {
                    break;
                }
                synchronized (this) {
                    index++;
                    if (index >= anim.length) {
                        index = 0;
                    }
                }
                repaint();
            }
        }
        // The background image fills our frame so we don't need to clear
        // the applet on repaints, just call the paint method.
        public void update(Graphics g) {
            paint(g);
        }
        // Paint a large red rectangle if there are any errors loading the
        // images.  Otherwise always paint the background so that it appears
        // incrementally as it is loading.  Finally, only paint the current
        // animation frame if all of the frames (id == 1) are done loading
        // so that we don't get partial animations.
        public void paint(Graphics g) {
            if ((tracker.statusAll() & MediaTracker.ERRORED) != 0) {
                g.setColor(Color.red);
                g.fillRect(0, 0, size().width, size().height);
                return;
            }
            g.drawImage(bg, 0, 0, this);
            if ((tracker.statusID(1) & MediaTracker.COMPLETE) != 0) {
                g.drawImage(anim[index], 10, 10, this);
            }
        }
}
```

Variable Index

- **ABORTED**
  Flag indicating the download of some media was aborted.
- **COMPLETE**

Flag indicating the download of media completed successfully.

- **ERRORED**
  Flag indicating the download of some media encountered an error.
- **LOADING**
  Flag indicating some media is currently being loaded.

Constructor Index

- **MediaTracker**(Component)
  Creates a Media tracker to track images for a given Component.

Method Index

- **addImage**(Image, int)
  Adds an image to the list of images being tracked.
- **addImage**(Image, int, int, int)
  Adds a scaled image to the list of images being tracked.
- **checkAll**()
  Checks to see if all images have finished loading but does not start loading the images if they are not already loading.
- **checkAll**(boolean)
  Checks to see if all images have finished loading.
- **checkID**(int)
  Checks to see if all images tagged with the indicated ID have finished loading, but does not start loading the images if they are not already loading.
- **checkID**(int, boolean)
  Checks to see if all images tagged with the indicated ID have finished loading.
- **getErrorsAny**()
  Returns a list of all media that have encountered an error.
- **getErrorsID**(int)
  Returns a list of media with the specified ID that have encountered an error.
- **isErrorAny**()
  Checks the error status of all of the images.
- **isErrorID**(int)
  Checks the error status of all of the images with the specified ID.
- **statusAll**(boolean)
  Returns the boolean OR of the status of all of the media being tracked.
- **statusID**(int, boolean)
  Returns the boolean OR of the status of all of the media with a given ID.
- **waitForAll**()
  Starts loading all images.
- **waitForAll**(long)
  Starts loading all images.
- **waitForID**(int)
  Starts loading all images with the specified ID and waits until they have finished

loading or receive an error.

- **waitForID**(int, long)
  Starts loading all images with the specified ID.

Variables

LOADING

```
public final static int LOADING
```

Flag indicating some media is currently being loaded.
See Also:
statusAll, statusID

ABORTED

```
public final static int ABORTED
```

Flag indicating the download of some media was aborted.
See Also:
statusAll, statusID

ERRORED

```
public final static int ERRORED
```

Flag indicating the download of some media encountered an error.
See Also:
statusAll, statusID

COMPLETE

```
public final static int COMPLETE
```

Flag indicating the download of media completed successfully.
See Also:
statusAll, statusID

Constructors

MediaTracker

```
public MediaTracker(Component comp)
```

Creates a Media tracker to track images for a given Component.

Parameters:

comp – the component on which the images will eventually be drawn

Methods

addImage

```
public void addImage(Image image,
                     int id)
```

Adds an image to the list of images being tracked. The image will eventually be rendered at its default (unscaled) size.

Parameters:

image – the image to be tracked
id – the identifier used to later track this image

addImage

```
public synchronized void addImage(Image image,
                                  int id,
                                  int w,
                                  int h)
```

Adds a scaled image to the list of images being tracked. The image will eventually be rendered at the indicated size.

Parameters:

image – the image to be tracked
id – the identifier used to later track this image
w – the width that the image will be rendered at
h – the height that the image will be rendered at

checkAll

```
public boolean checkAll()
```

Checks to see if all images have finished loading but does not start loading the images if they are not already loading. If there is an error while loading or scaling an image then that image is considered "complete." Use isErrorAny() or isErrorID() to check for errors.

Returns:

true if all images have finished loading, were aborted or encountered an error

See Also:

checkAll, checkID, isErrorAny, isErrorID

checkAll

```
public synchronized boolean checkAll(boolean load)
```

Checks to see if all images have finished loading. If load is true, starts loading any images that are not yet being loaded. If there is an error while loading or scaling an image then that image is considered "complete." Use isErrorAny() or isErrorID() to check for errors.

Parameters:

load – start loading the images if this parameter is true

Returns:

true if all images have finished loading, were aborted or encountered an error

See Also:

isErrorAny, isErrorID, checkID, checkAll

isErrorAny

```
public synchronized boolean isErrorAny()
```

Checks the error status of all of the images.

Returns:

true if any of the images had an error during loading

See Also:

isErrorID, getErrorsAny

getErrorsAny

```
public synchronized Object[] getErrorsAny()
```

Returns a list of all media that have encountered an error.

Returns:

an array of media objects or null if there are no errors

See Also:

isErrorAny, getErrorsID

waitForAll

```
public void waitForAll() throws InterruptedException
```

Starts loading all images. Waits until they have finished loading, are aborted, or it receives an error. If there is an error while loading or scaling an image then that image is considered "complete." Use isErrorAny() or statusAll() to check for errors.

Throws: InterruptedException

Another thread has interrupted this thread.

See Also:

waitForID, waitForAll, isErrorAny, isErrorID

waitForAll

```
public synchronized boolean waitForAll(long ms) throws InterruptedException
```

Starts loading all images. Waits until they have finished loading, are aborted, it receives an error, or until the specified timeout has elapsed. If there is an error while loading or scaling an image then that image is considered "complete." Use isErrorAny() or statusAll() to check for errors.

Parameters:

ms – the length of time to wait for the loading to complete

Returns:

true if all images were successfully loaded

Throws: InterruptedException

Another thread has interrupted this thread.

See Also:

waitForID, waitForAll, isErrorAny, isErrorID

statusAll

```
public int statusAll(boolean load)
```

Returns the boolean OR of the status of all of the media being tracked.

Parameters:

load – specifies whether to start the media loading

See Also:

statusID, LOADING, ABORTED, ERRORED, COMPLETE

checkID

```
public boolean checkID(int id)
```

Checks to see if all images tagged with the indicated ID have finished loading, but does not start loading the images if they are not already loading. If there is an error while loading or scaling an image then that image is considered "complete." Use isErrorAny() or isErrorID() to check for errors.

Parameters:

id – the identifier used to determine which images to check

Returns:

true if all tagged images have finished loading, were aborted, or an error occurred.

See Also:

checkID, checkAll, isErrorAny, isErrorID

checkID

```
public synchronized boolean checkID(int id,
                                    boolean load)
```

Checks to see if all images tagged with the indicated ID have finished loading. If load is true, starts loading any images with that ID that are not yet being loaded. If there is an error while loading or scaling an image then that image is considered "complete." Use isErrorAny() or isErrorID() to check for errors.

Parameters:

id – the identifier used to determine which images to check
load – start loading the images if this parameter is true

Returns:

true if all tagged images have finished loading, were aborted, or an error occurred

See Also:

checkID, checkAll, isErrorAny, isErrorID

isErrorID

```
public synchronized boolean isErrorID(int id)
```

Checks the error status of all of the images with the specified ID.

Parameters:

id – the identifier used to determine which images to check

Returns:

true if any of the tagged images had an error during loading

See Also:

isErrorAny, getErrorsID

getErrorsID

```
public synchronized Object[] getErrorsID(int id)
```

Returns a list of media with the specified ID that have encountered an error.

Parameters:

id – the identifier used to determine which images to return

Returns:

an array of media objects or null if there are no errors

See Also:

isErrorID, getErrorsAny

waitForID

```
public void waitForID(int id) throws InterruptedException
```

Starts loading all images with the specified ID and waits until they have finished loading or receive an error. If there is an error while loading or scaling an image then that image is considered "complete." Use statusID() or isErrorID() to check for errors.

Parameters:

id – the identifier used to determine which images to wait for

Throws: InterruptedException

Another thread has interrupted this thread.

See Also:

waitForAll, waitForID, isErrorAny, isErrorID

waitForID

```
public synchronized boolean waitForID(int id,
                                      long ms) throws InterruptedException
```

Starts loading all images with the specified ID. Waits until they have finished loading, an error occurs, or the specified timeout has elapsed. If there is an error while loading or scaling an image then that image is considered "complete." Use statusID or isErrorID to check for errors.

Parameters:

id – the identifier used to determine which images to wait for

ms – the length of time to wait for the loading to complete

Throws: InterruptedException

Another thread has interrupted this thread.

See Also:

waitForAll, waitForID, isErrorAny, isErrorID

statusID

```
public int statusID(int id,
                    boolean load)
```

Returns the boolean OR of the status of all of the media with a given ID.

Parameters:

id – the identifier used to determine which images to check

load – specifies whether to start the media loading

See Also:

statusAll, LOADING, ABORTED, ERRORED, COMPLETE

Class java.awt.Menu

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

---

public class **Menu**
extends MenuItem
implements MenuContainer

A Menu that is a component of a menu bar.

Constructor Index

- **Menu**(String)
  Constructs a new Menu with the specified label.
- **Menu**(String, boolean)
  Constructs a new Menu with the specified label.

Method Index

- **add**(MenuItem)
  Adds the specified item to this menu.
- **add**(String)
  Adds an item with with the specified label to this menu.
- **addNotify**()
  Creates the menu's peer.
- **addSeparator**()
  Adds a separator line, or a hypen, to the menu at the current position.
- **countItems**()
  Returns the number of elements in this menu.
- **getItem**(int)
  Returns the item located at the specified index of this menu.
- **isTearOff**()
  Returns true if this is a tear–off menu.

- **remove**(int)
  Deletes the item from this menu at the specified index.
- **remove**(MenuComponent)
  Deletes the specified item from this menu.
- **removeNotify**()
  Removes the menu's peer.

Constructors

Menu

```
public Menu(String label)
```

Constructs a new Menu with the specified label. This menu can not be torn off – the menu will still appear on screen after the the mouse button has been released.

Parameters:

label – the label to be added to this menu

Menu

```
public Menu(String label,
            boolean tearOff)
```

Constructs a new Menu with the specified label. If tearOff is true, the menu can be torn off – the menu will still appear on screen after the the mouse button has been released.

Parameters:

label – the label to be added to this menu
tearOff – the boolean indicating whether or not the menu will be able to be torn off.

Methods

addNotify

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

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

Overrides:

addNotify in class MenuItem

removeNotify

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

Removes the menu's peer. The peer allows us to modify the appearance of the menu without changing its functionality.

Overrides:

removeNotify in class MenuComponent

isTearOff

```
public boolean isTearOff()
```

Returns true if this is a tear–off menu.

countItems

```
public int countItems()
```

Returns the number of elements in this menu.

getItem

```
public MenuItem getItem(int index)
```

Returns the item located at the specified index of this menu.

Parameters:

index – the position of the item to be returned

add

```
public synchronized MenuItem add(MenuItem mi)
```

Adds the specified item to this menu.

Parameters:

mi – the item to be added

add

```
public void add(String label)
```

Adds an item with with the specified label to this menu.

Parameters:

label – the text on the item

addSeparator

```
public void addSeparator()
```

Adds a separator line, or a hypen, to the menu at the current position.

remove

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

Deletes the item from this menu at the specified index.
Parameters:
index – the position of the item to be removed

remove

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

Deletes the specified item from this menu.
Parameters:
item – the item to be removed from the menu

Class java.awt.MenuBar

```
java.lang.Object
   |
   +----java.awt.MenuComponent
           |
           +----java.awt.MenuBar
```

---

public class **MenuBar**
extends MenuComponent
implements MenuContainer

A class that encapsulates the platform's concept of a menu bar bound to a Frame. In order to associate the MenuBar with an actual Frame, the Frame.setMenuBar() method should be called.

See Also:
setMenuBar

Constructor Index

- **MenuBar**()
  Creates a new menu bar.

Method Index

- **add**(Menu)
  Adds the specified menu to the menu bar.
- **addNotify**()
  Creates the menu bar's peer.
- **countMenus**()
  Counts the number of menus on the menu bar.
- **getHelpMenu**()
  Gets the help menu on the menu bar.
- **getMenu**(int)
  Gets the specified menu.
- **remove**(int)

Removes the menu located at the specified index from the menu bar.

- **remove**(MenuComponent)
  Removes the specified menu from the menu bar.
- **removeNotify**()
  Removes the menu bar's peer.
- **setHelpMenu**(Menu)
  Sets the help menu to the specified menu on the menu bar.

Constructors

MenuBar

```
public MenuBar()
```

Creates a new menu bar.

Methods

addNotify

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

Creates the menu bar's peer. The peer allows us to change the appearance of the menu bar without changing any of the menu bar's functionality.

removeNotify

```
public void removeNotify()
```

Removes the menu bar's peer. The peer allows us to change the appearance of the menu bar without changing any of the menu bar's functionality.

Overrides:

removeNotify in class MenuComponent

getHelpMenu

```
public Menu getHelpMenu()
```

Gets the help menu on the menu bar.

setHelpMenu

```
public synchronized void setHelpMenu(Menu m)
```

Sets the help menu to the specified menu on the menu bar.

Parameters:
m – the menu to be set

add

```
public synchronized Menu add(Menu m)
```

Adds the specified menu to the menu bar.
Parameters:
m – the menu to be added to the menu bar

remove

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

Removes the menu located at the specified index from the menu bar.
Parameters:
index – the position of the menu to be removed

remove

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

Removes the specified menu from the menu bar.
Parameters:
m – the menu to be removed

countMenus

```
public int countMenus()
```

Counts the number of menus on the menu bar.

getMenu

```
public Menu getMenu(int i)
```

Gets the specified menu.
Parameters:
i – the menu to be returned

Class java.awt.MenuComponent

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

---

public class **MenuComponent**
extends Object

The super class of all menu related components.

Constructor Index

- **MenuComponent**()

Method Index

- **getFont**()
  Gets the font used for this MenuItem.
- **getParent**()
  Returns the parent container.
- **getPeer**()
  Gets the MenuComponent's peer.
- **paramString**()
  Returns the String parameter of this MenuComponent.
- **postEvent**(Event)
  Posts the specified event to the menu.
- **removeNotify**()
  Removes the menu component's peer.
- **setFont**(Font)
  Sets the font to be used for this MenuItem to the specified font.
- **toString**()
  Returns the String representation of this MenuComponent's values.

Constructors

MenuComponent

```
public MenuComponent()
```

Methods

getParent

```
public MenuContainer getParent()
```

Returns the parent container.

getPeer

```
public MenuComponentPeer getPeer()
```

Gets the MenuComponent's peer. The peer allows us to modify the appearance of the menu component without changing the functionality of the menu component.

getFont

```
public Font getFont()
```

Gets the font used for this MenuItem.
Returns:
the font if one is used; null otherwise.

setFont

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

Sets the font to be used for this MenuItem to the specified font.
Parameters:
f – the font to be set

removeNotify

```
public void removeNotify()
```

Removes the menu component's peer. The peer allows us to modify the appearance of the menu component without changing the functionality of the menu component.

postEvent

```
public boolean postEvent(Event evt)
```

Posts the specified event to the menu.
Parameters:
evt – the event which is to take place

paramString

```
protected String paramString()
```

Returns the String parameter of this MenuComponent.

toString

```
public String toString()
```

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

Class java.awt.MenuItem

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

---

public class **MenuItem**
extends MenuComponent

A String item that represents a choice in a menu.

Constructor Index

- **MenuItem**(String)
  Constructs a new MenuItem with the specified label.

Method Index

- **addNotify**()
  Creates the menu item's peer.
- **disable**()
  Makes this menu item unselectable by the user.
- **enable**()
  Makes this menu item selectable by the user.
- **enable**(boolean)
  Conditionally enables a component.
- **getLabel**()
  Gets the label for this menu item.
- **isEnabled**()
  Checks whether the menu item is enabled.
- **paramString**()
  Returns the String parameter of the menu item.
- **setLabel**(String)
  Sets the label to be the specified label.

Constructors

MenuItem

```
public MenuItem(String label)
```

Constructs a new MenuItem with the specified label.
Parameters:
label – the label for this menu item. Note that "–" is reserved to mean a separator between menu items.

Methods

addNotify

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

Creates the menu item's peer. The peer allows us to modify the appearance of the menu item without changing its functionality.

getLabel

```
public String getLabel()
```

Gets the label for this menu item.

setLabel

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

Sets the label to be the specified label.
Parameters:
label – the label for this menu item

isEnabled

```
public boolean isEnabled()
```

Checks whether the menu item is enabled.

enable

```
public void enable()
```

Makes this menu item selectable by the user.

enable

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

Conditionally enables a component.
Parameters:
cond – enabled if true; disabled otherwise.
See Also:
enable, disable

disable

```
public void disable()
```

Makes this menu item unselectable by the user.

paramString

```
public String paramString()
```

Returns the String parameter of the menu item.
Overrides:
paramString in class MenuComponent

Class java.awt.Panel

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

---

public class **Panel**
extends Container

A Panel Container class. This produces a generic container.

Constructor Index

- **Panel**()
  Creates a new panel.

Method Index

- **addNotify**()
  Creates the Panel's peer.

Constructors

Panel

```
public Panel()
```

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

Methods

addNotify

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

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

Overrides:

addNotify in class Container

Class java.awt.Point

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

---

public class **Point**
extends Object

An x,y coordinate.

Variable Index

- **x**
  The x coordinate.
- **y**
  The y coordinate.

Constructor Index

- **Point**(int, int)
  Constructs and initializes a Point from the specified x and y coordinates.

Method Index

- **equals**(Object)
  Checks whether two pointers are equal.
- **hashCode**()
  Returns the hashcode for this Point.
- **move**(int, int)
  Moves the point.
- **toString**()
  Returns the String representation of this Point's coordinates.
- **translate**(int, int)
  Translates the point.

Variables

x

```
public int x
```

The x coordinate.

y

```
public int y
```

The y coordinate.

Constructors

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

Methods

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
```

---

public class **Polygon**
extends Object

A polygon consists of a list of x and y coordinates.

Variable Index

- **npoints**
  The total number of points.
- **xpoints**
  The array of x coordinates.
- **ypoints**
  The array of y coordinates.

Constructor Index

- **Polygon**()
  Creates an empty polygon.
- **Polygon**(int[], int[], int)
  Constructs and initializes a Polygon from the specified parameters.

Method Index

- **addPoint**(int, int)
  Appends a point to a polygon.
- **getBoundingBox**()
  Determines the area spanned by this Polygon.
- **inside**(int, int)
  Determines whether the point (x,y) is inside the Polygon.

Variables

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.

Constructors

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

Methods

addPoint

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

Appends a point to a polygon. If inside(x, y) or another operation that calculates the bounding box has already been performed, this method updates the bounds accordingly.

Parameters:

x – the x coordinate of the point
y – the y coordinate of the point

getBoundingBox

```
public Rectangle getBoundingBox()
```

Determines the area spanned by this Polygon.

Returns:

a Rectangle defining the bounds of the Polygon.

inside

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

Determines whether the point (x,y) is inside the Polygon. Uses an even–odd insideness rule (also known as an alternating rule).

Parameters:

x – the X coordinate of the point to be tested
y – the Y coordinate of the point to be tested

Based on code by Hanpeter van Vliet .

Class java.awt.Rectangle

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

---

public class **Rectangle**
extends Object

A rectangle defined by x, y, width and height.

Variable Index

- **height**
  The height of the rectangle.
- **width**
  The width of the rectangle.
- **x**
  The x coordinate of the rectangle.
- **y**
  The y coordinate of the rectangle.

Constructor Index

- **Rectangle**()
  Constructs a new rectangle.
- **Rectangle**(int, int, int, int)
  Constructs and initializes a rectangle with the specified parameters.
- **Rectangle**(int, int)
  Constructs a rectangle and initializes it with the specified width and height parameters.
- **Rectangle**(Point, Dimension)
  Constructs a rectangle and initializes it to a specified point and dimension.
- **Rectangle**(Point)
  Constructs a rectangle and initializes it to the specified point.
- **Rectangle**(Dimension)
  Constructs a rectangle and initializes it to the specified width and height.

Method Index

- **add**(int, int)
  Adds a point to a rectangle.
- **add**(Point)
  Adds a point to a rectangle.
- **add**(Rectangle)
  Adds a rectangle to a rectangle.
- **equals**(Object)
  Checks whether two rectangles are equal.
- **grow**(int, int)
  Grows the rectangle horizontally and vertically.
- **hashCode**()
  Returns the hashcode for this Rectangle.
- **inside**(int, int)
  Checks if the specified point lies inside a rectangle.
- **intersection**(Rectangle)
  Computes the intersection of two rectangles.
- **intersects**(Rectangle)
  Checks if two rectangles intersect.
- **isEmpty**()
  Determines whether the rectangle is empty.
- **move**(int, int)
  Moves the rectangle.
- **reshape**(int, int, int, int)
  Reshapes the rectangle.
- **resize**(int, int)
  Resizes the rectangle.
- **toString**()
  Returns the String representation of this Rectangle's values.
- **translate**(int, int)
  Translates the rectangle.
- **union**(Rectangle)
  Computes the union of two rectangles.

Variables

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.

Constructors

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

Methods

reshape

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

Reshapes the rectangle.

move

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

Moves the rectangle.

translate

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

Translates the rectangle.

- **resize**

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

Resizes the rectangle.

- **inside**

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

Checks if the specified point lies inside a rectangle.
Parameters:
x – the x coordinate
y – the y coordinate

- **intersects**

```
public boolean intersects(Rectangle r)
```

Checks if two rectangles intersect.

- **intersection**

```
public Rectangle intersection(Rectangle r)
```

Computes the intersection of two rectangles.

- **union**

```
public Rectangle union(Rectangle r)
```

Computes the union of two rectangles.

- **add**

```
public void add(int newx,
                int newy)
```

Adds a point to a rectangle. This results in the smallest rectangle that contains both the rectangle and the point.

- **add**

```
public void add(Point pt)
```

Adds a point to a rectangle. This results in the smallest rectangle that contains both the rectangle and the point.

add

```
public void add(Rectangle r)
```

Adds a rectangle to a rectangle. This results in the union of the two rectangles.

grow

```
public void grow(int h,
                 int v)
```

Grows the rectangle horizontally and vertically.

isEmpty

```
public boolean isEmpty()
```

Determines whether the rectangle is empty.

hashCode

```
public int hashCode()
```

Returns the hashcode for this Rectangle.
Overrides:
hashCode in class Object

equals

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

Checks whether two rectangles are equal.
Overrides:
equals in class Object

toString

```
public String toString()
```

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

Class java.awt.Scrollbar

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

---

public class **Scrollbar**
extends Component

A Scrollbar component.

Variable Index

- **HORIZONTAL**
  The horizontal Scrollbar variable.
- **VERTICAL**
  The vertical Scrollbar variable.

Constructor Index

- **Scrollbar**()
  Constructs a new vertical Scrollbar.
- **Scrollbar**(int)
  Constructs a new Scrollbar with the specified orientation.
- **Scrollbar**(int, int, int, int, int)
  Constructs a new Scrollbar with the specified orientation, value, page size, and minumum and maximum values.

Method Index

- **addNotify**()
  Creates the Scrollbar's peer.
- **getLineIncrement**()
  Gets the line increment for this scrollbar.

- **getMaximum**()
  Returns the maximum value of this Scrollbar.
- **getMinimum**()
  Returns the minimum value of this Scrollbar.
- **getOrientation**()
  Returns the orientation for this Scrollbar.
- **getPageIncrement**()
  Gets the page increment for this scrollbar.
- **getValue**()
  Returns the current value of this Scrollbar.
- **getVisible**()
  Returns the visible amount of the Scrollbar.
- **paramString**()
  Returns the String parameters for this Scrollbar.
- **setLineIncrement**(int)
  Sets the line increment for this scrollbar.
- **setPageIncrement**(int)
  Sets the page increment for this scrollbar.
- **setValue**(int)
  Sets the value of this Scrollbar to the specified value.
- **setValues**(int, int, int, int)
  Sets the values for this Scrollbar.

Variables

HORIZONTAL

```
public final static int HORIZONTAL
```

The horizontal Scrollbar variable.

VERTICAL

```
public final static int VERTICAL
```

The vertical Scrollbar variable.

Constructors

Scrollbar

```
public Scrollbar()
```

Constructs a new vertical Scrollbar.

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.

Scrollbar

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

Constructs a new Scrollbar with the specified orientation, value, page size, and minumum and maximum values.
Parameters:
orientation – either Scrollbar.HORIZONTAL or Scrollbar.VERTICAL
value – the scrollbar's value
visible – the size of the visible portion of the scrollable area. The scrollbar will use this value when paging up or down by a page.
minimum – the minimum value of the scrollbar
maximum – the maximum value of the scrollbar

Methods

addNotify

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

Creates the Scrollbar's peer. The peer allows you to modify the appearance of the Scrollbar without changing any of its functionality.
Overrides:
addNotify in class Component

getOrientation

```
public int getOrientation()
```

Returns the orientation for this Scrollbar.

getValue

```
public int getValue()
```

Returns the current value of this Scrollbar.

See Also:

getMinimum, getMaximum

setValue

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

Sets the value of this Scrollbar to the specified value.

Parameters:

value – the new value of the Scrollbar. If this value is below the current minimum or above the current maximum, it becomes the new one of those values, respectively.

See Also:

getValue

getMinimum

```
public int getMinimum()
```

Returns the minimum value of this Scrollbar.

See Also:

getMaximum, getValue

getMaximum

```
public int getMaximum()
```

Returns the maximum value of this Scrollbar.

See Also:

getMinimum, getValue

getVisible

```
public int getVisible()
```

Returns the visible amount of the Scrollbar.

setLineIncrement

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

Sets the line increment for this scrollbar. This is the value that will be added (subtracted) when the user hits the line down (up) gadgets.

getLineIncrement

```
public int getLineIncrement()
```

Gets the line increment for this scrollbar.

setPageIncrement

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

Sets the page increment for this scrollbar. This is the value that will be added (subtracted) when the user hits the page down (up) gadgets.

getPageIncrement

```
public int getPageIncrement()
```

Gets the page increment for this scrollbar.

setValues

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

Sets the values for this Scrollbar.
Parameters:
value – is the position in the current window.
visible – is the amount visible per page
minimum – is the minimum value of the scrollbar
maximum – is the maximum value of the scrollbar

paramString

```
protected String paramString()
```

Returns the String parameters for this Scrollbar.
Overrides:
paramString in class Component

Class java.awt.TextArea

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

---

public class **TextArea**
extends TextComponent

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

Constructor Index

- **TextArea**()
  Constructs a new TextArea.
- **TextArea**(int, int)
  Constructs a new TextArea with the specified number of rows and columns.
- **TextArea**(String)
  Constructs a new TextArea with the specified text displayed.
- **TextArea**(String, int, int)
  Constructs a new TextArea with the specified text and number of rows and columns.

Method Index

- **addNotify**()
  Creates the TextArea's peer.
- **appendText**(String)
  Appends the given text to the end.
- **getColumns**()
  Returns the number of columns in the TextArea.
- **getRows**()
  Returns the number of rows in the TextArea.
- **insertText**(String, int)

Inserts the specified text at the specified position.

- **minimumSize**(int, int)
  Returns the specified minimum size Dimensions of the TextArea.
- **minimumSize**()
  Returns the minimum size Dimensions of the TextArea.
- **paramString**()
  Returns the String of parameters for this TextArea.
- **preferredSize**(int, int)
  Returns the specified row and column Dimensions of the TextArea.
- **preferredSize**()
  Returns the preferred size Dimensions of the TextArea.
- **replaceText**(String, int, int)
  Replaces text from the indicated start to end position with the new text specified.

Constructors

TextArea

```
public TextArea()
```

Constructs a new TextArea.

TextArea

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

Constructs a new TextArea with the specified number of rows and columns.

Parameters:

rows – the number of rows
cols – the number of columns

TextArea

```
public TextArea(String text)
```

Constructs a new TextArea with the specified text displayed.

Parameters:

text – the text to be displayed

TextArea

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

Constructs a new TextArea with the specified text and number of rows and

columns.

Parameters:

text – the text to be displayed
rows – the number of rows
cols – the number of cols

addNotify

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

Creates the TextArea's peer. The peer allows us to modify the appearance of the TextArea without changing any of its functionality.

Overrides:

addNotify in class Component

insertText

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

Inserts the specified text at the specified position.

Parameters:

str – the text to insert.
pos – the position at which to insert.

See Also:

setText, replaceText

appendText

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

Appends the given text to the end.

Parameters:

str – the text to insert

See Also:

insertText

replaceText

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

Replaces text from the indicated start to end position with the new text specified.

Parameters:

str – the text to use as the replacement.
start – the start position.
end – the end position.

See Also:

insertText, replaceText

getRows

```
public int getRows()
```

Returns the number of rows in the TextArea.

getColumns

```
public int getColumns()
```

Returns the number of columns in the TextArea.

preferredSize

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

Returns the specified row and column Dimensions of the TextArea.

Parameters:

rows – the preferred rows amount
cols – the preferred columns amount

preferredSize

```
public Dimension preferredSize()
```

Returns the preferred size Dimensions of the TextArea.

Overrides:

preferredSize in class Component

minimumSize

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

Returns the specified minimum size Dimensions of the TextArea.

Parameters:

rows – the minimum row size
cols – the minimum column size

minimumSize

```
public Dimension minimumSize()
```

Returns the minimum size Dimensions of the TextArea.
Overrides:
minimumSize in class Component

paramString

```
protected String paramString()
```

Returns the String of parameters for this TextArea.
Overrides:
paramString in class TextComponent

Class java.awt.TextComponent

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

---

public class **TextComponent**
extends Component

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

Method Index

- **getSelectedText**()
  Returns the selected text contained in this TextComponent.
- **getSelectionEnd**()
  Returns the selected text's end position.
- **getSelectionStart**()
  Returns the selected text's start position.
- **getText**()
  Returns the text contained in this TextComponent.
- **isEditable**()
  Returns the boolean indicating whether this TextComponent is editable or not.
- **paramString**()
  Returns the String of parameters for this TextComponent.
- **removeNotify**()
  Removes the TextComponent's peer.
- **select**(int, int)
  Selects the text found between the specified start and end locations.
- **selectAll**()
  Selects all the text in the TextComponent.
- **setEditable**(boolean)
  Sets the specified boolean to indicate whether or not this TextComponent should be editable.
- **setText**(String)
  Sets the text of this TextComponent to the specified text.

Methods

removeNotify

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

Removes the TextComponent's peer. The peer allows us to modify the appearance of the TextComponent without changing its functionality.

Overrides:

removeNotify in class Component

setText

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

Sets the text of this TextComponent to the specified text.

Parameters:

t – the new text to be set

See Also:

getText

getText

```
public String getText()
```

Returns the text contained in this TextComponent.

See Also:

setText

getSelectedText

```
public String getSelectedText()
```

Returns the selected text contained in this TextComponent.

See Also:

setText

isEditable

```
public boolean isEditable()
```

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

See Also:

setEditable

setEditable

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

Sets the specified boolean to indicate whether or not this TextComponent should be editable.

Parameters:

t – the boolean to be set

See Also:

isEditable

getSelectionStart

```
public int getSelectionStart()
```

Returns the selected text's start position.

getSelectionEnd

```
public int getSelectionEnd()
```

Returns the selected text's end position.

select

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

Selects the text found between the specified start and end locations.

Parameters:

selStart – the start position of the text

selEnd – the end position of the text

selectAll

```
public void selectAll()
```

Selects all the text in the TextComponent.

paramString

```
protected String paramString()
```

Returns the String of parameters for this TextComponent.

Overrides:

paramString in class Component

Class java.awt.TextField

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

---

public class **TextField**
extends TextComponent

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

Constructor Index

- **TextField**()
  Constructs a new TextField.
- **TextField**(int)
  Constructs a new TextField initialized with the specified columns.
- **TextField**(String)
  Constructs a new TextField initialized with the specified text.
- **TextField**(String, int)
  Constructs a new TextField initialized with the specified text and columns.

Method Index

- **addNotify**()
  Creates the TextField's peer.
- **echoCharIsSet**()
  Returns true if this TextField has a character set for echoing.
- **getColumns**()
  Returns the number of columns in this TextField.
- **getEchoChar**()
  Returns the character to be used for echoing.
- **minimumSize**(int)
  Returns the minimum size Dimensions needed for this TextField with the specified amount of columns.

- **minimumSize**()
  Returns the minimum size Dimensions needed for this TextField.
- **paramString**()
  Returns the String of parameters for this TExtField.
- **preferredSize**(int)
  Returns the preferred size Dimensions needed for this TextField with the specified amount of columns.
- **preferredSize**()
  Returns the preferred size Dimensions needed for this TextField.
- **setEchoCharacter**(char)
  Sets the echo character for this TextField.

Constructors

TextField

```
public TextField()
```

Constructs a new TextField.

TextField

```
public TextField(int cols)
```

Constructs a new TextField initialized with the specified columns.
Parameters:
cols – the number of columns

TextField

```
public TextField(String text)
```

Constructs a new TextField initialized with the specified text.
Parameters:
text – the text to be displayed

TextField

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

Constructs a new TextField initialized with the specified text and columns.
Parameters:
text – the text to be displayed
cols – the number of columns

Methods

addNotify

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

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

Overrides:

addNotify in class Component

getEchoChar

```
public char getEchoChar()
```

Returns the character to be used for echoing.

See Also:

setEchoCharacter, echoCharIsSet

echoCharIsSet

```
public boolean echoCharIsSet()
```

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

See Also:

setEchoCharacter, getEchoChar

getColumns

```
public int getColumns()
```

Returns the number of columns in this TextField.

setEchoCharacter

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

Sets the echo character for this TextField. This is useful for fields where the user input shouldn't be echoed to the screen, as in the case of a TextField that represents a password.

Parameters:

c – the echo character for this TextField

See Also:

echoCharIsSet, getEchoChar

preferredSize

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

Returns the preferred size Dimensions needed for this TextField with the specified amount of columns.

Parameters:

cols – the number of columns in this TextField

preferredSize

```
public Dimension preferredSize()
```

Returns the preferred size Dimensions needed for this TextField.

Overrides:

preferredSize in class Component

minimumSize

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

Returns the minimum size Dimensions needed for this TextField with the specified amount of columns.

Parameters:

cols – the number of columns in this TextField

minimumSize

```
public Dimension minimumSize()
```

Returns the minimum size Dimensions needed for this TextField.

Overrides:

minimumSize in class Component

paramString

```
protected String paramString()
```

Returns the String of parameters for this TExtField.

Overrides:

paramString in class TextComponent

Class java.awt.Toolkit

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

---

public class **Toolkit**
extends Object

An AWT toolkit. It is used to bind the abstract AWT classes to a particular native toolkit implementation.

Constructor Index

- **Toolkit**()

Method Index

- **checkImage**(Image, int, int, ImageObserver)
  Returns the status of the construction of the indicated method at the indicated width and height for the default screen.
- **createButton**(Button)
  Uses the specified Peer interface to create a new Button.
- **createCanvas**(Canvas)
  Uses the specified Peer interface to create a new Canvas.
- **createCheckbox**(Checkbox)
  Uses the specified Peer interface to create a new Checkbox.
- **createCheckboxMenuItem**(CheckboxMenuItem)
  Uses the specified Peer interface to create a new CheckboxMenuItem.
- **createChoice**(Choice)
  Uses the specified Peer interface to create a new Choice.
- **createDialog**(Dialog)
  Uses the specified Peer interface to create a new Dialog.
- **createFileDialog**(FileDialog)
  Uses the specified Peer interface to create a new FileDialog.
- **createFrame**(Frame)
  Uses the specified Peer interface to create a new Frame.
- **createImage**(ImageProducer)

Creates an image with the specified image producer.

- **createLabel**(Label)
  Uses the specified Peer interface to create a new Label.
- **createList**(List)
  Uses the specified Peer interface to create a new List.
- **createMenu**(Menu)
  Uses the specified Peer interface to create a new Menu.
- **createMenuBar**(MenuBar)
  Uses the specified Peer interface to create a new MenuBar.
- **createMenuItem**(MenuItem)
  Uses the specified Peer interface to create a new MenuItem.
- **createPanel**(Panel)
  Uses the specified Peer interface to create a new Panel.
- **createScrollbar**(Scrollbar)
  Uses the specified Peer interface to create a new Scrollbar.
- **createTextArea**(TextArea)
  Uses the specified Peer interface to create a new TextArea.
- **createTextField**(TextField)
  Uses the specified Peer interface to create a new TextField.
- **createWindow**(Window)
  Uses the specified Peer interface to create a new Window.
- **getColorModel**()
  Returns the ColorModel of the screen.
- **getDefaultToolkit**()
  Returns the default toolkit.
- **getFontList**()
  Returns the names of the available fonts.
- **getFontMetrics**(Font)
  Returns the screen metrics of the font.
- **getImage**(String)
  Returns an image which gets pixel data from the specified file.
- **getImage**(URL)
  Returns an image which gets pixel data from the specified URL.
- **getScreenResolution**()
  Returns the screen resolution in dots–per–inch.
- **getScreenSize**()
  Gets the size of the screen.
- **prepareImage**(Image, int, int, ImageObserver)
  Prepares an image for rendering on the default screen at the specified width and height.
- **sync**()
  Syncs the graphics state; useful when doing animation.

Constructors

- **Toolkit**

```
public Toolkit()
```

- **createButton**

```
protected abstract ButtonPeer createButton(Button target)
```

Uses the specified Peer interface to create a new Button.
Parameters:
target – the Button to be created

- **createTextField**

```
protected abstract TextFieldPeer createTextField(TextField target)
```

Uses the specified Peer interface to create a new TextField.
Parameters:
target – the TextField to be created

- **createLabel**

```
protected abstract LabelPeer createLabel(Label target)
```

Uses the specified Peer interface to create a new Label.
Parameters:
target – the Label to be created

- **createList**

```
protected abstract ListPeer createList(List target)
```

Uses the specified Peer interface to create a new List.
Parameters:
target – the List to be created

- **createCheckbox**

```
protected abstract CheckboxPeer createCheckbox(Checkbox target)
```

Uses the specified Peer interface to create a new Checkbox.
Parameters:
target – the Checkbox to be created

- **createScrollbar**

```
protected abstract ScrollbarPeer createScrollbar(Scrollbar target)
```

Uses the specified Peer interface to create a new Scrollbar.
Parameters:
target – the Scrollbar to be created

createTextArea

```
protected abstract TextAreaPeer createTextArea(TextArea target)
```

Uses the specified Peer interface to create a new TextArea.
Parameters:
target – the TextArea to be created

createChoice

```
protected abstract ChoicePeer createChoice(Choice target)
```

Uses the specified Peer interface to create a new Choice.
Parameters:
target – the Choice to be created

createFrame

```
protected abstract FramePeer createFrame(Frame target)
```

Uses the specified Peer interface to create a new Frame.
Parameters:
target – the Frame to be created

createCanvas

```
protected abstract CanvasPeer createCanvas(Canvas target)
```

Uses the specified Peer interface to create a new Canvas.
Parameters:
target – the Canvas to be created

createPanel

```
protected abstract PanelPeer createPanel(Panel target)
```

Uses the specified Peer interface to create a new Panel.
Parameters:
target – the Panel to be created

createWindow

```
protected abstract WindowPeer createWindow(Window target)
```

Uses the specified Peer interface to create a new Window.
Parameters:
target – the Window to be created

createDialog

```
protected abstract DialogPeer createDialog(Dialog target)
```

Uses the specified Peer interface to create a new Dialog.
Parameters:
target – the Dialog to be created

createMenuBar

```
protected abstract MenuBarPeer createMenuBar(MenuBar target)
```

Uses the specified Peer interface to create a new MenuBar.
Parameters:
target – the MenuBar to be created

createMenu

```
protected abstract MenuPeer createMenu(Menu target)
```

Uses the specified Peer interface to create a new Menu.
Parameters:
target – the Menu to be created

createMenuItem

```
protected abstract MenuItemPeer createMenuItem(MenuItem target)
```

Uses the specified Peer interface to create a new MenuItem.
Parameters:
target – the MenuItem to be created

createFileDialog

```
protected abstract FileDialogPeer createFileDialog(FileDialog target)
```

Uses the specified Peer interface to create a new FileDialog.
Parameters:
target – the FileDialog to be created

createCheckboxMenuItem

```
protected abstract CheckboxMenuItemPeer createCheckboxMenuItem(CheckboxMenuItem target)
```

Uses the specified Peer interface to create a new CheckboxMenuItem.

Parameters:
target – the CheckboxMenuItem to be created

getScreenSize

```
public abstract Dimension getScreenSize()
```

Gets the size of the screen.

getScreenResolution

```
public abstract int getScreenResolution()
```

Returns the screen resolution in dots–per–inch.

getColorModel

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

Returns the ColorModel of the screen.

getFontList

```
public abstract String[] getFontList()
```

Returns the names of the available fonts.

getFontMetrics

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

Returns the screen metrics of the font.

sync

```
public abstract void sync()
```

Syncs the graphics state; useful when doing animation.

getDefaultToolkit

```
public static synchronized Toolkit getDefaultToolkit()
```

Returns the default toolkit. This is controlled by the "awt.toolkit" property.
Throws:AWTError
Toolkit not found or could not be instantiated.

getImage

```
public abstract Image getImage(String filename)
```

Returns an image which gets pixel data from the specified file.

Parameters:

filename – the file containing the pixel data in one of the recognized file formats

getImage

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

Returns an image which gets pixel data from the specified URL.

Parameters:

url – the URL to use in fetching the pixel data

prepareImage

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

Prepares an image for rendering on the default screen at the specified width and height.

checkImage

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

Returns the status of the construction of the indicated method at the indicated width and height for the default screen.

createImage

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

Creates an image with the specified image producer.

Parameters:

producer – the image producer to be used

Class java.awt.Window

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

---

public class **Window**
extends Container

A Window is a top–level window with no borders and no menubar. It could be used to implement a pop–up menu. The default layout for a window is BorderLayout.

Constructor Index

- **Window**(Frame)
  Constructs a new Window initialized to an invisible state.

Method Index

- **addNotify**()
  Creates the Window's peer.
- **dispose**()
  Disposes of the Window.
- **getToolkit**()
  Returns the toolkit of this frame.
- **getWarningString**()
  Gets the warning string for this window.
- **pack**()
  Packs the components of the Window.
- **show**()
  Shows the Window.
- **toBack**()
  Sends the frame to the back of the Window.
- **toFront**()
  Brings the frame to the front of the Window.

Constructors

Window

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

Constructs a new Window initialized to an invisible state. It behaves as a modal dialog in that it will block input to other windows when shown.

Parameters:

parent – the owner of the dialog

See Also:

resize, show

Methods

addNotify

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

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

Overrides:

addNotify in class Container

pack

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

Packs the components of the Window.

show

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

Shows the Window. This will bring the window to the front if the window is already visible.

Overrides:

show in class Component

See Also:

hide

dispose

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

Disposes of the Window. This method must be called to release the resources that

are used for the window.

toFront

```
public void toFront()
```

Brings the frame to the front of the Window.

toBack

```
public void toBack()
```

Sends the frame to the back of the Window.

getToolkit

```
public Toolkit getToolkit()
```

Returns the toolkit of this frame.
Overrides:
getToolkit in class Component
See Also:
Toolkit

getWarningString

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

Gets the warning string for this window. This is a string that will be displayed somewhere in the visible area of windows that are not secure.

Class java.awt.AWTException

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

---

public class **AWTException**
extends Exception

Signals that an Absract Window Toolkit exception has occurred.

Constructor Index

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

Constructors

AWTException

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

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

Class java.awt.AWTError

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

---

public class **AWTError**
extends Error

An AWT Error.

Constructor Index

- **AWTError**(String)

Constructors

AWTError

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