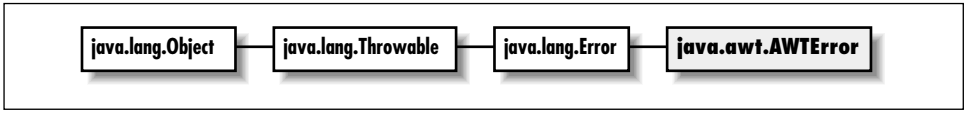
19

java.awt Reference

19.1 AWTError



Description

An `AWTError`; thrown to indicate a serious runtime error.

Class Definition

```
public class java.awt.AWTError
    extends java.lang.Error {

  // Constructors
  public AWTError (String message);
}
```

Constructors

AWTError

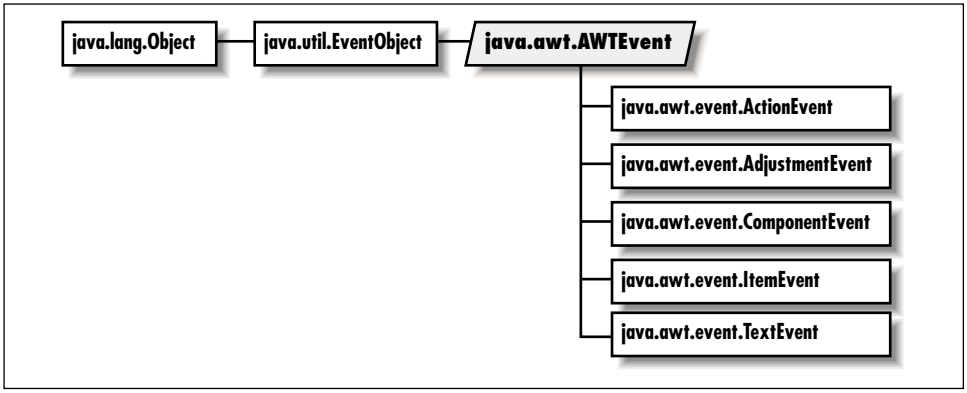
```
public AWTError (String message)
```

Parameters *message* Detail message

See Also

`Error`, `String`

19.2 AWTEvent ★



Description

The root class of all AWT events. Subclasses of this class are the replacement for `java.awt.Event`, which is only used for the Java 1.0.2 event model. In Java 1.1, event objects are passed from event source components to objects implementing a corresponding listener interface. Some event sources have a corresponding interface, too. For example, `AdjustmentEvents` are passed from `Adjustable` objects to `AdjustmentListeners`. Some event types do not have corresponding interfaces; for example, `ActionEvents` are passed from `Buttons` to `ActionListeners`, but there is no "Actionable" interface that `Button` implements.

Class Definition

```
public abstract class java.awt.AWTEvent
    extends java.util.EventObject {

  // Constants
  public final static long ACTION_EVENT_MASK;
  public final static long ADJUSTMENT_EVENT_MASK;
  public final static long COMPONENT_EVENT_MASK;
```

```
    public final static long CONTAINER_EVENT_MASK;
    public final static long FOCUS_EVENT_MASK;
    public final static long ITEM_EVENT_MASK;
    public final static long KEY_EVENT_MASK;
    public final static long MOUSE_EVENT_MASK;
    public final static long MOUSE_MOTION_EVENT_MASK;
    public final static long RESERVED_ID_MAX;
    public final static long TEXT_EVENT_MASK;
    public final static long WINDOW_EVENT_MASK;

    // Variables
    protected boolean consumed;
    protected int id;

    // Constructors
    public AWTEvent (Event event);
    public AWTEvent (Object source, int id);

    // Instance Methods
    public int getID();
    public String paramString();
    public String toString();

    // Protected Instance Methods
    protected void consume();
    protected boolean isConsumed();
}
```

Constants

ACTION_EVENT_MASK

```
public static final long ACTION_EVENT_MASK
```

The mask for action events.

ADJUSTMENT_EVENT_MASK

```
public static final long ADJUSTMENT_EVENT_MASK
```

The mask for adjustment events.

COMPONENT_EVENT_MASK

```
public static final long COMPONENT_EVENT_MASK
```

The mask for component events.

CONTAINER_EVENT_MASK

```
public static final long CONTAINER_EVENT_MASK
```

The mask for container events.

FOCUS_EVENT_MASK

```
public static final long FOCUS_EVENT_MASK
```

The mask for focus events.

ITEM_EVENT_MASK

```
public static final long ITEM_EVENT_MASK
```

The mask for item events.

KEY_EVENT_MASK

```
public static final long KEY_EVENT_MASK
```

The mask for key events.

MOUSE_EVENT_MASK

```
public static final long MOUSE_EVENT_MASK
```

The mask for mouse events.

MOUSE_MOTION_EVENT_MASK

```
public static final long MOUSE_MOTION_EVENT_MASK
```

The mask for mouse motion events.

RESERVED_ID_MAX

```
public static final int
```

The maximum reserved event id.

TEXT_EVENT_MASK

```
public static final long TEXT_EVENT_MASK
```

The mask for text events.

WINDOW_EVENT_MASK

```
public static final long WINDOW_EVENT_MASK
```

The mask for window events.

Variables

consumed

```
protected boolean consumed
```

If `consumed` is `true`, the event will not be sent back to the peer. Semantic events will never be sent back to a peer; thus `consumed` is always `true` for semantic events.

id

```
protected int id
```

The type ID of this event.

Constructors

AWTEvent

```
public AWTEvent (Event event)
```

Parameters	*event*	A version 1.0.2 `java.awt.Event` object.
Description	Constructs a 1.1 `java.awt.AWTEvent` derived from a 1.0.2 `java.awt.Event` object.	

```
public AWTEvent (Object source, int id)
```

Parameters	*source*	The object that the event originated from.
	id	An event type ID.
Description	Constructs an `AWTEvent` object.	

Instance Methods

getID

```
public int getID()
```

Returns	The type ID of the event.

paramString

```
public String paramString()
```

Returns	A string with the current settings of `AWTEvent`.
Description	Helper method for `toString()` that generates a string of current settings.

toString

```
public String toString()
```

Returns A string representation of the `AWTEvent` object.
Overrides `Object.toString()`

Protected Instance Methods

consume

```
protected void consume()
```

Description Consumes the event so it is not sent back to its source.

isConsumed

```
public boolean isConsumed()
```

Returns A flag indicating whether this event has been consumed.

See Also

`ActionEvent`, `AdjustmentEvent`, `ComponentEvent`, `Event`, `EventObject`, `FocusEvent`, `ItemEvent`, `KeyEvent`, `MouseEvent`, `WindowEvent`

19.3 AWTEventMulticaster ⋆

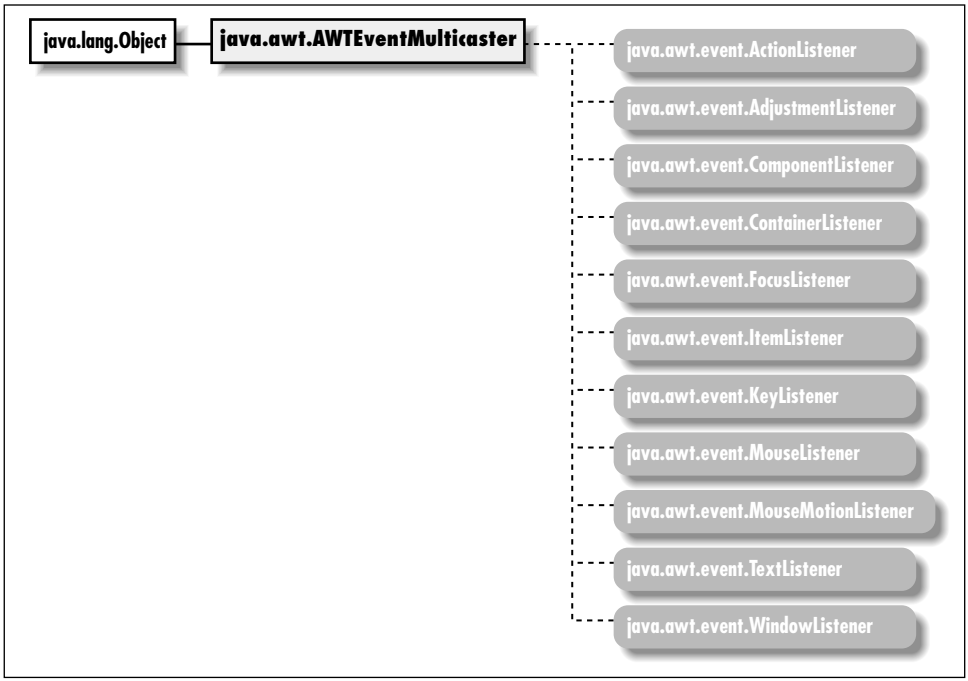
Description

This class multicasts events to event listeners. Each multicaster has two listeners, cunningly named `a` and `b`. When an event source calls one of the listener methods of the multicaster, the multicaster calls the same listener method on both `a` and `b`. Multicasters are built into trees using the static `add()` and `remove()` methods. In this way a single event can be sent to many listeners.

Static methods make it easy to implement event multicasting in component subclasses. Each time an `add<type>Listener()` function is called in the component subclass, call the corresponding `AWTEventMulticaster.add()` method to chain together (or "tree up") listeners. Similarly, when a `remove<type>Listener()` function is called, `AWTEventMulticaster.remove()` can be called to remove a chained listener.

Class Definition

```
public class java.awt.AWTEventMulticaster
  extends java.lang.Object
  implements java.awt.event.ActionListener, java.awt.event.AdjustmentListener,
             java.awt.event.ComponentListener, java.awt.event.ContainerListener,
             java.awt.event.FocusListener, java.awt.event.ItemListener,
             java.awt.event.KeyListener, java.awt.event.MouseListener,
```




```
            java.awt.event.MouseMotionListener, java.awt.event.TextListener,
            java.awt.event.WindowListener {

// Variables
protected EventListener a;
protected EventListener b;

// Constructors
protected AWTEventMulticaster(EventListener a, EventListener b);

// Class Methods
public static ActionListener add(ActionListener a, ActionListener b);
public static AdjustmentListener add(AdjustmentListener a,
    AdjustmentListener b);
public static ComponentListener add(ComponentListener a,
    ComponentListener b);
public static ContainerListener add(ContainerListener a,
    ContainerListener b);
public static FocusListener add(FocusListener a, FocusListener b);
public static ItemListener add(ItemListener a, ItemListener b);
public static KeyListener add(KeyListener a, KeyListener b);
public static MouseListener add(MouseListener a, MouseListener b);
public static MouseMotionListener add(MouseMotionListener a,
    MouseMotionListener b);
```

```
public static TextListener add(TextListener a, TextListener b);
public static WindowListener add(WindowListener a, WindowListener b);
protected static EventListener addInternal(EventListener a, EventListener b);
public static ActionListener remove(ActionListener l, ActionListener oldl);
public static AdjustmentListener remove(AdjustmentListener l,
    AdjustmentListener oldl);
public static ComponentListener remove(ComponentListener l,
    ComponentListener oldl);
public static ContainerListener remove(ContainerListener l,
    ContainerListener oldl);
public static FocusListener remove(FocusListener l, FocusListener oldl);
public static ItemListener remove(ItemListener l, ItemListener oldl);
public static KeyListener remove(KeyListener l, KeyListener oldl);
public static MouseListener remove(MouseListener l, MouseListener oldl);
public static MouseMotionListener remove(MouseMotionListener l,
    MouseMotionListener oldl);
public static TextListener remove(TextListener l, TextListener oldl);
public static WindowListener remove(WindowListener l, WindowListener;
protected static EventListener removeInternal(EventListener l,
    EventListener oldl);

// Instance Methods
public void actionPerformed(ActionEvent e);
public void adjustmentValueChanged(AdjustmentEvent e);
public void componentAdded(ContainerEvent e);
public void componentHidden(ComponentEvent e);
public void componentMoved(ComponentEvent e);
public void componentRemoved(ContainerEvent e);
public void componentResized(ComponentEvent e);
public void componentShown(ComponentEvent e);
public void focusGained(FocusEvent e);
public void focusLost(FocusEvent e);
public void itemStateChanged(ItemEvent e);
public void keyPressed(KeyEvent e);
public void keyReleased(KeyEvent e);
public void keyTyped(KeyEvent e);
public void mouseClicked(MouseEvent e);
public void mouseDragged(MouseEvent e);
public void mouseEntered(MouseEvent e);
public void mouseExited(MouseEvent e);
public void mouseMoved(MouseEvent e);
public void mousePressed(MouseEvent e);
public void mouseReleased(MouseEvent e);
public void textValueChanged(TextEvent e);
public void windowActivated(WindowEvent e);
public void windowClosed(WindowEvent e);
public void windowClosing(WindowEvent e);
public void windowDeactivated(WindowEvent e);
public void windowDeiconified(WindowEvent e);
```

```
  public void windowIconified(WindowEvent e);
  public void windowOpened(WindowEvent e);
  // Protected Instance Methods
  protected EventListener remove(EventListener oldl);
  protected void saveInternal(ObjectOutputStream s, String k) throws IOException;
}
```

Variables

a

```
protected EventListener a
```

One of the EventListeners this AWTEventMulticaster sends events to.

b

```
protected EventListener b
```

One of the EventListeners this AWTEventMulticaster sends events to.

Constructors

AWTEventMulticaster

```
protected AWTEventMulticaster (EventListener a,
EventListener b)
```

Parameters	*a*	A listener that receives events.
	b	A listener that receives events.
Description	Constructs an AWTEventMulticaster that sends events it receives to the supplied listeners. The constructor is protected because it is only the class methods of AWTEventMulticaster that ever instantiate this class.	

Class Methods

add

```
public static ActionListener add (ActionListener a,
ActionListener b)
```

Parameters	*a*	An event listener.
	b	An event listener.
Returns	A listener object that passes events to a and b.	

```
public static AdjustmentListener add (AdjustmentListener
a, AdjustmentListener b)
```

Parameters	*a*	An event listener.

b An event listener.

Returns A listener object that passes events to a and b.

```
public static ComponentListener add (ComponentListener a,
ComponentListener b)
```

Parameters *a* An event listener.
b An event listener.

Returns A listener object that passes events to a and b.

```
public static ContainerListener add (ContainerListener a,
ContainerListener b)
```

Parameters *a* An event listener.
b An event listener.

Returns A listener object that passes events to a and b.

```
public static FocusListener add (FocusListener a,
FocusListener b)
```

Parameters *a* An event listener.
b An event listener.

Returns A listener object that passes events to a and b.

```
public static ItemListener add (ItemListener a,
ItemListener b)
```

Parameters *a* An event listener.
b An event listener.

Returns A listener object that passes events to a and b.

```
public static KeyListener add (KeyListener a, KeyListener
b)
```

Parameters *a* An event listener.
b An event listener.

Returns A listener object that passes events to a and b.

```
public static MouseListener add (MouseListener a,
MouseListener b)
```

Parameters *a* An event listener.
b An event listener.

Returns A listener object that passes events to a and b.

```
public static MouseMotionListener add (MouseMotionListener
a, MouseMotionListener b)
```

Parameters *a* An event listener.
b An event listener.
Returns A listener object that passes events to a and b.

```
public static TextListener add (TextListener a,
TextListener b)
```

Parameters *a* An event listener.
b An event listener.
Returns A listener object that passes events to a and b.

```
public static WindowListener add (WindowListener a,
WindowListener b)
```

Parameters *a* An event listener.
b An event listener.
Returns A listener object that passes events to a and b.

addInternal

```
public static EventListener addInternal (EventListener a,
EventListener b)
```

Parameters *a* An event listener.
b An event listener.
Returns A listener object that passes events to a and b.
Description This method is a helper for the `add()` methods.

remove

```
public static ActionListener remove (ActionListener l,
ActionListener oldl)
```

Parameters *l* An event listener.
oldl An event listener.
Returns A listener object that multicasts to `l` but not `oldl`.

```
public static AdjustmentListener remove
(AdjustmentListener l, AdjustmentListener oldl)
```

Parameters *l* An event listener.
oldl An event listener.
Returns A listener object that multicasts to `l` but not `oldl`.

```
public static ComponentListener remove (ComponentListener
l, ComponentListener oldl)
```

Parameters *l* An event listener.

Parameters *oldl* An event listener.
Returns A listener object that multicasts to `l` but not `oldl`.

```
public static ContainerListener remove (ContainerListener
l, ContainerListener oldl)
```

Parameters *l* An event listener.
oldl An event listener.
Returns A listener object that multicasts to `l` but not `oldl`.

```
public static FocusListener remove (FocusListener l,
FocusListener oldl)
```

Parameters *l* An event listener.
oldl An event listener.
Returns A listener object that multicasts to `l` but not `oldl`.

```
public static ItemListener remove (ItemListener l,
ItemListener oldl)
```

Parameters *l* An event listener.
oldl An event listener.
Returns A listener object that multicasts to `l` but not `oldl`.

```
public static KeyListener remove (KeyListener l,
KeyListener oldl)
```

Parameters *l* An event listener.
oldl An event listener.
Returns A listener object that multicasts to `l` but not `oldl`.

```
public static MouseListener remove (MouseListener l,
MouseListener oldl)
```

Parameters *l* An event listener.
oldl An event listener.
Returns A listener object that multicasts to `l` but not `oldl`.

```
public static MouseMotionListener remove
(MouseMotionListener l, MouseMotionListener oldl)
```

Parameters *l* An event listener.
oldl An event listener.
Returns A listener object that multicasts to `l` but not `oldl`.

```
public static TextListener remove (TextListener l,
TextListener oldl)
```

Parameters	*l*	An event listener.
	oldl	An event listener.
Returns	A listener object that multicasts to `l` but not `oldl`.	

```
public static WindowListener remove (WindowListener l,
WindowListener oldl)
```

Parameters	*l*	An event listener.
	oldl	An event listener.
Returns	A listener object that multicasts to `l` but not `oldl`.	

```
public static WindowListener remove (WindowListener l,
WindowListener oldl)
```

Parameters	*l*	An event listener.
	oldl	An event listener.
Returns	A listener object that multicasts to `l` but not `oldl`.	

removeInternal

```
public static EventListener removeInternal (EventListener
l, EventListener oldl)
```

Parameters	*l*	An event listener.
	oldl	An event listener.
Returns	A listener object that multicasts to `l` but not `oldl`.	
Description	This method is a helper for the `remove()` methods.	

Instance Methods

actionPerformed

```
public void actionPerformed (ActionEvent e)
```

Parameters	*e*	The action event that occurred.
Description	Handles the event by passing it on to listeners `a` and `b`.	

adjustmentValueChanged

```
public void adjustmentValueChanged (AdjustmentEvent e)
```

Parameters	*e*	The adjustment event that occurred.
Description	Handles the event by passing it on to listeners `a` and `b`.	

componentAdded

```
public void componentAdded (ContainerEvent e)
```

Parameters *e* The container event that occurred.
Description Handles the event by passing it on to listeners `a` and `b`.

componentHidden

```
public void componentHidden (ComponentEvent e)
```

Parameters *e* The component event that occurred.
Description Handles the event by passing it on to listeners `a` and `b`.

componentMoved

```
public void componentMoved (ComponentEvent e)
```

Parameters *e* The component event that occurred.
Description Handles the event by passing it on to listeners `a` and `b`.

componentRemoved

```
public void componentRemoved (ContainerEvent e)
```

Parameters *e* The container event that occurred.
Description Handles the event by passing it on to listeners `a` and `b`.

componentResized

```
public void componentResized (ComponentEvent e)
```

Parameters *e* The component event that occurred.
Description Handles the event by passing it on to listeners `a` and `b`.

componentShown

```
public void componentShown (ComponentEvent e)
```

Parameters *e* The component event that occurred.
Description Handles the event by passing it on to listeners `a` and `b`.

focusGained

```
public void focusGained (FocusEvent e)
```

Parameters *e* The focus event that occurred.
Description Handles the event by passing it on to listeners `a` and `b`.

focusLost

```
public void focusLost (FocusEvent e)
```

Parameters *e* The focus event that occurred.
Description Handles the event by passing it on to listeners `a` and `b`.

itemStateChanged

```
public void itemStateChanged (ItemEvent e)
```

Parameters *e* The item event that occurred.
Description Handles the event by passing it on to listeners `a` and `b`.

keyPressed

```
public void keyPressed (KeyEvent e)
```

Parameters *e* The key event that occurred.
Description Handles the event by passing it on to listeners `a` and `b`.

keyReleased

```
public void keyReleased (KeyEvent e)
```

Parameters *e* The key event that occurred.
Description Handles the event by passing it on to listeners `a` and `b`.

keyTyped

```
public void keyTyped (KeyEvent e)
```

Parameters *e* The key event that occurred.
Description Handles the event by passing it on to listeners `a` and `b`.

mouseClicked

```
public void mouseClicked (MouseEvent e)
```

Parameters *e* The mouse event that occurred.
Description Handles the event by passing it on to listeners `a` and `b`.

mouseDragged

```
public void mouseDragged (MouseEvent e)
```

Parameters *e* The mouse event that occurred.
Description Handles the event by passing it on to listeners `a` and `b`.

mouseEntered

```
public void mouseEntered (MouseEvent e)
```

Parameters *e* The mouse event that occurred.
Description Handles the event by passing it on to listeners `a` and `b`.

mouseExited

```
public void mouseExited (MouseEvent e)
```

Parameters *e* The mouse event that occurred.
Description Handles the event by passing it on to listeners `a` and `b`.

mouseMoved

```
public void mouseMoved (MouseEvent e)
```

Parameters *e* The mouse event that occurred.
Description Handles the event by passing it on to listeners `a` and `b`.

mousePressed

```
public void mousePressed (MouseEvent e)
```

Parameters *e* The mouse event that occurred.
Description Handles the event by passing it on to listeners `a` and `b`.

mouseReleased

```
public void mouseReleased (MouseEvent e)
```

Parameters *e* The mouse event that occurred.
Description Handles the event by passing it on to listeners `a` and `b`.

textValueChanged

```
public void textValueChanged (TextEvent e)
```

Parameters *e* The text event that occurred.
Description Handles the event by passing it on to listeners `a` and `b`.

windowActivated

```
public void windowActivated (WindowEvent e)
```

Parameters *e* The window event that occurred.
Description Handles the event by passing it on to listeners `a` and `b`.

windowClosed

```
public void windowClosed (WindowEvent e)
```

Parameters *e* The window event that occurred.
Description Handles the event by passing it on to listeners `a` and `b`.

windowClosing

```
public void windowClosing (WindowEvent e)
```

Parameters *e* The window event that occurred.
Description Handles the event by passing it on to listeners `a` and `b`.

windowDeactivated

```
public void windowDeactivated (WindowEvent e)
```

Parameters *e* The window event that occurred.
Description Handles the event by passing it on to listeners `a` and `b`.

windowDeiconified

```
public void windowDeiconified (WindowEvent e)
```

Parameters *e* The window event that occurred.
Description Handles the event by passing it on to listeners `a` and `b`.

windowIconified

```
public void windowIconified (WindowEvent e)
```

Parameters *e* The window event that occurred.
Description Handles the event by passing it on to listeners `a` and `b`.

windowOpened

```
public void windowOpened (WindowEvent e)
```

Parameters *e* The window event that occurred.
Description Handles the event by passing it on to listeners `a` and `b`.

Protected Instance Methods

remove

```
protected EventListener remove(EventListener oldl)
```

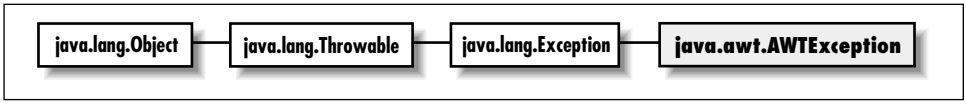
Parameters *oldl* The listener to remove.
Returns The resulting `EventListener`.

Description This method removes `oldl` from the `AWTEventMulticaster` and returns the resulting listener.

See Also

`ActionEvent`, `AdjustmentEvent`, `ComponentEvent`, `Event`, `EventListener`, `EventObject`, `FocusEvent`, `ItemEvent`, `KeyEvent`, `MouseEvent`, `WindowEvent`

19.4 AWTException



Description

An `AWTException`; thrown to indicate an exceptional condition; must be caught or declared in a `throws` clause.

Class Definition

```
public class java.awt.AWTException
    extends java.lang.Exception {

  // Constructors
  public AWTException (String message);
}
```

Constructors

`AWTException`

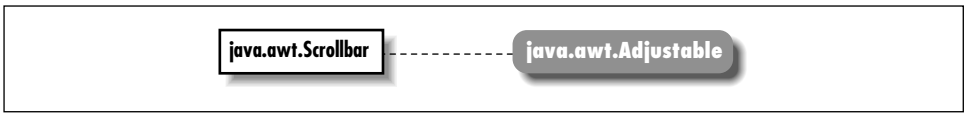
```
public AWTException (String message)
```

Parameters *message* Detailed message.

See Also

`Exception`, `String`

19.5 Adjustable ★



Description

The `Adjustable` interface is useful for scrollbars, sliders, dials, and other components that have an adjustable numeric value. Classes that implement the `Adjustable` interface should send `AdjustmentEvent` objects to listeners that have registered via `addAdjustmentListener(AdjustmentListener)`.

Interface Definition

```
public abstract interface java.awt.Adjustable {

  // Constants
  public final static int HORIZONTAL = 0;
  public final static int VERTICAL = 1;

  // Interface Methods
  public abstract void addAdjustmentListener (AdjustmentListener l);
  public abstract int getBlockIncrement();
  public abstract int getMaximum();
  public abstract int getMinimum();
  public abstract int getOrientation();
  public abstract int getUnitIncrement();
  public abstract int getValue();
  public abstract int getVisibleAmount();
  public abstract void removeAdjustmentListener (AdjustmentListener l);
  public abstract void setBlockIncrement (int b);
  public abstract void setMaximum (int max);
  public abstract void setMinimum (int min);
  public abstract void setUnitIncrement (int u);
  public abstract void setValue (int v);
  public abstract void setVisibleAmount (int v);
}
```

Constants

HORIZONTAL

```
public static final int HORIZONTAL
```

A constant representing horizontal orientation.

VERTICAL

```
public static final int VERTICAL
```

A constant representing vertical orientation.

Interface Methods

addAdjustmentListener

```
public abstract void addAdjustmentListener (ActionListener
l)
```

Parameters *l* An object that implements the `AdjustmentListener` interface.

Description Add a listener for adjustment event.

getBlockIncrement

```
public abstract int getBlockIncrement()
```

Returns The amount to scroll when a paging area is selected.

getMaximum

```
public abstract int getMaximum()
```

Returns The maximum value that the `Adjustable` object can take.

getMinimum

```
public abstract int getMinimum()
```

Returns The minimum value that the `Adjustable` object can take.

getOrientation

```
public abstract int getOrientation()
```

Returns A value representing the direction of the `Adjustable` object.

getUnitIncrement

```
public abstract int getUnitIncrement()
```

Returns The unit amount to scroll.

getValue

```
public abstract int getValue()
```

Returns The current setting for the `Adjustable` object.

getVisibleAmount

```
public abstract int getVisibleAmount()
```

Returns The current visible setting (i.e., size) for the `Adjustable` object.

removeAdjustmentListener

```
public abstract void removeAdjustmentListener
(AdjustmentListener l)
```

Parameters *l* One of the object's `AdjustmentListeners`.
Description Remove an adjustment event listener.

setBlockIncrement

```
public abstract void setBlockIncrement (int b)
```

Parameters *b* New block increment amount.
Description Changes the block increment amount for the `Adjustable` object.

setMaximum

```
public abstract void setMaximum (int max)
```

Parameters *max* New maximum value.
Description Changes the maximum value for the `Adjustable` object.

setMinimum

```
public abstract void setMinimum (int min)
```

Parameters *min* New minimum value.
Description Changes the minimum value for the `Adjustable` object.

setUnitIncrement

```
public abstract void setUnitIncrement (int u)
```

Parameters *u* New unit increment amount.
Description Changes the unit increment amount for the `Adjustable` object.

setValue

```
public abstract void setValue (int v)
```

Parameters	*v*	New value.
Description	Changes the current value of the `Adjustable` object.	

setVisibleAmount

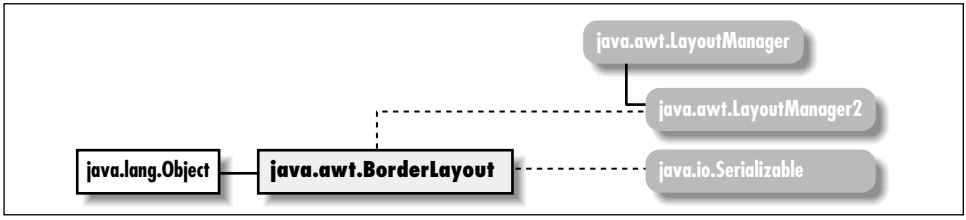
```
public abstract void setVisibleAmount (int v)
```

Parameters	*v*	New amount visible.
Description	Changes the current visible amount of the `Adjustable` object.	

See Also

`AdjustmentEvent`, `AdjustmentListener`, `Scrollbar`

19.6 BorderLayout



Description

`BorderLayout` is a `LayoutManager` that provides the means to lay out components along the edges of a container. It divides the container into five regions, named North, East, South, West, and Center. Normally you won't call the `LayoutManager`'s methods yourself. When you `add()` a `Component` to a `Container`, the `Container` calls the `addLayoutComponent()` method of its `LayoutManager`.

Class Definition

```
public class java.awt.BorderLayout
    extends java.lang.Object
    implements java.awt.LayoutManager2, java.io.Serializable {

  // Constants
  public final static String CENTER; ★
  public final static String EAST; ★
  public final static String NORTH; ★
  public final static String SOUTH; ★
  public final static String WEST; ★
```

```
    // Constructors
    public BorderLayout();
    public BorderLayout (int hgap, int vgap);

    // Instance Methods
    public void addLayoutComponent (Component comp, Object constraints); ★
    public void addLayoutComponent (String name, Component component); ☆
    public int getHgap(); ★
    public abstract float getLayoutAlignmentX(Container target); ★
    public abstract float getLayoutAlignmentY(Container target); ★
    public int getVgap(); ★
    public abstract void invalidateLayout(Container target); ★

    public void layoutContainer (Container target);
    public abstract Dimension maximumLayoutSize(Container target); ★
    public Dimension minimumLayoutSize (Container target);
    public Dimension preferredLayoutSize (Container target);
    public void removeLayoutComponent (Component component);
    public void setHgap (int hgap); ★
    public void setVgap (int vgap); ★
    public String toString();
}
```

Constants

CENTER

```
public final static String CENTER
```

A constant representing center orientation.

EAST

```
public final static String EAST
```

A constant representing east orientation.

NORTH

```
public final static String NORTH
```

A constant representing north orientation.

SOUTH

```
public final static String SOUTH
```

A constant representing south orientation.

WEST

```
public final static String WEST
```

A constant representing west orientation.

Constructors

BorderLayout

```
public BorderLayout()
```

Description Constructs a `BorderLayout` object.

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

Parameters *hgap* Horizontal space between each component in the container.

vgap Vertical space between each component in the container.

Description Constructs a `BorderLayout` object with the values specified as the gaps between each component in the container managed by this instance of `BorderLayout`.

Instance Methods

addLayoutComponent

```
public void addLayoutComponent (Component comp,
Object constraints) ★
```

Parameters *comp* The component being added.

constraints An object describing the constraints on this component.

Implements `LayoutManager2.addLayoutComponent()`

Description Adds the component `comp` to a container subject to the given `constraints`. This is a more general version of `addLayoutComponent(String, Component)` method. It corresponds to `java.awt.Container`'s `add(Component, Object)` method. In practice, it is used the same in version 1.1 as in Java 1.0.2, except with the parameters swapped:

```
Panel p = new Panel(new BorderLayout());
p.add(new Button("OK"), BorderLayout.SOUTH);
```

addLayoutComponent

```
public void addLayoutComponent (String name,
Component component) ☆
```

Parameters	*name*	Name of region to add component to.
	component	Actual component being added.
Implements	`LayoutManager.addLayoutComponent()`	
Description	Adds a `component` to a container in region `name`. This has been replaced in version 1.1 with the more general `addLayoutComponent(Component, Object)`.	

getHgap

```
public int getHgap() ★
```

Returns	The horizontal gap for this `BorderLayout` instance.

getLayoutAlignmentX

```
public abstract float getLayoutAlignmentX (Container
target) ★
```

Parameters	*target*	The container to inspect.
Returns	The value .5 for all containers.	
Description	This method returns the preferred alignment of the given container `target`. A return value of 0 is left aligned, .5 is centered, and 1 is right aligned.	

getLayoutAlignmentY

```
public abstract float getLayoutAlignmentY (Container
target) ★
```

Parameters	*target*	The container to inspect.
Returns	The value .5 for all containers.	
Description	This method returns the preferred alignment of the given container `target`. A return value of 0 is top aligned, .5 is centered, and 1 is bottom aligned.	

getVgap

```
public int getVgap() ★
```

Returns	The vertical gap for this `BorderLayout` instance.

invalidateLayout

```
public abstract void invalidateLayout (Container target)
```
★

Parameters	*target*	The container to invalidate.
Description	Does nothing.	

layoutContainer

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

Parameters	*target*	The container that needs to be redrawn.
Implements	`LayoutManager.layoutContainer()`	
Description	Draws components contained within `target`.	

maximumLayoutSize

```
public abstract Dimension maximumLayoutSize (Container
target)
```
★

Parameters	*target*	The container to inspect.
	Returns	A `Dimension` whose horizontal and vertical components are `Integer.MAX_VALUE`.
Description	For `BorderLayout`, a maximal `Dimension` is always returned.	

minimumLayoutSize

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

Parameters	*target*	The container whose size needs to be calculated.
Returns	Minimum `Dimension` of the container `target`.	
Implements	`LayoutManager.minimumLayoutSize()`	
Description	Calculates minimum size of `target`. container.	

preferredLayoutSize

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

Parameters	*target*	The container whose size needs to be calculated.
Returns	Preferred `Dimension` of the container `target`.	
Implements	`LayoutManager.preferredLayoutSize()`	
Description	Calculates preferred size of `target` container.	

removeLayoutComponent

```
public void removeLayoutComponent (Component component)
```

Parameters	*component*	Component to stop tracking.
Implements	`LayoutManager.removeLayoutComponent()`	
Description	Removes `component` from any internal tracking systems.	

`setHgap`

```
public void setHgap (int hgap) ★
```

Parameters	*hgap*	The horizontal gap value.
Description	Sets the horizontal gap between components.	

`setVgap`

```
public void setVgap (int vgap) ★
```

Parameters	*vgap*	The vertical gap value.
Description	Sets the vertical gap between components.	

`toString`

```
public String toString()
```

Returns	A string representation of the `BorderLayout` object.
Overrides	`Object.toString()`

See Also

`Component`, `Container`, `Dimension`, `LayoutManager`, `LayoutManager2`, `Object`, `String`

19.7 Button

Description

The `Button` is the familiar labeled button object. It inherits most of its functionality from `Component`. For example, to change the font of the `Button`, you would use `Component`'s `setFont()` method. The `Button` sends `java.awt.event.ActionEvent` objects to its listeners when it is pressed.

Class Definition

```
public class java.awt.Button
    extends java.awt.Component {

  // Constructors
  public Button();
  public Button (String label);

  // Instance Methods
  public void addActionListener (ActionListener l); ★
  public void addNotify();
  public String getActionCommand(); ★
  public String getLabel();
  public void removeActionListener (ActionListener l); ★
  public void setActionCommand (String command); ★
  public synchronized void setLabel (String label);

  // Protected Instance Methods
  protected String paramString();
  protected void processActionEvent (ActionEvent e); ★
  protected void processEvent (AWTEvent e); ★
}
```

Constructors

Button

```
public Button()
```

Description Constructs a `Button` object with no label.

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

Parameters *label* The text for the label on the button
Description Constructs a `Button` object with text of `label`.

Instance Methods

addActionListener

```
public void addActionListener (ActionListener l) ★
```

Parameters *l* An object that implements the `ActionListener` interface.
Description Add a listener for the action event.

addNotify

```
public void addNotify()
```

Overrides `Component.addNotify()`
Description Creates `Button`'s peer.

getActionCommand

```
public String getActionCommand() ★
```

Returns Current action command string.
Description Returns the string used for the action command.

getLabel

```
public String getLabel()
```

Returns Text of the `Button`'s label.

removeActionListener

```
public void removeActionListener (ActionListener l) ★
```

Parameters *l* One of this `Button`'s `ActionListeners`.
Description Remove an action event listener.

setActionCommand

```
public void setActionCommand (String command) ★
```

Parameters *command* New action command string.
Description Specify the string used for the action command.

setLabel

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

Parameters *label* New text for label of `Button`.
Description Changes the `Button`'s label to `label`.

Protected Instance Methods

paramString

```
protected String paramString()
```

Returns String with current settings of `Button`.
Overrides `Component.paramString()`
Description Helper method for `toString()` used to generate a string of current settings.

processActionEvent

```
protected void processActionEvent (ActionEvent e) ★
```

Parameters *e* The action event to process.
Description Action events are passed to this method for processing. Normally, this method is called by `processEvent()`.

processEvent

```
protected void processEvent (AWTEvent e) ★
```

Parameters *e* The event to process.
Description Low level `AWTEvents` are passed to this method for processing.

See Also

`ActionListener`, `Component`, `String`

19.8 Canvas

Description

`Canvas` is a `Component` that provides a drawing area and is often used as a base class for new components.

Class Definition

```
public class java.awt.Canvas
    extends java.awt.Component {

  // Constructors
  public Canvas();

  // Instance Methods
  public void addNotify();
  public void paint (Graphics g);
}
```

Constructors

`Canvas`

```
public Canvas()
```

Description Constructs a `Canvas` object.

Instance Methods

`addNotify`

```
public void addNotify()
```

Overrides `Component.addNotify()`
Description Creates `Canvas`'s peer.

`paint`

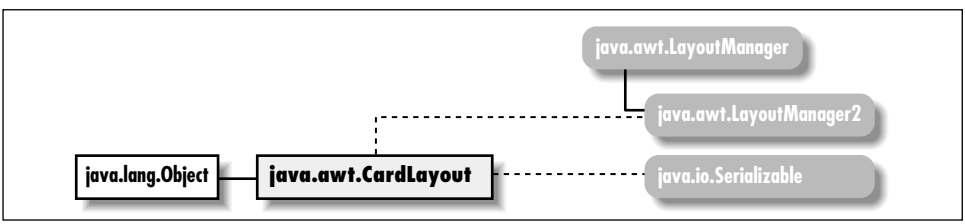
```
public void paint (Graphics g)
```

Parameters *g* Graphics context of component.
Description Empty method to be overridden in order to draw something in graphics context.

See Also

`Component`, `Graphics`

19.9 CardLayout



Description

The `CardLayout` `LayoutManager` provides the means to manage multiple components, displaying one at a time. Components are displayed in the order in which they are added to the layout, or in an arbitrary order by using an assignable name.

Class Definition

```
public class java.awt.CardLayout
    extends java.lang.Object
    implements java.awt.LayoutManager2, java.io.Serializable {

  // Constructors
  public CardLayout();
  public CardLayout (int hgap, int vgap);

  // Instance Methods
  public void addLayoutComponent (Component comp,
      Object constraints); ★
  public void addLayoutComponent (String name, Component component); ☆
  public void first (Container parent);
  public int getHgap(); ★
  public abstract float getLayoutAlignmentX(Container target); ★
  public abstract float getLayoutAlignmentY(Container target); ★
  public int getVgap(); ★
  public abstract void invalidateLayout(Container target); ★
  public void last (Container parent);
  public void layoutContainer (Container target);
  public abstract Dimension maximumLayoutSize(Container target); ★
  public Dimension minimumLayoutSize (Container target);
  public void next (Container parent);
  public Dimension preferredLayoutSize (Container target);
  public void previous (Container parent);
  public void removeLayoutComponent (Component component);
  public void setHgap (int hgap); ★
  public void setVgap (int vgap); ★
```

```
  public void show (Container parent, String name);
  public String toString();
}
```

Constructors

CardLayout

```
public CardLayout()
```

Description Constructs a `CardLayout` object.

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

Parameters	*hgap*	Horizontal space around left and right of container
	vgap	Vertical space around top and bottom of container

Description Constructs a `CardLayout` object with the values specified as the gaps around the container managed by this instance of `CardLayout`.

Instance Methods

addLayoutComponent

```
public void addLayoutComponent (Component comp,
Object constraints) ★
```

Parameters	*comp*	The component being added.
	constraints	An object describing the constraints on this component.

Implements `LayoutManager2.addLayoutComponent()`

Description Adds the component `comp` to a container subject to the given `constraints`. This is a more generalized version of `addLayoutComponent(String, Component)`. It corresponds to `java.awt.Container's add(Component, Object)`. In practice, it is used the same in Java 1.1 as in Java 1.0.2, except with the parameters swapped:

```
Panel p = new Panel();
p.setLayoutManager(new CardLayout());
p.add(new Button("OK"), "Don Julio");
```

addLayoutComponent

```
public void addLayoutComponent (String name,
Component component) ☆
```

Parameters	*name*	Name of the component to add.
	component	The actual component being added.
Implements	`LayoutManager.addLayoutComponent()`	
Description	Places `component` under the layout's management, assigning it the given `name`. This has been replaced in version 1.1 with the more general `addLayoutComponent(Component, Object)`.	

first

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

Parameters	*parent*	The container whose displayed component is changing.
Throws	`IllegalArgumentException`	If the `LayoutManager` of `parent` is not `CardLayout`.
Description	Sets the container to display the first component in parent.	

getHgap

```
public int getHgap() ★
```

Returns	The horizontal gap for this `CardLayout` instance.

getLayoutAlignmentX

```
public abstract float getLayoutAlignmentX (Container
target) ★
```

Parameters	*target*	The container to inspect.
Returns	The value .5 for all containers.	
Description	This method returns the preferred alignment of the given container `target`. A return value of 0 is left aligned, .5 is centered, and 1 is right aligned.	

getLayoutAlignmentY

```
public abstract float getLayoutAlignmentY (Container
target) ★
```

Parameters	*target*	The container to inspect.
Returns	The value .5 for all containers.	

Description This method returns the preferred alignment of the given container `target`. A return value of 0 is top aligned, .5 is centered, and 1 is bottom aligned.

getVgap

```
public int getVgap() ★
```

Returns The vertical gap for this `CardLayout` instance.

invalidateLayout

```
public abstract void invalidateLayout (Container target)
★
```

Parameters *target* The container to invalidate.
Description Does nothing.

last

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

Parameters *parent* The container whose displayed component is changing.
Throws `IllegalArgumentException` If the `LayoutManager` of `parent` is not `CardLayout`.
Description Sets the container to display the final component in parent.

layoutContainer

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

Parameters *target* The container that needs to be redrawn.
Implements `LayoutManager.layoutContainer()`
Description Displays the currently selected component contained within `target`.

maximumLayoutSize

```
public abstract Dimension maximumLayoutSize .hw Container
(Container target) ★
```

Parameters *target* The container to inspect.
Returns A `Dimension` whose horizontal and vertical components are `Integer.MAX_VALUE`.
Description For `CardLayout`, a maximal `Dimension` is always returned.

minimumLayoutSize

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

Parameters	*target*	The container whose size needs to be calculated.
Returns	Minimum `Dimension` of the container `target`.	
Implements	`LayoutManager.minimumLayoutSize()`	
Description	Calculates minimum size of the `target` container.	

next

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

Parameters	*parent*	The container whose displayed component is changing.
Throws	`IllegalArgumentException`	If the `LayoutManager` of `parent` is not `CardLayout`.
Description	Sets the container to display the following component in the parent.	

preferredLayoutSize

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

Parameters	*target*	The container whose size needs to be calculated.
Returns	Preferred `Dimension` of the container `target`.	
Implements	`LayoutManager.preferredLayoutSize()`	
Description	Calculates preferred size of the `target` container.	

previous

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

Parameters	*parent*	The container whose displayed component is changing.
Throws	`IllegalArgumentException`	If the `LayoutManager` of `parent` is not `CardLayout`.
Description	Sets the container to display the prior component in `parent`.	

removeLayoutComponent

```
public void removeLayoutComponent (Component component)
```

Parameters	*component*	Component to stop tracking.

Implements	`LayoutManager.removeLayoutComponent()`
Description	Removes `component` from the layout manager's internal tables.

`setHgap`

```
public void setHgap (int hgap) ★
```

Parameters	*hgap*	The horizontal gap value.
Description	Sets the horizontal gap for the left and right of the container.	

`setVgap`

```
public void setVgap (int vgap) ★
```

Parameters	*vgap*	The vertical gap value.
Description	Sets the vertical gap for the top and bottom of the container.	

`show`

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

Parameters	*parent*	The container whose displayed component is changing.
	name	Name of component to display.
Throws	`IllegalArgumentException`	If `LayoutManager` of `parent` is not `CardLayout`.
Description	Sets the container to display the component `name` in `parent`.	

`toString`

```
public String toString()
```

Returns	A string representation of the `CardLayout` object.
Overrides	`Object.toString()`

See Also

`Component`, `Container`, `Dimension`, `LayoutManager`, `LayoutManager2`, `Object`, `String`

19.10 Checkbox

Description

The Checkbox is a Component that provides a true or false toggle switch for user input.

Class Definition

```
public class java.awt.Checkbox
    extends java.awt.Component
    implements java.awt.ItemSelectable {

  // Constructors
  public Checkbox();
  public Checkbox (String label);
  public Checkbox (String label, boolean state); ★
  public Checkbox (String label, boolean state, CheckboxGroup group); ★
  public Checkbox (String label, CheckboxGroup group, boolean state);

  // Instance Methods
  public void addItemListener (ItemListener l); ★
  public void addNotify();
  public CheckboxGroup getCheckboxGroup();
  public String getLabel();
  public Object[] getSelectedObjects(); ★
  public boolean getState();
  public void removeItemListener (ItemListener l); ★
  public void setCheckboxGroup (CheckboxGroup group);
  public synchronized void setLabel (String label);
  public void setState (boolean state);

  // Protected Instance Methods
  protected String paramString();
  protected void processEvent (AWTEvent e); ★
  protected void processItemEvent (ItemEvent e); ★
}
```

Constructors

Checkbox

```
public Checkbox()
```

Description Constructs a `Checkbox` object with no label that is initially `false`.

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

Parameters *label* Text to display with the `Checkbox`.
Description Constructs a `Checkbox` object with the given `label` that is initially `false`.

```
public Checkbox (String label, boolean state) ★
```

Parameters *label* Text to display with the `Checkbox`.
state Intial value of the `Checkbox`.
Description Constructs a `Checkbox` with the given `label`, initialized to the given `state`.

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

Parameters *label* Text to display with the `Checkbox`.
state Intial value of the `Checkbox`.
group The `CheckboxGroup` this `Checkbox` should belong to.
Description Constructs a `Checkbox` with the given `label`, initialized to the given `state` and belonging to `group`.

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

Parameters *label* Text to display with the `Checkbox`.
group The `CheckboxGroup` this `Checkbox` should belong to.
state Intial value of the `Checkbox`.
Description Constructs a `Checkbox` object with the given settings.

Instance Methods

`addItemListener`

```
public void addItemListener (ItemListener l) ★
```

Parameters *l* The listener to be added.
Implements `ItemSelectable.addItemListener(ItemListener l)`
Description Adds a listener for the `ItemEvent` objects this `Checkbox` generates.

addNotify

```
public void addNotify()
```

Overrides `Component.addNotify()`
Description Creates `Checkbox` peer.

getCheckboxGroup

```
public CheckboxGroup getCheckboxGroup()
```

Returns The current `CheckboxGroup` associated with the `Checkbox`, if any.

getLabel

```
public String getLabel()
```

Returns The text associated with the `Checkbox`.

getSelectedObjects

```
public Object[] getSelectedObjects() ★
```

Implements `ItemSelectable.getSelectedObjects()`
Description If the `Checkbox` is checked, returns an array with length 1 containing the label of the `Checkbox`; otherwise returns `null`.

getState

```
public boolean getState()
```

Returns The current state of the `Checkbox`.

removeItemListener

```
public void removeItemListener (ItemListener l) ★
```

Parameters *l* The listener to be removed.
Implements `ItemSelectable.removeItemListener (ItemListener l)`
Description Removes the specified `ItemListener` so it will not receive `ItemEvent` objects from this `Checkbox`.

setCheckboxGroup

```
public void setCheckboxGroup (CheckboxGroup group)
```

Parameters *group* New group in which to place the `Checkbox`.
Description Associates the `Checkbox` with a different `CheckboxGroup`.

setLabel

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

Parameters	*label*	New text to associate with `Checkbox`.
Description	Changes the text associated with the `Checkbox`.	

setState

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

Parameters	*state*	New state for the `Checkbox`.
Description	Changes the state of the `Checkbox`.	

Protected Instance Methods

paramString

```
protected String paramString()
```

Returns	String with current settings of `Checkbox`.
Overrides	`Component.paramString()`
Description	Helper method for `toString()` to generate string of current settings.

processEvent

```
protected void processEvent(AWTEvent e) ★
```

Parameters	*e*	The event to process.
Description	Low level `AWTEvents` are passed to this method for processing.	

processItemEvent

```
protected void processItemEvent(ItemEvent e) ★
```

Parameters	*e*	The item event to process.
Description	Item events are passed to this method for processing. Normally, this method is called by `processEvent()`.	

See Also

`CheckboxGroup`, `Component`, `ItemEvent`, `ItemSelectable`, `String`

19.11 CheckboxGroup

Description

The `CheckboxGroup` class provides the means to group multiple `Checkbox` items into a mutual exclusion set, so that only one checkbox in the set has the value `true` at any time. The checkbox with the value `true` is the currently selected checkbox. Mutually exclusive checkboxes usually have a different appearance from regular checkboxes and are also called "radio buttons."

Class Definition

```
public class java.awt.CheckboxGroup
    extends java.lang.Object
    implements java.io.Serializable {

  // Constructors
  public CheckboxGroup();

  // Instance Methods
  public Checkbox getCurrent(); ☆
  public Checkbox getSelectedCheckbox() ★
  public synchronized void setCurrent (Checkbox checkbox); ☆
  public synchronized void setSelectedCheckbox (Checkbox checkbox); ★
  public String toString();
}
```

Constructors

CheckboxGroup

```
public CheckboxGroup()
```

Description Constructs a `CheckboxGroup` object.

Instance Methods

getCurrent

```
public Checkbox getCurrent() ☆
```

Returns The currently selected `Checkbox` within the `CheckboxGroup`.
Description Replaced by the more aptly named `getSelectedCheckbox()`.

getSelectedCheckbox

```
public Checkbox getSelectedCheckbox() ★
```

Returns The currently selected `Checkbox` within the `CheckboxGroup`.

setCurrent

```
public synchronized void setCurrent (Checkbox checkbox) ☆
```

Parameters *checkbox* The `Checkbox` to select.
Description Changes the currently selected `Checkbox` within the `CheckboxGroup`.
Description Replaced by `setSelectedCheckbox(Checkbox)`.

setSelectedCheckbox

```
public synchronized void setSelectedCheckbox (Checkbox
checkbox) ★
```

Parameters *checkbox* The `Checkbox` to select.
Description Changes the currently selected `Checkbox` within the `CheckboxGroup`.

toString

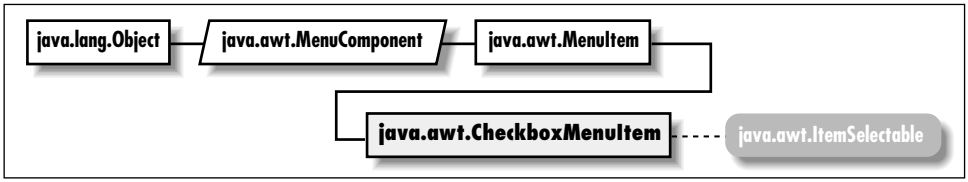
```
public String toString()
```

Returns A string representation of the `CheckboxGroup` object.
Overrides `Object.toString()`

See Also

`Checkbox`, `Object`, `String`

19.12 CheckboxMenuItem



Description

The `CheckboxMenuItem` class represents a menu item with a boolean state.

Class Definition

```
public class java.awt.CheckboxMenuItem
    extends java.awt.MenuItem
    implements java.awt.ItemSelectable {

  // Constructors
  public CheckboxMenuItem(); ★
  public CheckboxMenuItem (String label);
  public CheckboxMenuItem (String label, boolean state); ★

  // Instance Methods
  public void addItemListener (ItemListener l); ★
  public void addNotify();
  public Object[] getSelectedObjects(); ★
  public boolean getState();
  public String paramString();
  public void removeItemListener (ItemListener l); ★
  public synchronized void setState (boolean condition);

  // Protected Instance Methods
  protected void processEvent (AWTEvent e); ★
  protected void processItemEvent (ItemEvent e); ★
}
```

Constructors

CheckboxMenuItem

```
public CheckboxMenuItem() ★
```

Description Constructs a `CheckboxMenuItem` object with no label.

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

Parameters *label* Text that appears on `CheckboxMenuItem`.
Description Constructs a `CheckboxMenuItem` object whose value is initially `false`.

```
public CheckboxMenuItem (String label, boolean state) ★
```

Parameters *label* Text that appears on `CheckboxMenuItem`.
state The initial state of the menu item.
Description Constructs a `CheckboxMenuItem` object with the specified `label` and `state`.

Instance Methods

addItemListener

```
public void addItemListener (ItemListener l) ★
```

Parameters	*l*	The listener to be added.
Implements	`ItemSelectable.addItemListener(ItemListener l)`	
Description	Adds a listener for the `ItemEvent` objects this `CheckboxMenuItem` fires off.	

addNotify

```
public void addNotify()
```

Overrides	`MenuItem.addNotify()`
Description	Creates `CheckboxMenuItem`'s peer.

getSelectedObjects

```
public Object[] getSelectedObjects() ★
```

Implements	`ItemSelectable.getSelectedObjects()`
Description	If the `CheckboxMenuItem` is checked, returns an array with length `1` containing the label of the `CheckboxMenuItem`; otherwise returns `null`.

getState

```
public boolean getState()
```

Returns	The current state of the `CheckboxMenuItem`.

paramString

```
public String paramString()
```

Returns	A string with current settings of `CheckboxMenuItem`.
Overrides	`MenuItem.paramString()`
Description	Helper method for `toString()` to generate string of current settings.

removeItemListener

```
public void removeItemListener (ItemListener l) ★
```

Parameters	*l*	The listener to be removed.
Implements	`ItemSelectable.removeItemListener (ItemListener l)`	

Description Removes the specified `ItemListener` so it will not receive `ItemEvent` objects from this `CheckboxMenuItem`.

`setState`

```
public synchronized void setState (boolean condition)
```

Parameters *condition* New state for the `CheckboxMenuItem`.
Description Changes the state of the `CheckboxMenuItem`.

Protected Instance Methods

`processEvent`

```
protected void processEvent(AWTEvent e) ★
```

Parameters *e* The event to process.
Overrides `MenuItem.processEvent(AWTEvent)`
Description Low level `AWTEvents` are passed to this method for processing.

`processItemEvent`

```
protected void processItemEvent(ItemEvent e) ★
```

Parameters *e* The item event to process.
Description Item events are passed to this method for processing. Normally, this method is called by `processEvent()`.

See Also

`ItemEvent`, `ItemSelectable`, `MenuItem`, `String`

19.13 Choice

Description

The `Choice` is a `Component` that provides a drop-down list of choices to choose from.

Class Definition

```
public class java.awt.Choice
    extends java.awt.Component
    implements java.awt.ItemSelectable {
```

```
    // Constructors
    public Choice();

    // Instance Methods
    public synchronized void add (String item); ★
    public synchronized void addItem (String item); ☆
    public void addItemListener (ItemListener l); ★
    public void addNotify();
    public int countItems();  ☆
    public String getItem (int index);
    public int getItemCount(); ★
    public int getSelectedIndex();
    public synchronized String getSelectedItem();
    public synchronized Object[] getSelectedObjects(); ★
    public synchronized void insert (String item, int index); ★
    public synchronized void remove (int position); ★
    public synchronized void remove (String item); ★
    public synchronized void removeAll(); ★
    public void removeItemListener (ItemListener l); ★
    public synchronized void select (int pos);
    public synchronized void select (String str);

    // Protected Instance Methods
    protected String paramString();
    protected void processEvent (AWTEvent e);  ★
    protected void processItemEvent (ItemEvent e); ★
}
```

Constructors

`Choice`

```
public Choice()
```

Description Constructs a `Choice` object.

Instance Methods

`add`

```
public synchronized void add (String item) ★
```

Parameters	*item*	Text for new entry.
Throws	`NullPointerException`	If `item` is null.
Description	Adds a new entry to the available choices.	

addItem

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

Parameters	*item*	Text for new entry.
Throws	`NullPointerException`	If `item` is null.
Description	Replaced by `add(String)`.	

addItemListener

```
public void addItemListener (ItemListener l) ★
```

Parameters	*l*	The listener to be added.
Implements	`ItemSelectable.addItemListener(ItemListener l)`	
Description	Adds a listener for the `ItemEvent` objects this `Choice` generates.	

addNotify

```
public void addNotify()
```

Overrides	`Component.addNotify()`
Description	Creates `Choice`'s peer.

countItems

```
public int countItems() ☆
```

Returns	Number of items in the `Choice`.
Description	Replaced by `getItemCount()`.

getItem

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

Parameters	*index*	Position of entry.
Returns	A string for an entry at a given position.	
Throws	`ArrayIndexOutOfBoundsException`	If `index` is invalid; indices start at zero.

getItemCount

```
public int getItemCount() ★
```

Returns	Number of items in the `Choice`.

getSelectedIndex

```
public int getSelectedIndex()
```

Returns Position of currently selected entry.

getSelectedItem

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

Returns Currently selected entry as a String.

getSelectedObjects

```
public synchronized Object[] getSelectedObjects() ★
```

Implements `ItemSelectable.getSelectedObjects()`
Description A single-item array containing the current selection.

insert

```
public synchronized void insert (String item, int index)
★
```

Parameters *item* The string to add.
index The position for the new string.
Throws `IllegalArgumentException`
If `index` is less than zero.
Description Inserts `item` in the given position.

remove

```
public synchronized void remove (int position) ★
```

Parameters *position* The index of an entry in the `Choice` component.
Description Removes the entry in the given position.

```
public synchronized void remove (String string) ★
```

Parameters *string* Text of an entry within the `Choice` component.
Throws `IllegalArgumentException`
If `string` is not in the `Choice`.
Description Makes the first entry that matches `string` the selected item.

removeAll

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

Description Removes all the entries from the `Choice`.

`removeItemListener`

```
public void removeItemListener (ItemListener l) ★
```

Parameters *l* The listener to be removed.
Implements `ItemSelectable.removeItemListener (ItemListener l)`
Description Removes the specified `ItemListener` so it will not receive `ItemEvent` objects from this `Choice`.

`select`

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

Parameters *pos* The index of an entry in the `Choice` component.
Throws `IllegalArgumentException` If the position is not valid.
Description Makes the entry in the given position.

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

Parameters *str* Text of an entry within the `Choice` component.
Description Makes the first entry that matches `str` the selected item for the `Choice`.

Protected Instance Methods

`paramString`

```
protected String paramString()
```

Returns A string with current settings of `Choice`.
Overrides `Component.paramString()`
Description Helper method for `toString()` to generate string of current settings.

`processEvent`

```
protected void processEvent (AWTEvent e) ★
```

Parameters *e* The event to process.
Description Low level `AWTEvents` are passed to this method for processing.

processItemEvent

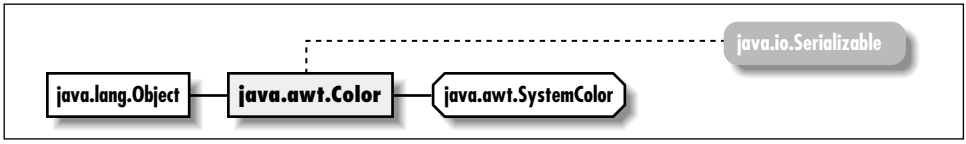
```
protected void processItemEvent (ItemEvent e) ★
```

Parameters	*e*	The item event to process.
Description	Item events are passed to this method for processing. Normally, this method is called by `processEvent()`.	

See Also

`Component`, `ItemSelectable`, `String`

19.14 Color



Description

The `Color` class represents a specific color to the system.

Class Definition

```
public final class java.awt.Color
    extends java.lang.Object
    implements java.io.Serializable {

  // Constants
  public static final Color black;
  public static final Color blue;
  public static final Color cyan;
  public static final Color darkGray;
  public static final Color gray;
  public static final Color green;
  public static final Color lightGray;
  public static final Color magenta;
  public static final Color orange;
  public static final Color pink;
  public static final Color red;
  public static final Color white;
  public static final Color yellow;

  // Constructors
  public Color (int rgb);
  public Color (int red, int green, int blue);
  public Color (float red, float green, float blue);
```

```
    // Class Methods
    public static Color decode (String name); ★
    public static Color getColor (String name);
    public static Color getColor (String name, Color defaultColor);
    public static Color getColor (String name, int defaultColor);
    public static Color getHSBColor (float hue, float saturation,
        float brightness);
    public static int HSBtoRGB (float hue, float saturation, float brightness);
    public static float[] RGBtoHSB (int red, int green, int blue,
        float hsbvalues[]);

    // Instance Methods
    public Color brighter();
    public Color darker();
    public boolean equals (Object object);
    public int getBlue();
    public int getGreen();
    public int getRed();
    public int getRGB();
    public int hashCode();
    public String toString();
}
```

Constants

black

```
public static final Color black
```

The color black.

blue

```
public static final Color blue
```

The color blue.

cyan

```
public static final Color cyan
```

The color cyan.

darkGray

```
public static final Color darkGray
```

The color dark gray.

gray

```
public static final Color gray
```

The color gray.

green

```
public static final Color green
```

The color green.

lightGray

```
public static final Color lightGray
```

The color light gray.

magenta

```
public static final Color magenta
```

The color magenta.

orange

```
public static final Color orange
```

The color orange.

pink

```
public static final Color pink
```

The color pink.

red

```
public static final Color red
```

The color red.

white

```
public static final Color white
```

The color white.

yellow

```
public static final Color yellow
```

The color yellow.

Constructors

Color

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

Parameters	*rgb*	Composite color value
Description	Constructs a `Color` object with the given `rgb` value.	

```
public Color (int red, int green, int blue)
```

Parameters	*red*	Red component of color in the range[0, 255]
	green	Green component of color in the range[0, 255]
	blue	Blue component of color in the range[0, 255]
Description	Constructs a `Color` object with the given `red`, `green`, and `blue` values.	

```
public Color (float red, float green, float blue)
```

Parameters	*red*	Red component of color in the range[0.0, 1.0]
	green	Green component of color in the range[0.0, 1.0]
	blue	Blue component of color in the range[0.0, 1.0]
Description	Constructs a `Color` object with the given `red`, `green`, and `blue` values.	

Class Methods

decode

```
public static Color decode (String nm) ★
```

Parameters	*nm*	A `String` representing a color as a 24-bit integer.
Returns	The color requested.	
Throws	`NumberFormatException`	If `nm` cannot be converted to a number.
Description	Gets color specified by the given string.	

getColor

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

Parameters	*name*	The name of a system property indicating which color to fetch.
Returns	Color instance of `name` requested, or `null` if the `name` is invalid.	
Description	Gets color specified by the system property name.	

```
public static Color getColor (String name, Color
defaultColor)
```

Parameters	*name*	The `name` of a system property indicating which color to fetch.
	defaultColor	Color to return if `name` is not found in properties, or invalid.
Returns	Color instance of `name` requested, or `defaultColor` if the `name` is invalid.	
Description	Gets color specified by the system property `name`.	

```
public static Color getColor (String name, int
defaultColor)
```

Parameters	*name*	The `name` of a system property indicating which color to fetch.
	defaultColor	Color to return if `name` is not found in properties, or invalid.
Returns	Color instance of `name` requested, or `defaultColor` if the `name` is invalid.	
Description	Gets color specified by the system property `name`. The default color is specified as a 32-bit RGB value.	

getHSBColor

```
public static Color getHSBColor (float hue, float
saturation, float brightness)
```

Parameters	*hue*	Hue component of `Color` to create, in the range[0.0, 1.0].
	saturation	Saturation component of `Color` to create, in the range[0.0, 1.0].
	brightness	Brightness component of `Color` to create, in the range[0.0, 1.0].
Returns	Color instance for values provided.	
Description	Create an instance of `Color` by using hue, saturation, and brightness instead of red, green, and blue values.	

HSBtoRGB

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

Parameters	*hue*	Hue component of `Color` to convert, in the range[0.0, 1.0].

	saturation	Saturation component of `Color` to convert, in the range[0.0, 1.0].
	brightness	Brightness component of `Color` to convert, in the range[0.0, 1.0].
Returns	Color value for hue, saturation, and brightness provided.	
Description	Converts a specific hue, saturation, and brightness to a `Color` and returns the red, green, and blue values in a composite integer value.	

RGBtoHSB

```
public static float[] RGBtoHSB (int red, int green, int
blue, float[] hsbvalues)
```

Parameters	*red*	Red component of `Color` to convert, in the range[0, 255].
	green	Green component of `Color` to convert, in the range[0, 255].
	blue	Blue component of `Color` to convert, in the range[0, 255].
	hsbvalues	Three element array in which to put the result. This array is used as the method's return object. If `null`, a new array is allocated.
Returns	Hue, saturation, and brightness values for `Color` provided, in elements 0, 1, and 2 (respectively) of the returned array.	
Description	Allows you to convert specific red, green, blue value to the hue, saturation, and brightness equivalent.	

Instance Methods

brighter

```
public Color brighter()
```

Returns	Brighter version of current color.
Description	Creates new `Color` that is somewhat brighter than current.

darker

```
public Color darker()
```

Returns	Darker version of current color.
Description	Creates new `Color` that is somewhat darker than current.

equals

```
public boolean equals (Object object)
```

Parameters	*object*	The object to compare.
Returns	`true` if `object` represents the same color, `false` otherwise.	
Overrides	`Object.equals(Object)`	
Description	Compares two different `Color` instances for equivalence.	

getBlue

```
public int getBlue()
```

Returns	Blue component of current color.

getGreen

```
public int getGreen()
```

Returns	Green component of current color.

getRed

```
public int getRed()
```

Returns	Red component of current color.

getRGB

```
public int getRGB()
```

Returns	Current color as a composite value.
Description	Gets integer value of current color.

hashCode

```
public int hashCode()
```

Returns	A hashcode to use when storing `Color` in a `Hashtable`.
Overrides	`Object.hashCode()`
Description	Generates a hashcode for the `Color`.

toString

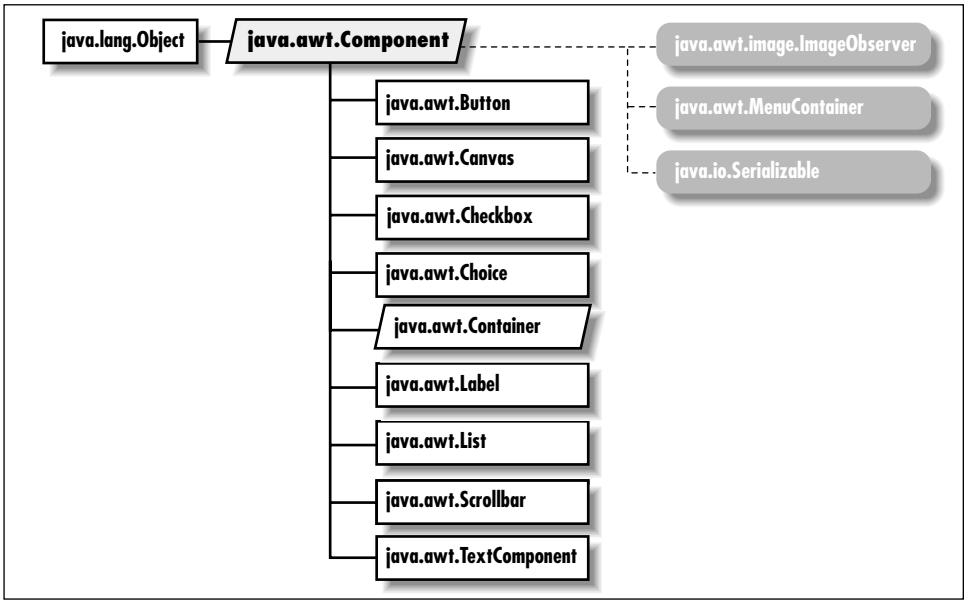
```
public String toString()
```

Returns	A string representation of the `Color` object.
Overrides	`Object.toString()`

See Also

`Object`, `Properties`, `Serializable`, `String`

19.15 Component



Description

The `Component` class is the parent of all non-menu GUI components.

Class Definition

```
public abstract class java.awt.Component
    extends java.lang.Object
    implements java.awt.image.ImageObserver
    implements java.awt.MenuContainer
    implements java.io.Serializable {

  // Constants
  public final static float BOTTOM_ALIGNMENT; ★
  public final static float CENTER_ALIGNMENT; ★
  public final static float LEFT_ALIGNMENT; ★
  public final static float RIGHT_ALIGNMENT; ★
  public final static float TOP_ALIGNMENT; ★

  // Variables
  protected Locale locale; ★
```

```
// Constructors
protected Component(); ★

// Instance Methods
public boolean action (Event e, Object o); ☆
public synchronized void add (PopupMenu popup); ★
public synchronized void addComponentListener
    (ComponentListener l); ★
public synchronized void addFocusListener (FocusListener l); ★
public synchronized void addKeyListener (KeyListener l); ★
public synchronized void addMouseListener (MouseListener l); ★
public synchronized void addMouseMotionListener
    (MouseMotionListener l); ★
public void addNotify();
public Rectangle bounds(); ☆
public int checkImage (Image image, ImageObserver observer);
public int checkImage (Image image, int width, int height,
    ImageObserver observer);
public boolean contains (int x, int y); ★
public boolean contains (Point p); ★
public Image createImage (ImageProducer producer);
public Image createImage (int width, int height);
public void deliverEvent (Event e); ☆
public void disable(); ☆
public final void dispatchEvent (AWTEvent e) ★
public void doLayout(); ★
public void enable(); ☆
public void enable (boolean condition); ☆
public float getAlignmentX(); ★
public float getAlignmentY(); ★
public Color getBackground();
public Rectangle getBounds(); ★
public synchronized ColorModel getColorModel();
public Component getComponentAt (int x, int y); ★
public Component getComponentAt (Point p); ★
public Cursor getCursor(); ★
public Font getFont();
public FontMetrics getFontMetrics (Font f);
public Color getForeground();
public Graphics getGraphics();
public Locale getLocale(); ★
public Point getLocation(); ★
public Point getLocationOnScreen(); ★
public Dimension getMaximumSize(); ★
public Dimension getMinimumSize(); ★
public String getName(); ★
public Container getParent();
public ComponentPeer getPeer(); ☆
public Dimension getPreferredSize(); ★
```

```
public Dimension getSize(); ★
public Toolkit getToolkit();
public final Object getTreeLock(); ★
public boolean gotFocus (Event e, Object o); ☆
public boolean handleEvent (Event e); ☆
public void hide(); ☆
public boolean imageUpdate (Image image, int infoflags, int x, int y,
    int width, int height);
public boolean inside (int x, int y); ☆
public void invalidate();
public boolean isEnabled();
public boolean isFocusTraversable(); ★
public boolean isShowing();
public boolean isValid();
public boolean isVisible();
public boolean keyDown (Event e, int key); ☆
public boolean keyUp (Event e, int key); ☆
public void layout(); ☆
public void list();
public void list (PrintStream out);
public void list (PrintStream out, int indentation);
public void list (PrintWriter out); ★
public void list (PrintWriter out, int indentation); ★
public Component locate (int x, int y); ☆
public Point location(); ☆
public boolean lostFocus (Event e, Object o); ☆
public Dimension minimumSize(); ☆
public boolean mouseDown (Event e, int x, int y); ☆
public boolean mouseDrag (Event e, int x, int y); ☆
public boolean mouseEnter (Event e, int x, int y); ☆
public boolean mouseExit (Event e, int x, int y); ☆
public boolean mouseMove (Event e, int x, int y); ☆
public boolean mouseUp (Event e, int x, int y); ☆
public void move (int x, int y); ☆
public void nextFocus(); ☆
public void paint (Graphics g);
public void paintAll (Graphics g);
public boolean postEvent (Event e); ☆
public Dimension preferredSize(); ☆
public boolean prepareImage (Image image, ImageObserver observer);
public boolean prepareImage (Image image, int width, int height,
    ImageObserver observer);
public void print (Graphics g);
public void printAll (Graphics g);
public synchronized void remove (MenuComponent popup); ★
public synchronized void removeComponentListener
    (ComponentListener l); ★
public synchronized void removeFocusListener (FocusListener l); ★
public synchronized void removeKeyListener (KeyListener l); ★
```

```
  public synchronized void removeMouseListener (MouseListener l); ★
  public synchronized void removeMouseMotionListener
      (MouseMotionListener l); ★
  public void removeNotify();
  public void repaint();
  public void repaint (long tm);
  public void repaint (int x, int y, int width, int height);
  public void repaint (long tm, int x, int y, int width, int height);
  public void requestFocus();
  public void reshape (int x, int y, int width, int height); ☆
  public void resize (Dimension d); ☆
  public void resize (int width, int height); ☆
  public void setBackground (Color c);
  public void setBounds (int x, int y, int width, int height); ★
  public void setBounds (Rectangle r); ★
  public synchronized void setCursor (Cursor cursor); ★
  public void setEnabled (boolean b); ★
  public synchronized void setFont (Font f);
  public void setForeground (Color c);
  public void setLocale (Locale l); ★
  public void setLocation (int x, int y); ★
  public void setLocation (Point p); ★
  public void setName (String name); ★
  public void setSize (int width, int height); ★
  public void setSize (Dimension d); ★
  public void setVisible (boolean b); ★
  public void show(); ☆
  public void show (boolean condition); ☆
  public Dimension size(); ☆
  public String toString();
  public void transferFocus(); ★
  public void update (Graphics g);
  public void validate();

  // Protected Instance Methods
  protected final void disableEvents (long eventsToDisable); ★
  protected final void enableEvents (long eventsToEnable); ★
  protected String paramString();
  protected void processComponentEvent (ComponentEvent e); ★
  protected void processEvent (AWTEvent e); ★
  protected void processFocusEvent (FocusEvent e); ★
  protected void processKeyEvent (KeyEvent e); ★
  protected void processMouseEvent (MouseEvent e); ★
  protected void processMouseMotionEvent (MouseEvent e); ★
}
```

Constants

BOTTOM_ALIGNMENT

```
public final static float BOTTOM_ALIGNMENT ★
```

Constant representing bottom alignment in `getAlignmentY()`.

CENTER_ALIGNMENT

```
public final static float CENTER_ALIGNMENT ★
```

Constant representing center alignment in `getAlignmentX()` and `getAlignmentY()`.

LEFT_ALIGNMENT

```
public final static float LEFT_ALIGNMENT ★
```

Constant representing left alignment in `getAlignmentX()`.

RIGHT_ALIGNMENT

```
public final static float RIGHT_ALIGNMENT ★
```

Constant representing right alignment in `getAlignmentX()`.

TOP_ALIGNMENT

```
public final static float TOP_ALIGNMENT ★
```

Constant representing top alignment in `getAlignmentY()`.

Variables

locale

```
protected Locale locale ★
```

Description The locale for the component. Used for internationalization support.

Constructors

Component

```
protected Component() ★
```

Description This constructor creates a "lightweight" component. This constructor allows `Component` to be directly subclassed using code written entirely in Java.

Instance Methods

action

```
public boolean action (Event e, Object o) ☆
```

Parameters	*e*	`Event` instance identifying what triggered the call to this method.
	o	Argument specific to the component subclass that generated the event.
Returns	`true` if event handled, `false` to propagate it to parent container.	
Description	Method called when user performs some action in `Component`. This method is a relic of the old 1.0.2 event model and is replaced by the `process . . . Event()` methods.	

add

```
public synchronized void add (PopupMenu popup) ★
```

Parameters	*popup*	The menu to add.
Description	After the `PopupMenu` is added to a component, it can be shown in the component's coordinate space.	

addComponentListener

```
public void addComponentListener (ComponentListener l) ★
```

Description: Adds a listener for the `ComponentEvent` objects this `Component` generates.

addFocusListener

```
public void addFocusListener (FocusListener l) ★
```

Description: Adds a listener for the `FocusEvent` objects this `Component` generates.

addKeyListener

```
public void addKeyListener (KeyListener l) ★
```

Description: Adds a listener for the `KeyEvent` objects this `Component` generates.

addMouseListener

```
public void addMouseListener (MouseListener l) ★
```

Description Adds a listener for the `MouseEvent` objects this `Component` generates.

addMouseMotionListener

```
public void addMouseMotionListener (MouseMotionListener l)
★
```

Description Adds a listener for the motion `MouseEvent` objects this `Component` generates.

addNotify

```
public void addNotify()
```

Description Creates peer of `Component`'s subclass.

bounds

```
public Rectangle bounds() ☆
```

Returns Gets bounding rectangle of `Component`.
Description A `Rectangle` that returns the outer limits of the `Component`. Replaced by `getBounds()` in 1.1.

checkImage

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

Parameters *image* `Image` to check.
observer The object an image will be rendered onto.
Returns `ImageObserver` Flags ORed together indicating the image's status.
Description Checks status of image construction.

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

Parameters *image* Image to check.
width Horizontal size image will be scaled to.
height Vertical size image will be scaled to.
observer Object image will be rendered onto.
Returns `ImageObserver` flags ORed together indicating the image's status.

Description Checks status of image construction.

contains

```
public boolean contains (int x, int y) ★
```

Parameters *x* The x coordinate, in this `Component`'s coordinate system.
y The y coordinate, in this `Component`'s coordinate system.
Returns `true` if the `Component` contains the point; `false` otherwise.

```
public boolean contains (Point p) ★
```

Parameters *p* The point to be tested, in this `Component`'s coordinate system.
Returns `true` if the `Component` contains the point; `false` otherwise.

createImage

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

Parameters *producer* Class that implements `ImageProducer` interface to create the new image.
Returns Newly created image instance.
Description Creates an `Image` based upon an `ImageProducer`.

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

Parameters *width* Horizontal size for in-memory `Image`.
height Vertical size for in-memory `Image`.
Returns Newly created image instance.
Description Creates an empty in-memory `Image` for double buffering; to draw on the image, use its graphics context.

deliverEvent

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

Parameters *e* `Event` instance to deliver.
Description Delivers event to the component for processing.

disable

```
public void disable() ☆
```

Description Disables component so that it is unresponsive to user interactions. Replaced by `setEnabled(false)`.

dispatchEvent

```
public final void dispatchEvent (AWTEvent e) ★
```

Parameters *e* The `AWTEvent` to process.
Description Tells the component to deal with the `AWTEvent e`.

doLayout

```
public void doLayout() ★
```

Description Lays out component. This method is a replacement for `layout()`.

enable

```
public void enable() ☆
```

Description Enables component so that it is responsive to user interactions. Use `setEnabled(true)` instead.

```
public void enable (boolean condition) ☆
```

Parameters *condition* `true` to enable the component; `false` to disable it.
Description Enables or disables the component based upon `condition`. Use `setEnabled(boolean)` instead.

getAlignmentX

```
public float getAlignmentX() ★
```

Returns A number between 0 and 1 representing the horizontal alignment of this component.
Description One of the constants `LEFT_ALIGNMENT`, `CENTER_ALIGNMENT`, or `RIGHT_ALIGNMENT` may be returned. `CENTER_ALIGNMENT` is returned by default.

getAlignmentY

```
public float getAlignmentY() ★
```

Returns A number between 0 and 1 representing the vertical alignment of this component.
Description One of the constants `TOP_ALIGNMENT`, `CENTER_ALIGNMENT`, or `BOTTOM_ALIGNMENT` may be returned. `CENTER_ALIGNMENT` is returned by default.

getBackground

```
public Color getBackground()
```

Returns Background color of the component.

getBounds

```
public Rectangle getBounds() ★
```

Returns Gets bounding rectangle of `Component`.

Description Returns a `Rectangle` that returns the outer limits of the `Component`.

getColorModel

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

Returns `ColorModel` used to display the current component.

getComponentAt

```
public Component getComponentAt (int x, int y) ★
```

Parameters *x* The x coordinate, in this `Component`'s coordinate system.

y The y coordinate, in this `Component`'s coordinate system.

Returns Returns the `Component` containing the given point.

```
public Component getComponentAt (Point p) ★
```

Parameters *p* The point to be tested, in this `Component`'s coordinate system.

Returns Returns the `Component` containing the given point.

getCursor

```
public Cursor getCursor() ★
```

Returns Current cursor of the component.

getFont

```
public Font getFont()
```

Returns Current font of the component.

getFontMetrics

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

Parameters	*f*	A `Font` object, whose platform specific information is desired.
Returns	Size information for the given `Font`.	

getForeground

```
public Color getForeground()
```

Returns	Foreground color of component.

getGraphics

```
public Graphics getGraphics()
```

Throws	`InternalException`	If acquiring graphics context is unsupported.
Returns	`Component`'s graphics context.	

getLocale

```
public Locale getLocale() ★
```

Throws	`IllegalComponentStateException`	If the component does not have a locale or it has not been added to a hierarchy that does.
Returns	`Component`'s locale.	

getLocation

```
public Point getLocation() ★
```

Returns	Position of component.
Description	Gets the current position of this `Component` in its parent's coordinate space.

getLocationOnScreen

```
public Point getLocationOnScreen() ★
```

Returns	Position of component.
Description	Gets the current position of this `Component` in the screen's coordinate space.

getMaximumSize

```
public Dimension getMaximumSize() ★
```

Returns The maximum dimensions of the component.
Description By default, a maximal `Dimension` is returned.

getMinimumSize

```
public Dimension getMinimumSize() ★
```

Returns The minimum dimensions of the component.

getName

```
public String getName() ★
```

Returns This component's name.

getParent

```
public Container getParent()
```

Returns Parent `Container` of `Component`.
Description Gets container that this `Component` is held in.

getPeer

```
public ComponentPeer getPeer() ☆
```

Returns Peer of `Component`.

getPreferredSize

```
public Dimension getPreferredSize() ★
```

Returns The preferred dimensions of the component.

getSize

```
public Dimension getSize() ★
```

Returns Dimensions of component.
Description Gets width and height of component.

getToolkit

```
public Toolkit getToolkit()
```

Returns `Toolkit` of `Component`.

getTreeLock

```
public final Object getTreeLock() ★
```

Returns	The AWT tree locking object.
Description	Returns the object used for tree locking and layout operations.

gotFocus

```
public boolean gotFocus (Event e, Object o) ☆
```

Parameters	*e*	`Event` instance identifying what triggered the call to this method.
	o	Argument specific to the component subclass that generated the event.
Returns	`true` if event handled, `false` to propagate it to parent container.	
Description	Called when `Component` gets input focus. This method is not used in the 1.1 event model.	

handleEvent

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

Parameters	*e*	`Event` instance identifying what triggered the call to this method.
Returns	`true` if event handled, `false` to propagate it to parent container.	
Description	High-level event handling routine that calls helper routines. Replaced by `processEvent(AWTEvent)`.	

hide

```
public void hide() ☆
```

Description	Hides component from view. Replaced by `setVisible(false)`.

imageUpdate

```
public boolean imageUpdate (Image image, int infoflags,
int x,
int y, int width, int height)
```

Parameters	*image*	`Image` being loaded.
	infoflags	`ImageObserver` flags ORed together of available information.

	x	x coordinate of upper-left corner of `Image`.
	y	y coordinate of upper-left corner of `Image`.
	width	Horizontal dimension of `Image`.
	height	Vertical dimension of `Image`.
Returns	`true` if `Image` fully loaded, `false` otherwise.	
Implements	`ImageObserver.imageUpdate()`	
Description	An asynchronous update interface for receiving notifications about `Image` information as it is loaded. Meaning of parameters changes with values of flags.	

inside

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

Parameters	*x*	Horizontal position.
	y	Vertical position.
Returns	`true` if the point (`x`, `y`) falls within the component's bounds, `false` otherwise.	
Description	Checks if coordinates are within bounding box of `Component`. Replaced by `contains(int, int)`.	

invalidate

```
public void invalidate()
```

Description	Sets the component's valid state to `false`.

isEnabled

```
public boolean isEnabled()
```

Returns	`true` if enabled, `false` otherwise.
Description	Checks to see if the `Component` is currently enabled.

isFocusTraversable

```
public boolean isFocusTraversable() ★
```

Returns	`true` if this `Component` can be traversed using Tab and Shift-Tab, `false` otherwise.
Description	Checks to see if the `Component` is navigable using the keyboard.

isShowing

```
public boolean isShowing()
```

Returns	`true` if showing, `false` otherwise.
Description	Checks to see if the `Component` is currently showing.

isValid

```
public boolean isValid()
```

Returns	`true` if valid, `false` otherwise.
Description	Checks to see if the `Component` is currently valid.

isVisible

```
public boolean isVisible()
```

Returns	`true` if visible, `false` otherwise.
Description	Checks to see if the `Component` is currently visible.

keyDown

```
public boolean keyDown (Event e, int key) ☆
```

Parameters	*e*	`Event` instance identifying what triggered the call to this method.
	key	Integer representation of key pressed.
Returns	`true` if event handled, `false` to propagate it to parent container.	
Description	Method called whenever the user presses a key. Replaced by `processKeyEvent(KeyEvent)`.	

keyUp

```
public boolean keyUp (Event e, int key) ☆
```

Parameters	*e*	`Event` instance identifying what triggered the call to this method.
	key	Integer representation of key released.
Returns	`true` if event handled, `false` to propagate it to parent container.	
Description	Method called whenever the user releases a key. Replaced by `processKeyEvent(KeyEvent)`.	

layout

```
public void layout() ☆
```

Description Lays out component. Replaced by `doLayout()`.

list

```
public void list()
```

Description Prints the contents of the `Component` to `System.out`.

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

Parameters *out* Output stream to send results to.
Description Prints the contents of the `Component` to a `PrintStream`.

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

Parameters *out* Output stream to send results to.
indentation Indentation to use when printing.
Description Prints the contents of the `Component` indented to a `PrintStream`.

```
public void list (PrintWriter out)
```

Parameters *out* Output stream to send results to.
Description Prints the contents of the `Component` to a `PrintWriter`.

```
public void list (PrintWriter out, int indentation)
```

Parameters *out* Output stream to send results to.
indentation Indentation to use when printing.
Description Prints the contents of the `Component` indented to a `PrintWriter`.

locate

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

Parameters *x* Horizontal position.
y Vertical position.
Returns `Component` if the point (`x`, `y`) falls within the component, `null` otherwise.
Description Replaced by `getComponentAt(int, int)`.

location

```
public Point location() ☆
```

Returns Position of component.

Description Gets the current position of this `Component` in its parent's coordinate space. Replaced by `getLocation()`.

`lostFocus`

```
public boolean lostFocus (Event e, Object o) ☆
```

Parameters
- *e* `Event` instance identifying what triggered the call to this method.
- *o* Argument specific to the component subclass that generated the event.

Returns `true` if event handled, `false` to propagate it to parent container.

Description Method called when `Component` loses input focus. Replaced by `processFocusEvent(FocusEvent)`.

`minimizeSize`

```
public Dimension minimumSize() ☆
```

Returns The minimum dimensions of the component. Replaced by `getMinimumSize()`.

`mouseDown`

```
public boolean mouseDown (Event e, int x, int y) ☆
```

Parameters
- *e* `Event` instance identifying what triggered the call to this method.
- *x* Horizontal position of the mouse within `Component` when `Event` initiated
- *y* Vertical position of the mouse within `Component` when `Event` initiated

Returns `true` if event handled, `false` to propagate it to parent container.

Description Method called when the user presses a mouse button over `Component`. Replaced by `processMouseEvent(MouseEvent)`.

`mouseDrag`

```
public boolean mouseDrag (Event e, int x, int y) ☆
```

Parameters
- *e* `Event` instance identifying what triggered the call to this method.
- *x* Horizontal position of the mouse within `Component` when `Event` initiated

	y	Vertical position of the mouse within `Component` when `Event` initiated
Returns	`true` if event handled, `false` to propagate it to parent container.	
Description	Method called when the user is pressing a mouse button and moves the mouse. Replaced by `processMouseMotionEvent(MouseEvent)`.	

mouseEnter

```
public boolean mouseEnter (Event e, int x, int y) ☆
```

Parameters	*e*	`Event` instance identifying what triggered the call to this method.
	x	Horizontal position of the mouse within `Component` when `Event` initiated
	y	Vertical position of the mouse within `Component` when `Event` initiated
Returns	`true` if event handled, `false` to propagate it to parent container.	
Description	Method called when the mouse enters `Component`. Replaced by `processMouseEvent(MouseEvent)`.	

mouseExit

```
public boolean mouseExit (Event e, int x, int y) ☆
```

Parameters	*e*	`Event` instance identifying what triggered the call to this method.
	x	Horizontal position of the mouse within `Component` when `Event` initiated
	y	Vertical position of the mouse within `Component` when `Event` initiated
Returns	`true` if event handled, `false` to propagate it to parent container.	
Description	Method called when the mouse exits `Component`. Replaced by `processMouseEvent(MouseEvent)`.	

mouseMove

```
public boolean mouseMove (Event e, int x, int y) ☆
```

Parameters	*e*	`Event` instance identifying what triggered the call to this method.

	x	Horizontal position of the mouse within `Component` when `Event` initiated
	y	Vertical position of the mouse within `Component` when `Event` initiated
Returns	`true` if event handled, `false` to propagate it to parent container.	
Description	Method called when the user is not pressing a mouse button and moves the mouse. Replaced by `processMouseMotionEvent(MouseEvent)`.	

mouseUp

```
public boolean mouseUp (Event e, int x, int y) ☆
```

Parameters	*e*	`Event` instance identifying what triggered the call to this method.
	x	Horizontal position of the mouse within `Component` when `Event` initiated
	y	Vertical position of the mouse within `Component` when `Event` initiated
Returns	`true` if event is handled, `false` to propagate it to the parent container.	
Description	Method called when user releases mouse button over `Component`. Replaced by `processMouseEvent(MouseEvent)`.	

move

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

Parameters	*x*	New horizontal position for component.
	y	New vertical position for component.
Description	Relocates component. Replaced by `setLocation(int, int)`.	

nextFocus

```
public void nextFocus() ☆
```

Description	Moves focus from current component to next one in parent container. Replaced by `transferFocus()`.

paint

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

Parameters	*g*	Graphics context of component.
Description	Empty method to be overridden to draw something in the graphics context.	

paintAll

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

Parameters	*g*	Graphics context of component.
Description	Method to validate component and paint its peer if it is visible.	

postEvent

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

Parameters	*e*	`Event` instance to post to component
Returns	If `Event` is handled, `true` is returned. Otherwise, `false` is returned.	
Description	Tells `Component` to deal with `Event`.	

preferredSize

```
public Dimension preferredSize() ☆
```

Returns	The preferred dimensions of the component. Replaced by `getPreferredSize()`.

prepareImage

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

Parameters	*image*	`Image` to start loading.
	observer	Component on which image will be rendered.
Returns	`true` if `Image` is fully loaded, `false` otherwise.	
Description	Forces `Image` to start loading.	

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

Parameters	*image*	`Image` to start loading.
	width	Horizontal size of the `Image` after scaling.
	height	Vertical size of the `Image` after scaling.
	observer	Component on which image will be rendered.
Returns	`true` if `Image` is fully loaded, `false` otherwise.	

Description Forces `Image` to start loading.

print

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

Parameters *g* Graphics context.
Description Empty method to be overridden to print something into the graphics context.

printAll

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

Parameters *g* Graphics context.
Description Method to print this component and its children.

remove

```
public void remove (MenuComponent popup) ★
```

Parameters *popup* The menu to remove.
Description After adding a `PopupMenu`, you can use this method to remove it.

removeComponentListener

```
public void removeComponentListener (ComponentListener l)
★
```

Description Removes the specified `ComponentListener` from this `Component`.

removeFocusListener

```
public void removeFocusListener (FocusListener l) ★
```

Description Removes the specified `FocusListener` from this `Component`.

removeKeyListener

```
public void removeKeyListener (KeyListener l) ★
```

Description Removes the specified `KeyListener` from this `Component`.

removeMouseListener

```
public void removeMouseListener (MouseListener l) ★
```

Description Removes the specified `MouseListener` from this `Component`.

removeMouseMotionListener

```
public void removeMouseMotionListener (MouseMotionListener
l) ★
```

Description Removes the specified `MouseMotionListener` from this `Component`.

removeNotify

```
public void removeNotify()
```

Description Removes peer of `Component`'s subclass.

repaint

```
public void repaint()
```

Description Requests scheduler to redraw the component as soon as possible.

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

Parameters *tm* Millisecond delay allowed before repaint.
Description Requests scheduler to redraw the component within a time period.

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

Parameters *x* Horizontal origin of bounding box to redraw.
y Vertical origin of bounding box to redraw.
width Width of bounding box to redraw.
height Height of bounding box to redraw.
Description Requests scheduler to redraw a portion of component as soon as possible.

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

Parameters *tm* Millisecond delay allowed before repaint.
x Horizontal origin of bounding box to redraw.
y Vertical origin of bounding box to redraw.
width Width of bounding box to redraw.
height Height of bounding box to redraw.

Description Requests scheduler to redraw a portion of component within a time period.

requestFocus

```
public void requestFocus()
```

Description Requests the input focus for this `Component`.

reshape

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

Parameters	*x*	New horizontal position for component.
	y	New vertical position for component.
	width	New width for component.
	height	New height for component.

Description Relocates and resizes component. Replaced by `setBounds(int, int, int, int)`.

resize

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

Parameters	*d*	New dimensions for the component.

Description Resizes component. Replaced by `setSize(Dimension)`.

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

Parameters	*width*	New width for component.
	height	New height for component.

Description Resizes component. Replaced by `setSize(int, int)`.

setBackground

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

Parameters	*c*	New background color.

Description Changes the component's background color.

setBounds

```
public void setBounds (int x, int y, int width, int
height)
```
★

Parameters	*x*	New horizontal position for component.
	y	New vertical position for component.

	width	New width for component.
	height	New height for component.
Description	Relocates and resizes the component.	

```
public void setBounds (Rectangle r) ★
```

Parameters	*r*	New coordinates for component.
Description	Relocates and resizes component.	

setCursor

```
public synchronized void setCursor (Cursor cursor) ★
```

Parameters	*cursor*	The new cursor for the component.
Description	Changes the component's cursor.	

setEnabled

```
public void setEnabled (boolean b) ★
```

Parameters	*b*	`true` to enable the component, `false` to disable it.
Description	Enables or disables the component. Replaces `enable()`, `enable(boolean)`, and `disable()`.	

setFont

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

Parameters	*f*	Font to change component to.
Description	Changes the font of the component.	

setForeground

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

Parameters	*c*	New foreground color.
Description	Changes the foreground color of component's area.	

setLocale

```
public void setLocale (Locale l) ★
```

Parameters	*l*	The locale object for the component.
Description	Sets the `Component`'s locale.	

setLocation

```
public void setLocation (int x, int y) ★
```

Parameters	*x*	New horizontal position for component.
	y	New vertical position for component.
Description	Relocates the component.	

```
public void setLocation (Point p) ★
```

Parameters	*p*	New position for component.
Description	Relocates the component.	

setName

```
public void setName (String name) ★
```

Parameters	*name*	New name for component.
Description	Sets the component's name.	

setSize

```
public void setSize (int width, int height) ★
```

Parameters	*width*	New width for component.
	height	New height for component.
Description	Resizes the component.	

```
public void setSize (Dimension d) ★
```

Parameters	*d*	New dimensions for the component.
Description	Resizes the component.	

setVisible

```
public void setVisible (boolean b) ★
```

Parameters	*b*	`true` to show component, `false` to hide it.
Description	Shows or hides the component based on the `b` parameter.	

show

```
public void show() ☆
```

Description: Replaced by `setVisible(true)`.

```
public void show (boolean condition) ☆
```

Parameters	*condition*	`true` to show the component, `false` to hide it.
Description	Replaced by `setVisible(boolean)`.	

size

```
public Dimension size() ☆
```

Returns Dimensions of the component.
Description Gets width and height of the component. Replaced by `getSize()`.

toString

```
public String toString()
```

Returns A string representation of the `Component` object.
Overrides `Object.toString()`

transferFocus

```
public void transferFocus() ★
```

Description Transfers focus to the next component in the container hierarchy.

update

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

Parameters *g* Graphics context of component.
Description Called to update the component's display area.

validate

```
public void validate()
```

Description Sets the component's valid state to `true`.

Protected Instance Methods

disableEvents

```
protected final void disableEvents (long eventsToDisable)
★
```

Parameters *eventsToDisable*
A value representing certain kinds of events. This can be constructed by ORing the event mask constants defined in `java.awt.AWTEvent`.
Description By default, a component receives events corresponding to the event listeners that have registered. If a component should not receive events of a certain type, even if there is a listener registered for that type of event, this method can be used to disable that event type.

enableEvents

```
protected final void enableEvents (long eventsToEnable) ★
```

Parameters *eventsToEnable* A value representing certain kinds of events. This can be constructed by ORing the event mask constants defined in `java.awt.AWTEvent`.

Description By default, a component receives events corresponding to the event listeners that have registered. If a component should receive other types of events as well, this method can be used to request them.

paramString

```
protected String paramString()
```

Returns A `String` with the current settings of the `Component`.

Description Helper method for `toString()` to generate a string of current settings.

processComponentEvent

```
protected void processComponentEvent(ComponentEvent e) ★
```

Parameters *e* The event to process.

Description Component events are passed to this method for processing. Normally, this method is called by `processEvent()`.

processEvent

```
protected void processEvent(AWTEvent e) ★
```

Parameters *e* The event to process.

Description Low level `AWTEvents` are passed to this method for processing.

processFocusEvent

```
protected void processFocusEvent(FocusEvent e) ★
```

Parameters *e* The event to process.

Description Focus events are passed to this method for processing. Normally, this method is called by `processEvent()`.

processKeyEvent

```
protected void processKeyEvent(KeyEvent e) ★
```

Parameters	*e*	The event to process.
Description	Key events are passed to this method for processing. Normally, this method is called by `processEvent()`.	

processMouseEvent

```
protected void processMouseEvent(MouseEvent e) ★
```

Parameters	*e*	The event to process.
Description	Mouse events are passed to this method for processing. Normally, this method is called by `processEvent()`.	

processMouseMotionEvent

```
protected void processMouseMotionEvent(MouseEvent e) ★
```

Parameters	*e*	The event to process.
Description	Mouse motion events are passed to this method for processing. Normally, this method is called by `processEvent()`.	

See Also

`Button`, `Canvas`, `Checkbox`, `Choice`, `Color`, `ColorModel`, `ComponentPeer`, `Container`, `Dimension`, `Event`, `Font`, `FontMetrics`, `Graphics`, `ImageObserver`, `ImageProducer`, `Label`, `List`, `MenuContainer`, `Object`, `Point`, `PrintStream`, `Rectangle`, `Scrollbar`, `Serializable`, `String`, `TextComponent`, `Toolkit`

19.16 Container

Description

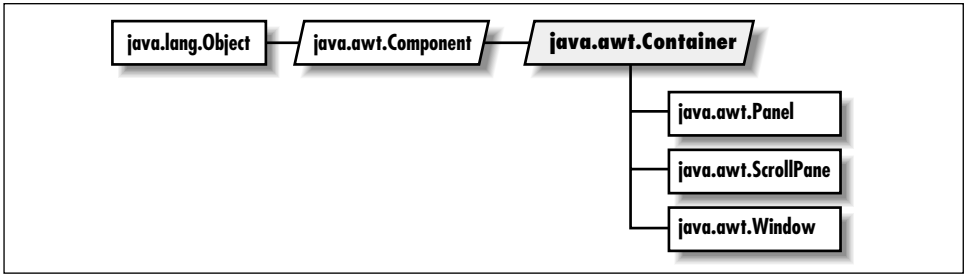
The `Container` class serves as a general purpose holder of other `Component` objects.

Class Definition

```
public abstract class java.awt.Container
    extends java.awt.Component {

  // Constructors
  protected Container(); ★

  // Instance Methods
  public Component add (Component component);
  public Component add (Component component, int position);
```




```
public void add (Component comp, Object constraints); ★
public void add (Component comp, Object constraints,
    int position); ★
public Component add (String name, Component component); ☆
public synchronized void addContainerListener (ContainerListener l); ★
public void addNotify();
public int countComponents();
public void deliverEvent (Event e); ★
public void doLayout(); ★
public float getAlignmentX(); ★
public float getAlignmentY(); ★
public Component getComponent (int n);
public Component getComponentAt (int x, int y);  ★
public Component getComponentAt (Point p);  ★
public int getComponentCount(); ★
public Component[] getComponents();
public Insets getInsets(); ★
public LayoutManager getLayout();
public Dimension getMaximumSize(); ★
public Dimension getMinimumSize();  ★
public Dimension getPreferredSize();  ★
public Insets insets();
public void invalidate(); ★
public boolean isAncestorOf (Component c); ★
public void layout(); ☆
public void list (PrintStream out, int indentation);
public void list (PrintWriter out, int indentation); ★
public Component locate (int x, int y); ☆
public Dimension minimumSize(); ☆
public void paint (Graphics g); ★
public void paintComponents (Graphics g);
public Dimension preferredSize(); ☆
public void print (Graphics g); ★
public void printComponents (Graphics g);
public void remove (int index); ★
public void remove (Component component);
public void removeAll();
public void removeContainerListener (ContainerListener l); ★
```

```
  public void removeNotify();
  public void setLayout (LayoutManager manager);
  public void validate();

  // Protected Instance Methods
  protected void addImpl (Component comp, Object constraints,
      int index); ★
  protected String paramString();
  protected void processContainerEvent (ContainerEvent e); ★
  protected void processEvent (AWTEvent e); ★
  protected void validateTree(); ★
}
```

Constructors

`Container`

```
protected Container() ★
```

Description This constructor creates a "lightweight" container. This constructor allows `Container` to be subclassed using code written entirely in Java.

Instance Methods

`add`

```
public Component add (Component component)
```

Parameters	*component*	`Component` to add to container.
Returns	`Component` just added.	
Throws	`IllegalArgumentException` if you add `component` to itself.	
Description	Adds `component` as the last component in the container.	

```
public Component add (Component component, int position)
```

Parameters	*component*	`Component` to add to container.
	position	Position of component; -1 adds the component as the last in the container.
Returns	`Component just added.`	
Throws	`ArrayIndexOutOfBoundsException`	If `position` invalid.
	`IllegalArgumentException`	If you add `Component` to itself.
Description	Adds `component` to container at a certain position.	

```
public void add (Component component, Object constraints)
```
★

Parameters	*component*	`Component` to add to container.
	constraints	An object describing constraints on the component being added.
Description	Adds `component` to container subject to `contraints`.	

```
public void add (Component component, Object constraints,
int index)
```
★

Parameters	*component*	`Component` to add to container.
	constraints	An object describing constraints on the component being added.
	index	The position of the component in the container's list.
Description	Adds `component` to container subject to `contraints` at position `index`.	

```
public Component add (String name, Component component)
```
☆

Parameters	*name*	Name of component being added. This parameter is often significant to the layout manager of the container (e.g "North", "Center").
	component	`Component` to add to container.
Returns	`Component` just added.	
Throws	*IllegalArgumentException*	If you add `component` to itself.
Description	Adds the component to the container with the given `name`. Replaced by the more general `add(Component, Object)`.	

addContainerListener

```
public synchronized void addContainerListener (ContainerListener l)
```
★

Parameters	*l*	An object that implements the `ContainerListener` interface.
Description	Add a listener for the container events.	

addNotify

```
public void addNotify()
```

Overrides	`Component.addNotify()`
Description	Creates `Container`'s peer and peers of contained components.

countComponents

```
public int countComponents()
```

Returns Number of components within `Container`.

deliverEvent

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

Parameters *e* `Event` instance to deliver.
Overrides `Component.deliverEvent(Event)`
Description Tries to locate the component contained in the container that should receive the event.

doLayout

```
public void doLayout() ★
```

Description Lays out the container. This method is a replacement for `layout()`.

getAlignmentX

```
public float getAlignmentX() ★
```

Returns A number between 0 and 1 representing the horizontal alignment of this component.
Overrides `Component.getAlignmentX()`
Description If the container's layout manager implements `LayoutManager2`, this method returns the `getLayoutAlignmentX()` value of the layout manager. Otherwise the `getAlignmentX()` value of `Component` is returned.

getAlignmentY

```
public float getAlignmentY() ★
```

Returns A number between 0 and 1 representing the vertical alignment of this component.
Overrides `Component.getAlignmentY()`
Description If the container's layout manager implements `LayoutManager2`, this method returns the `getLayoutAlignmentY()` value of the layout manager. Otherwise the `getAlignmentY()` value of `Component` is returned.

getComponent

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

Parameters	*position*	Position of component to get.
Throws	`ArrayIndexOutOfBoundsException`	If `position` is invalid.
Returns	`Component` at designated `position` within `Container`.	

getComponentAt

```
public Component getComponentAt (int x, int y) ★
```

Parameters	*x*	The x coordinate, in this `Container`'s coordinate system.
	y	The y coordinate, in this `Container`'s coordinate system.
Returns	Returns the `Component` containing the give point.	

```
public Component getComponentAt (Point p) ★
```

Parameters	*p*	The point to be tested, in this `Container`'s coordinate system.
Returns	Returns the `Component` containing the give point.	

getComponentCount

```
public int getComponentCount() ★
```

Returns	Returns the number of components in the container.

getComponents

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

Returns	Array of components within the container.

getInsets

```
public Insets getInsets()
```

Returns	The insets of the container.

getLayout

```
public LayoutManager getLayout()
```

Returns	`LayoutManager` of `Container`.

getMaximumSize

```
public Dimension getMaximumSize() ★
```

Overrides `Component.getMaximumSize()`
Returns The maximum dimensions of the component.

getMinimumSize

```
public Dimension getMinimumSize() ★
```

Overrides `Component.getMinimumSize()`
Returns The minimum dimensions of the component.

getPreferredSize

```
public Dimension getPreferredSize() ★
```

Returns The preferred dimensions of the component.

insets

```
public Insets insets() ☆
```

Returns Current `Insets` of `Container`. Replaced by `getInsets()`.

invalidate

```
public void invalidate()
```

Overrides `Component.invalidate()`
Description Sets the container's valid state to `false`.

isAncestorOf

```
public boolean isAncestorOf (Component c) ★
```

Parameters *c* The component in question.
Returns If `c` is contained in the container's hierarchy, returns `true`; otherwise `false`.

layout

```
public void layout() ☆
```

Overrides `Component.layout()`
Description Replaced by `doLayout()`.

list

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

Parameters	*out*	Output `Stream` to send results to.
	indentation	Indentation to use when printing.
Overrides	`Component.list(PrintStream, int)`	
Description	Recursively lists all components in `Container`.	

```
public void list (PrintWriter out, int indentation)
```

Parameters	*out*	Output `Writer` to send results to.
	indentation	Indentation to use when printing.
Overrides	`Component.list(PrintWriter, int)`	
Description	Recursively lists all components in `Container`.	

locate

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

Parameters	*x*	Horizontal position to check.
	y	Vertical position to check.
Returns	`Component` within `Container` at given coordinates, or `Container`.	
Overrides	`Component.locate(int, int)`	
Description	Replaced by `getComponentAt(int, int)`.	

minimizeSize

```
public Dimension minimumSize() ☆
```

Returns	Minimum dimensions of contained objects.
Overrides	`Component.minimumSize()`
Description	Replaced by `getMinimumSize()`.

paint

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

Parameters	*g*	Graphics context of container.
Overrides	`Component.paint()`	
Description	This method tells any lightweight components that are children of this container to paint themselves.	

paintComponents

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

Parameters	*g*	Graphics context of `Container`.
Description	Paints the different components in `Container`.	

preferredSize

```
public Dimension preferredSize() ☆
```

Returns	Preferred dimensions of contained objects.
Overrides	`Component.preferredSize()`
Description	Replaced by `getPreferredSize()`.

print

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

Parameters	*g*	Graphics context of container.
Overrides	`Component.print()`	
Description	This method tells any lightweight components that are children of this container to print themselves.	

printComponents

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

Parameters	*g*	Graphics context of `Container`.
Description	Prints the different components in `Container`.	

remove

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

Parameters	*index*	Index of the component to remove.
Description	Removes the component in position `index` from `Container`.	

```
public void remove (Component component)
```

Parameters	*component*	`Component` to remove.
Description	Removes `component` from `Container`.	

removeAll

```
public void removeAll()
```

Description	Removes all components from `Container`.

removeContainerListener

```
public void removeContainerListener (ContainerListener l)
```
★

Parameters	*l*	One of this `Container`'s `ContainerListeners`.
Description	Remove a container event listener.	

removeNotify

```
public void removeNotify()
```

Overrides	`Component.removeNotify()`
Description	Removes `Container`'s peer and peers of contained components.

setLayout

```
public void setLayout (LayoutManager manager)
```

Parameters	*manager*	New `LayoutManager` for `Container`.
Description	Changes `LayoutManager` of `Container`.	

validate

```
public void validate()
```

Overrides	`Component.validate()`
Description	Sets `Container`'s valid state to `true` and recursively validates its children.

Protected Instance Methods

addImpl

```
protected void addImpl (Component comp, Object
constraints, int index)
```
★

Parameters	*comp*	The component to add.
	constraints	Constraints on the component.
	index	Position at which to add this component. Pass -1 to add the component at the end.
Description	This method adds a component subject to the given constraints at a specific position in the container's list of components. It is a helper method for the various overrides of `add()`.	

paramString

```
protected String paramString()
```

Returns	String with current settings of `Container`.
Overrides	`Component.paramString()`
Description	Helper method for `toString()` to generate string of current settings.

processContainerEvent

```
protected void processContainerEvent (ContainerEvent e) ★
```

Parameters	*e*	The event to process.
Description	Container events are passed to this method for processing. Normally, this method is called by `processEvent()`.	

processEvent

```
protected void processEvent (AWTEvent e) ★
```

Parameters	*e*	The event to process.
Overrides	`Component.processEvent()`	
Description	Low level `AWTEvents` are passed to this method for processing.	

validateTree

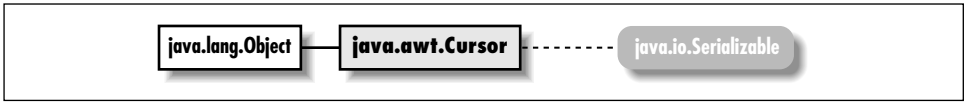
```
protected void validateTree() ★
```

Description	Descends recursively into the `Container`'s components and recalculates layout for any subtrees that are marked invalid.

See Also

`Component`, `Dimension`, `Event`, `Graphics`, `Insets`, `LayoutManager`, `Panel`, `PrintStream`, `String`, `Window`

19.17 Cursor ⋆



Description

The `Cursor` class represents the mouse pointer. It encapsulates information that used to be in `java.awt.Frame` in the 1.0.2 release.

Class Definition

```
public class java.awt.Cursor
    extends java.lang.Object
    implements java.io.Serializable {

  // Constants
  public final static int CROSSHAIR_CURSOR;
  public final static int DEFAULT_CURSOR;
  public final static int E_RESIZE_CURSOR;
  public final static int HAND_CURSOR;
  public final static int MOVE_CURSOR;
  public final static int N_RESIZE_CURSOR;
  public final static int NE_RESIZE_CURSOR;
  public final static int NW_RESIZE_CURSOR;
  public final static int S_RESIZE_CURSOR;
  public final static int SE_RESIZE_CURSOR;
  public final static int SW_RESIZE_CURSOR;
  public final static int TEXT_CURSOR;
  public final static int W_RESIZE_CURSOR;
  public final static int WAIT_CURSOR;

  // Class Variables
  protected static Cursor[] predefined;

  // Class Methods
  public static Cursor getDefaultCursor();
  public static Cursor getPredefinedCursor (int type);

  // Constructors
  public Cursor (int type);

  // Instance Methods
  public int getType();
}
```

Constants

CROSSHAIR_CURSOR

```
public final static int CROSSHAIR_CURSOR
```

Constant representing a cursor that looks like a crosshair.

DEFAULT_CURSOR

```
public final static int DEFAULT_CURSOR
```

Constant representing the platform's default cursor.

E_RESIZE_CURSOR

```
public final static int E_RESIZE_CURSOR
```

Constant representing the cursor for resizing an object on the left.

HAND_CURSOR

```
public final static int HAND_CURSOR
```

Constant representing a cursor that looks like a hand.

MOVE_CURSOR

```
public final static int MOVE_CURSOR
```

Constant representing a cursor used to move an object.

N_RESIZE_CURSOR

```
public final static int N_RESIZE_CURSOR
```

Constant representing a cursor for resizing an object on the top.

NE_RESIZE_CURSOR

```
public final static int NE_RESIZE_CURSOR
```

Constant representing a cursor for resizing an object on the top left corner.

NW_RESIZE_CURSOR

```
public final static int NW_RESIZE_CURSOR
```

Constant representing a cursor for resizing an object on the top right corner.

S_RESIZE_CURSOR

```
public final static int S_RESIZE_CURSOR
```

Constant representing a cursor for resizing an object on the bottom.

SE_RESIZE_CURSOR

```
public final static int SE_RESIZE_CURSOR
```

Constant representing a cursor for resizing an object on the bottom left corner.

SW_RESIZE_CURSOR

```
public final static int SW_RESIZE_CURSOR
```

Constant representing a cursor for resizing an object on the bottom right corner.

TEXT_CURSOR

```
public final static int TEXT_CURSOR
```

Constant representing a cursor used within text.

W_RESIZE_CURSOR

```
public final static int W_RESIZE_CURSOR
```

Constant representing a cursor for resizing an object on the right side.

WAIT_CURSOR

```
public final static int WAIT_CURSOR
```

Constant representing a cursor that indicates the program is busy.

Class Variables

predefined

```
protected static Cursor[] predefined
```

An array of cursor instances corresponding to the predefined cursor types.

Class Methods

getDefaultCursor

```
public static Cursor getDefaultCursor()
```

Returns The default system cursor.

getPredefinedCursor

```
public static Cursor getPredefinedCursor (int type)
```

Parameters	*type*	One of the type constants defined in this class.
Returns	A `Cursor` object with the specified type.	

Constructors

Cursor

```
public Cursor (int type)
```

Parameters	*type*	One of the type constants defined in this class.
Description	Constructs a `Cursor` object with the specified type.	

Instance Methods

getType

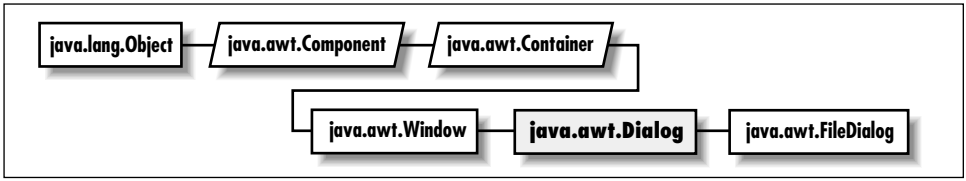
```
public int getType()
```

Returns	The type of cursor.

See Also

`Frame`

19.18 Dialog



Description

The `Dialog` class provides a special type of display window that is used for pop-up messages and acquiring input from the user. Unlike most other components, dialogs are hidden by default; you must call `show()` to display them. Dialogs are always associated with a parent `Frame`. A `Dialog` may be either modal or non-modal; a modal dialog attracts all input typed by the user. The default layout for a `Dialog` is `BorderLayout`.

Class Definition

```
public class java.awt.Dialog
    extends java.awt.Window {

  // Constructors
  public Dialog (Frame parent); ★
  public Dialog (Frame parent, boolean modal);
  public Dialog (Frame parent, String title); ★
  public Dialog (Frame parent, String title, boolean modal);

  // Instance Methods
  public void addNotify();
  public String getTitle();
  public boolean isModal();
  public boolean isResizable();
  public void setModal (boolean b); ★
  public synchronized void setResizable (boolean resizable);
  public synchronized void setTitle (String title);
  public void show();  ★

  // Protected Instance Methods
  protected String paramString();
}
```

Constructors

Dialog

```
public Dialog (Frame parent) ★
```

Parameters	*parent*	`Frame` that is to act as the parent of `Dialog`.
Throws	`IllegalArgumentException`	If `parent` is `null`.
Description	Constructs a `Dialog` object.	

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

Parameters	*parent*	`Frame` that is to act as the parent of `Dialog`.
	modal	`true` if the `Dialog` is modal; `false` otherwise.
Throws	`IllegalArgumentException`	If `parent` is `null`.
Description	Replaced with `Dialog(Frame, String, boolean)`.	

```
public Dialog (Frame parent, String title) ★
```

Parameters	*parent*	`Frame` that is to act as parent of `Dialog`.
	title	Initial title to use for `Dialog`.

Throws	`IllegalArgumentException`	If `parent` is `null`.
Description	Constructs a `Dialog` object with given characteristics.	

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

Parameters	*parent*	`Frame` that is to act as parent of `Dialog`.
	title	Initial title to use for `Dialog`.
	modal	`true` if the `Dialog` is modal; `false` otherwise.
Throws	`IllegalArgumentException`	If `parent` is `null`.
Description	Constructs a `Dialog` object with given characteristics.	

Instance Methods

addNotify

```
public void addNotify()
```

Overrides	`Window.addNotify()`
Description	Creates `Dialog`'s peer and peers of contained components.

getTitle

```
public String getTitle()
```

Returns	The current title for the `Dialog`.

isModal

```
public boolean isModal()
```

Returns	`true` if modal, `false` otherwise.

isResizable

```
public boolean isResizable()
```

Returns	`true` if resizable, `false` otherwise.

setModal

```
public void setModal (boolean b) ★
```

Parameters	*b*	`true` makes the `Dialog` modal; `false` if the `Dialog` should be modeless.
Description	Changes the modal state of the `Dialog`.	

setResizable

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

Parameters	*resizable*	`true` makes the `Dialog` resizable; `false` if the `Dialog` cannot be resized.
Description	Changes the resize state of the `Dialog`.	

setTitle

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

Parameters	*title*	New title for the `Dialog`.
Description	Changes the title of the `Dialog`.	

show

```
public void show() ★
```

Overrides	`Window.show()`
Description	If the dialog is hidden, this method shows it. If the dialog is already visible, this method brings it to the front.

Protected Instance Methods

paramString

```
protected String paramString()
```

Returns	String with current settings of `Dialog`.
Overrides	`Container.paramString()`
Description	Helper method for `toString()` to generate string of current settings.

See Also

`FileDialog`, `Frame`, `String`, `Window`, `WindowEvent`, `WindowListener`

19.19 Dimension

Description

The `Dimension` class encapsulates width and height in a single object.

Class Definition

```
public class java.awt.Dimension
    extends java.lang.Object
    implements java.io.Serializable {

  // Variables
  public int height;
  public int width;

  // Constructors
  public Dimension();
  public Dimension (int width, int height);
  public Dimension (Dimension d);

  // Instance Methods
  public boolean equals (Object obj); ★
  public Dimension getSize(); ★
  public void setSize (Dimension d); ★
  public void setSize (int width, int height); ★
  public String toString();
}
```

Variables

height

```
public int height
```

The height of the `Dimension`.

width

```
public int width
```

The width of the `Dimension`.

Constructors

Dimension

```
public Dimension()
```

Description Constructs an empty `Dimension` object.

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

Parameters	*width*	Initial width of the object
	height	Initial height of the object
Description	Constructs a `Dimension` object with an initial dimension of `width` x `height`.	

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

Parameters	*d*	Initial dimensions of the object
Description	Constructs a `Dimension` object that is a clone of `d`.	

Instance Methods

`equals`

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

Parameters	*obj*	The object to compare.
Returns	`true` if this `Dimension` is equivalent to `obj`; false otherwise.	
Overrides	`Object.equals(Object)`	
Description	Compares two `Dimension` instances.	

`getSize`

```
public Dimension getSize() ★
```

Returns	The size of the `Dimension`.

`setSize`

```
public void setSize (Dimension d) ★
```

Parameters	*d*	The new size.
Description	Changes the size of the `Dimension`.	

```
public void setSize (int width, int height) ★
```

Parameters	*width*	The new width.
	height	The new height.
Description	Changes the size of the `Dimension`.	

`toString`

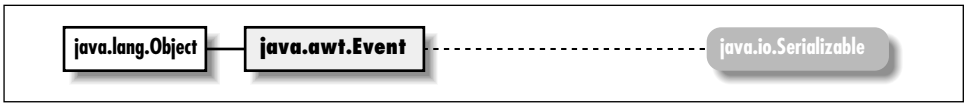
```
public String toString()
```

Returns	A string representation of the `Dimension` object.
Overrides	`Object.toString()`

See Also

`Object`, `String`, `Serializable`

19.20 Event



Description

The `Event` class represents events that happen within the Java environment in a platform independent way. Events typically represent user actions, like typing a key or clicking the mouse. Although this class has been updated for the 1.1 release, it is only used for the 1.0 event model. When using the 1.1 event model, all events are represented by subclasses of `java.awt.AWTEvent`.

Class Definition

```
public class java.awt.Event
    extends java.lang.Object
    implements java.io.Serializable {

  // Constants
  public static final int ACTION_EVENT;
  public static final int ALT_MASK;
  public static final int BACK_SPACE; ★
  public static final int CAPS_LOCK; ★
  public static final int CTRL_MASK;
  public static final int DELETE; ★
  public static final int DOWN;
  public static final int END;
  public static final int ENTER; ★
  public static final int ESCAPE; ★
  public static final int F1;
  public static final int F2;
  public static final int F3;
  public static final int F4;
  public static final int F5;
  public static final int F6;
  public static final int F7;
  public static final int F8;
  public static final int F9;
  public static final int F10;
  public static final int F11;
  public static final int F12;
```

```
public static final int GOT_FOCUS;
public static final int HOME;
public static final int INSERT; ★
public static final int KEY_ACTION;
public static final int KEY_ACTION_RELEASE;
public static final int KEY_PRESS;
public static final int KEY_RELEASE;
public static final int LEFT;
public static final int LIST_DESELECT;
public static final int LIST_SELECT;
public static final int LOAD_FILE;
public static final int LOST_FOCUS;
public static final int META_MASK;
public static final int MOUSE_DOWN;
public static final int MOUSE_DRAG;
public static final int MOUSE_ENTER;
public static final int MOUSE_EXIT;
public static final int MOUSE_MOVE;
public static final int MOUSE_UP;
public static final int NUM_LOCK; ★
public static final int PAUSE; ★
public static final int PGDN;
public static final int PGUP;
public static final int PRINT_SCREEN; ★
public static final int RIGHT;
public static final int SAVE_FILE;
public static final int SCROLL_ABSOLUTE;
public static final int SCROLL_BEGIN; ★
public static final int SCROLL_END; ★
public static final int SCROLL_LINE_DOWN;
public static final int SCROLL_LINE_UP;
public static final int SCROLL_LOCK; ★
public static final int SCROLL_PAGE_DOWN;
public static final int SCROLL_PAGE_UP;
public static final int SHIFT_MASK;
public static final int TAB; ★
public static final int UP;
public static final int WINDOW_DEICONIFY;
public static final int WINDOW_DESTROY;
public static final int WINDOW_EXPOSE;
public static final int WINDOW_ICONIFY;
public static final int WINDOW_MOVED;

// Variables
public Object arg;
public int clickCount;
public Event evt;
public int id;
public int key;
```

```
    public int modifiers;
    public Object target;
    public long when;
    public int x;
    public int y;

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

    // Instance Methods
    public boolean controlDown();
    public boolean metaDown();
    public boolean shiftDown();
    public String toString();
    public void translate (int x, int y);

    // Protected Instance Methods
    protected String paramString();
}
```

Constants

ACTION_EVENT

```
public static final int ACTION_EVENT
```

ID constant for Action `Event`.

ALT_MASK

```
public static final int ALT_MASK
```

Mask for ALT key.

BACK_SPACE

```
public static final int BACK_SPACE ★
```

ID constant for Backspace.

CAPS_LOCK

```
public static final int CAPS_LOCK ★
```

ID constant for Caps Lock key.

CTRL_MASK

```
public static final int CTRL_MASK
```

Mask for Control key.

DELETE

```
public static final int DELETE ★
```

ID constant for Delete.

DOWN

```
public static final int DOWN
```

ID constant for the down arrow key.

END

```
public static final int END
```

ID constant for End key.

ENTER

```
public static final int ENTER ★
```

ID constant for Enter key.

ESCAPE

```
public static final int ESCAPE ★
```

ID constant for Escape key.

F1

```
public static final int F1
```

ID constant for F1 key.

F2

```
public static final int F2
```

ID constant for F2 key.

F3

```
public static final int F3
```

ID constant for F3 key.

F4

```
public static final int F4
```

ID constant for F4 key.

F5

```
public static final int F5
```

ID constant for F5 key.

F6

```
public static final int F6
```

ID constant for F6 key.

F7

```
public static final int F7
```

ID constant for F7 key.

F8

```
public static final int F8
```

ID constant for F8 key.

F9

```
public static final int F9
```

ID constant for F9 key.

F10

```
public static final int F10
```

ID constant for F10 key.

F11

```
public static final int F11
```

ID constant for F11 key.

F12

```
public static final int F12
```

ID constant for F12 key.

GOT_FOCUS

`public static final int GOT_FOCUS`

ID constant for getting input focus `Event`.

HOME

`public static final int HOME`

ID constant for Home key.

INSERT

`public static final int INSERT` ★

ID constant for Insert key.

KEY_ACTION

`public static final int KEY_ACTION`

ID constant for Special Key Down Event.

KEY_ACTION_RELEASE

`public static final int KEY_ACTION_RELEASE`

ID constant for Special Key Up Event.

KEY_PRESS

`public static final int KEY_PRESS`

ID constant for Key Down `Event`.

KEY_RELEASE

`public static final int KEY_RELEASE`

ID constant for Key Up `Event`.

LEFT

`public static final int LEFT`

ID constant for the left arrow key.

LIST_DESELECT

`public static final int LIST_DESELECT`

ID constant for List DeSelect `Event`.

LIST_SELECT

```
public static final int LIST_SELECT
```

ID constant for List Select `Event`.

LOAD_FILE

```
public static final int LOAD_FILE
```

ID constant for File Load `Event`.

LOST_FOCUS

```
public static final int LOST_FOCUS
```

ID constant for losing input focus `Event`.

META_MASK

```
public static final int META_MASK
```

Mask for ALT key.

MOUSE_DOWN

```
public static final int MOUSE_DOWN
```

ID constant for Mouse Down `Event`.

MOUSE_DRAG

```
public static final int MOUSE_DRAG
```

ID constant for Mouse Drag `Event`.

MOUSE_ENTER

```
public static final int MOUSE_ENTER
```

ID constant for Mouse Enter `Event`.

MOUSE_EXIT

```
public static final int MOUSE_EXIT
```

ID constant for Mouse Exit `Event`.

MOUSE_MOVE

```
public static final int MOUSE_MOVE
```

ID constant for Mouse Move `Event`.

MOUSE_UP

```
public static final int MOUSE_UP
```

ID constant for Mouse Up `Event`.

NUM_LOCK

```
public static final int NUM_LOCK ★
```

ID constant for Num Lock key.

PAUSE

```
public static final int PAUSE ★
```

ID constant for Pause key.

PGDN

```
public static final int PGDN
```

ID constant for PageDown key.

PGUP

```
public static final int PGUP
```

ID constant for PageUp key.

PRINT_SCREEN

```
public static final int PRINT_SCREEN ★
```

ID constant for Print Screen key.

RIGHT

```
public static final int RIGHT
```

ID constant for the right arrow key.

SAVE_FILE

```
public static final int SAVE_FILE
```

ID constant for File Save `Event`.

SCROLL_ABSOLUTE

```
public static final int SCROLL_ABSOLUTE
```

ID constant for Absolute Scroll `Event`.

SCROLL_BEGIN

`public static final int SCROLL_ BEGIN` ★

ID constant for Begin Scroll `Event`.

SCROLL_END

`public static final int SCROLL_ END` ★

ID constant for End Scroll `Event`.

SCROLL_LINE_DOWN

`public static final int SCROLL_LINE_DOWN`

ID constant for Line Down Scroll `Event`.

SCROLL_LINE_UP

`public static final int SCROLL_LINE_UP`

ID constant for Line Up Scroll `Event`.

SCROLL_LOCK

`public static final int SCROLL_LOCK` ★

Mask for Scroll Lock key.

SCROLL_PAGE_DOWN

`public static final int SCROLL_PAGE_DOWN`

ID constant for Page Down Scroll `Event`.

SCROLL_PAGE_UP

`public static final int SCROLL_PAGE_UP`

ID constant for Page Up Scroll `Event`.

SHIFT_MASK

`public static final int SHIFT_MASK`

Mask for SHIFT key.

TAB

`public static final int TAB` ★

ID constant for Tab key.

UP

```
public static final int UP
```

ID constant for the up arrow key.

WINDOW_DEICONIFY

```
public static final int WINDOW_DEICONIFY
```

ID constant for Window DeIconify `Event`.

WINDOW_DESTROY

```
public static final int WINDOW_DESTROY
```

ID constant for Window Destroy `Event`.

WINDOW_EXPOSE

```
public static final int WINDOW_EXPOSE
```

ID constant for Window Expose `Event`.

WINDOW_ICONIFY

```
public static final int WINDOW_ICONIFY
```

ID constant for Window Iconify `Event`.

WINDOW_MOVED

```
public static final int WINDOW_MOVED
```

ID constant for Window Move `Event`.

Variables

arg

```
public Object arg
```

A variable argument that is specific to the event type.

clickCount

```
public int clickCount
```

The number of consecutive MOUSE_DOWN events.

evt

```
public Event evt
```

A means of passing a linked list of events as one.

id

```
public int id
```

The ID constant that identifies the `Event` type.

key

```
public int key
```

Integer value of key pressed, or ID constant identifying a special key.

modifiers

```
public int modifiers
```

The state of the shift/alt/control/meta keys, formed by ORing the masks for the appropriate keys.

target

```
public Object target
```

The `Object` that generated the event.

when

```
public long when
```

The time the event happened.

x

```
public int x
```

The x position at which the event happened.

y

```
public int y
```

The y position at which the event happened.

Constructors

`Event`

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

Parameters	*target*	The component to which the `Event` should be delivered
	id	The identifier of `Event`
	arg	The `Object` that is the cause of the event
Description	Constructs an `Event` object with the given values.	

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

Parameters	*target*	The component to which the `Event` should be delivered
	when	The time the event happened
	id	The identifier of `Event`
	x	The x position at which the event happened
	y	The y position at which the event happened
	key	Integer value of key pressed, or a constant identifying a special key
	modifiers	The state of the shift/alt/control/meta keys
Description	Constructs an `Event` object with the given values.	

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

Parameters	*target*	The component to which the `Event` should be delivered
	when	The time the event happened
	id	The identifier of `Event`
	x	The x position at which the event happened
	y	The y position at which the event happened
	key	Integer value of key pressed, or a constant identifying a special key
	modifiers	The state of the shift/alt/control/meta keys
	arg	The `Object` that is the cause of the event
Description	Constructs an `Event` object with the given values.	

Instance Methods

controlDown

```
public boolean controlDown()
```

Returns	`true` if the control key was down when the event was triggered, `false` otherwise.
Description	Checks current settings for modifiers of the `Event`.

metaDown

```
public boolean metaDown()
```

Returns	`true` if the meta key was down when the event was triggered, `false` otherwise.
Description	Checks current settings for modifiers of the `Event`.

shiftDown

```
public boolean shiftDown()
```

Returns	`true` if the shift key was down when the event was triggered, `false` otherwise.
Description	Checks current settings for modifiers of the `Event`.

toString

```
public String toString()
```

Returns	A string representation of the `Event` object.
Overrides	`Object.toString()`

translate

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

Parameters	*x*	Amount to move `Event` in horizontal direction.
	y	Amount to move `Event` in vertical direction.
Description	Translates x and y coordinates of `Event` instance by `x` and `y`.	

Protected Instance Methods

paramString

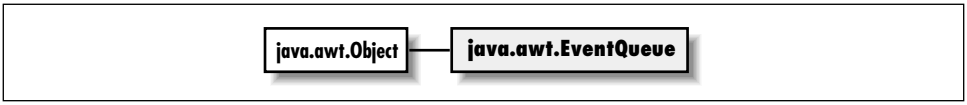
```
protected String paramString()
```

Returns	String with current settings of `Event`.
Description	Helper method for `toString()` to generate string of current settings.

See Also

`AWTEvent`, `Component`, `Object`, `String`

19.21 EventQueue ★



Description

The `EventQueue` class is a facility for queuing Java 1.1 AWT events, either for the system or for some other purpose. You rarely need to create your own event queue; for most purposes, you will want to work with the system's event queue, which you acquire using the `Toolkit`.

Class Definition

```
public class EventQueue extends Object {

  // Constructor
  public EventQueue();

  // Instance Methods
  public synchronized AWTEvent getNextEvent() throws InterruptedException;
  public synchronized AWTEvent peekEvent();
  public synchronized AWTEvent peekEvent (int id);
  public synchronized void postEvent (AWTEvent theEvent);
}
```

Constructor

`EventQueue`

```
public EventQueue()
```

Description Creates an `EventQueue` for your own use.

Instance Methods

`getNextEvent`

```
public synchronized AWTEvent getNextEvent() throws
InterruptedException
```

Throws `InterruptedException`
If the thread is interrupted before an event is posted to the queue.

Returns	`AWTEvent` taken from the event queue.
Description	Removes the next event from the event queue and returns it. If there are no events in the queue, this method will block until another thread posts one.

`peekEvent`

```
public synchronized AWTEvent peekEvent()
```

Returns	Next `AWTEvent` on the event queue.
Description	Returns a reference to the next event on the queue without removing it from the queue.

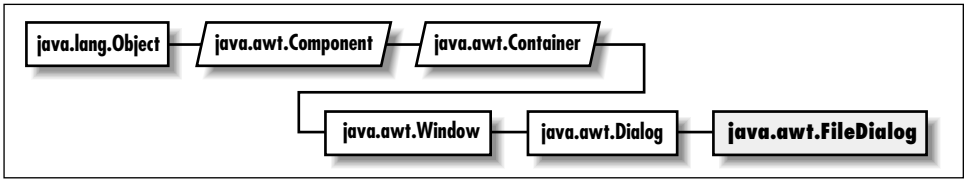
```
public synchronized AWTEvent peekEvent (int id)
```

Parameters	*id*	Type of event to find.
Returns	`AWTEvent` with the given type `id`; `null` if no event with the given type is currently in the queue.	
Description	Returns an event with the given type if one exists, but doesn't remove the event from the queue.	

See Also

`AWTEvent`, `Event`

19.22 FileDialog



Description

The `FileDialog` class provides file selection capabilities for opening or saving files. Because `FileDialog` is a subclass of `Dialog`, a `FileDialog` is always associated with a `Frame` and is hidden by default. `FileDialog`s are always modal (i.e., they always attract all user input). In addition, `FileDialog`s have a load/save mode; the `LOAD` mode is for selecting files for an application to load, `SAVE` is for selecting a filename to save.

Class Definition

```
public class java.awt.FileDialog
    extends java.awt.Dialog {

  // Constants
  public final static int LOAD;
  public final static int SAVE;

  // Constructors
  public FileDialog (Frame parent); ★
  public FileDialog (Frame parent, String title);
  public FileDialog (Frame parent, String title, int mode);

  // Instance Methods
  public void addNotify();
  public String getDirectory();
  public String getFile();
  public FilenameFilter getFilenameFilter();
  public int getMode();
  public synchronized void setDirectory (String directory);
  public synchronized void setFile (String file);
  public synchronized void setFilenameFilter (FilenameFilter filter);
  public void setMode(int mode); ★

  // Protected Instance Methods
  protected String paramString();
}
```

Constants

LOAD

```
public final static int LOAD
```

Constant to specify the `FileDialog`'s load mode.

SAVE

```
public final static int SAVE
```

Constant to specify the `FileDialog`'s save mode.

Constructors

FileDialog

```
public FileDialog (Frame parent) ★
```

Parameters *parent* `Frame` that is to act as parent of `FileDialog`.

Description Constructs a `FileDialog` object in `LOAD` mode.

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

Parameters *parent* `Frame` that is to act as parent of `FileDialog`.
title Title to use for `FileDialog`.

Description Constructs a `FileDialog` object in `LOAD` mode.

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

Parameters *parent* `Frame` that is to act as parent of `Dialog`.
title Title to use for `FileDialog`.
mode The constant `LOAD` or `SAVE`, specifying the dialog's mode.

Description Constructs a `FileDialog` object in the given mode.

Instance Methods

addNotify

```
public void addNotify()
```

Overrides `Dialog.addNotify()`

Description Creates `FileDialog`'s peer for the native platform.

getDirectory

```
public String getDirectory()
```

Returns The current directory for the `FileDialog`.

getFile

```
public String getFile()
```

Returns The current file selected by the `FileDialog`.

getFilenameFilter

```
public FilenameFilter getFilenameFilter()
```

Returns The current filename filter for the `FileDialog`.

getMode

```
public int getMode()
```

Returns The current mode of the `FileDialog`.

setDirectory

```
public synchronized void setDirectory (String directory)
```

Parameters	*directory*	Directory to be displayed by the `FileDialog`.
Description	Changes the directory displayed in the `FileDialog`.	

setFile

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

Parameters	*file*	Initial file string for `FileDialog`.
Description	Change the default file selected by the `FileDialog`.	

setFilenameFilter

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

Parameters	*filter*	Initial filter for `FileDialog`.
Description	Changes the current filename filter of the `FileDialog`.	

setMode

```
public void setMode (int mode) ★
```

Parameters	*mode*	The constant `LOAD` or `SAVE`, specifying the dialog's mode.
Description	Change the mode of the file dialog.	

Protected Instance Methods

paramString

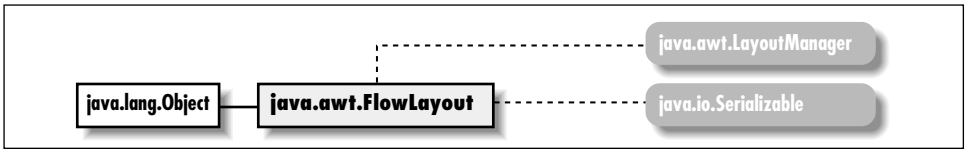
```
protected String paramString()
```

Returns	String with current settings of `FileDialog`.
Overrides	`Dialog.paramString()`
Description	Helper method for `toString()` to generate string of current settings.

See Also

`Dialog`, `FilenameFilter`, `String`

19.23 FlowLayout



Description

The `FlowLayout` `LayoutManager` provides the means to lay out components in a row by row fashion. As each row fills up, the components continue on the next row.

Class Definition

```
public class java.awt.FlowLayout
    extends java.lang.Object
    implements java.awt.LayoutManager, java.io.Serializable {

  // Constants
  public static final int CENTER;
  public static final int LEFT;
  public static final int RIGHT;

  // Constructors
  public FlowLayout();
  public FlowLayout (int alignment);
  public FlowLayout (int alignment, int hgap, int vgap);

  // Instance Methods
  public void addLayoutComponent (String name, Component component);
  public int getAlignment(); ★
  public int getHgap(); ★
  public int getVgap(); ★
  public void layoutContainer (Container target);
  public Dimension minimumLayoutSize (Container target);
  public Dimension preferredLayoutSize (Container target);
  public void removeLayoutComponent (Component component);
  public void setAlignment (int align); ★
  public void setHgap (int hgap); ★
  public void setVgap (int vgap); ★
  public String toString();
}
```

Constants

CENTER

```
public static final int CENTER
```

The default alignment for a `FlowLayout` object; rows of components are centered within the container.

LEFT

```
public static final int LEFT
```

An alignment for a `FlowLayout` object; rows of components start on the left side of the container.

RIGHT

```
public static final int RIGHT
```

An alignment for a `FlowLayout` object; rows of components start on the right side of the container.

Constructors

FlowLayout

```
public FlowLayout()
```

Description Constructs a `FlowLayout` object with `CENTER` alignment.

```
public FlowLayout (int alignment)
```

Parameters *alignment* Alignment of components within the container.
Description Constructs a `FlowLayout` object with the given `alignment`.

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

Parameters *alignment* Alignment of components within container
hgap Horizontal space between each component in a row
vgap Vertical space between each row
Description Constructs a `FlowLayout` object with the given `alignment` and the values specified as the gaps between each component in the container managed by this instance of `FlowLayout`.

Instance Methods

addLayoutComponent

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

Parameters	*name*	Name of component to add.
	component	Actual component being added.
Implements	`LayoutManager.addLayoutComponent()`	
Description	Does nothing.	

getAlignment

```
public int getAlignment() ★
```

Returns	The alignment constant for this `FlowLayout`.

getHgap

```
public int getHgap() ★
```

Returns	The horizontal gap between components.

getVgap

```
public int getVgap() ★
```

Returns	The vertical gap between components.

layoutContainer

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

Parameters	*target*	The container that needs to be redrawn.
Implements	`LayoutManager.layoutContainer()`	
Description	Draws the components contained within the `target` container.	

minimumLayoutSize

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

Parameters	*target*	The container whose size needs to be calculated.
Returns	Minimum `Dimension` of container `target`	
Implements	`LayoutManager.minimumLayoutSize()`	
Description	Calculates minimum size of `target` container.	

preferredLayoutSize

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

Parameters	*target*	The container whose size needs to be calculated.
Returns	Preferred `Dimension` of container `target`	
Implements	`LayoutManager.preferredLayoutSize()`	
Description	Calculates preferred size of `target` container.	

removeLayoutComponent

```
public void removeLayoutComponent (Component component)
```

Parameters	*component*	Component to stop tracking.
Implements	`LayoutManager.removeLayoutComponent()`	
Description	Does nothing.	

setAlignment

```
public void setAlignment(int align) ★
```

Parameters	*alignment*	Alignment of components within container
Description	Sets the alignment for the `FlowLayout`.	

setHgap

```
public void setHgap(int hgap) ★
```

Parameters	*hgap*	The horizontal gap value.
Description	Sets the horizontal gap between components.	

setVgap

```
public void setVgap(int vgap) ★
```

Parameters	*vgap*	The vertical gap value.
Description	Sets the vertical gap between components.	

toString

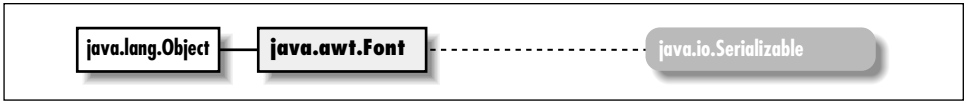
```
public String toString()
```

Returns	A string representation of the `FlowLayout` object.
Overrides	`Object.toString()`

See Also

`Component`, `Container`, `Dimension`, `LayoutManager`, `Object`, `Serializable`, `String`

19.24 Font



Description

The Font class represents a specific font to the system.

Class Definition

```
public class java.awt.Font
    extends java.lang.Object
    implements java.io.Serializable {

  // Constants
  public static final int BOLD;
  public static final int ITALIC;
  public static final int PLAIN;

  // Variables
  protected String name;
  protected int size;
  protected int style;

  // Constructors
  public Font (String name, int style, int size);

  // Class Methods
  public static Font decode (String str); ★
  public static Font getFont (String name)
  public static Font getFont (String name, Font defaultFont)

  // Instance Methods
  public boolean equals (Object object);
  public String getFamily();
  public String getName();
  public FontPeer getPeer(); ★
  public int getSize();
  public int getStyle();
  public int hashCode();
  public boolean isBold();
  public boolean isItalic();
  public boolean isPlain();
  public String toString();
}
```

Constants

BOLD

```
public static final int BOLD
```

Constant for specifying bold fonts.

ITALIC

```
public static final int ITALIC
```

Constant for specifying fonts.

PLAIN

```
public static final int PLAIN
```

Constant for specifying plain fonts.

Variables

name

```
protected String name
```

The font's logical name.

size

```
protected int size
```

The font size; allegedly in points, though probably not true typographer's points.

style

```
protected int style
```

The font style, e.g., bold or italic or a combination thereof.

Constructors

Font

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

Parameters	*name*	The name of the desired font.
	style	One of the style flags (`PLAIN`, `BOLD`, or `ITALIC`) or a combination.
	size	The size of the font to create.
Description	Constructs a `Font` object with the given characteristics.	

Class Methods

decode

```
public static Font decode (String str) ★
```

Parameters	*str*	The string describing the font.
Returns	`Font` instance requested, or default if `str` is invalid.	
Description	Gets font specified by `str`.	

getFont

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

Parameters	*name*	The name of a system property specifying a font to fetch.
Returns	`Font` instance for name requested, or null if `name` is invalid.	
Description	Gets font specified by the system property `name`.	

```
public static Font getFont (String name, Font defaultFont)
```

Parameters	*name*	The name of a system property specifying a font to fetch.
	defaultFont	Font to return if `name` not found in properties.
Returns	`Font` instance of name requested, or `defaultFont` if `name` is invalid	
Description	Gets font specified by the system property `name`.	

Instance Methods

equals

```
public boolean equals (Object object)
```

Parameters	*object*	The object to compare.
Returns	`true` if the objects are equivalent fonts (same name, style, and point size), `false` otherwise.	
Overrides	`Object.equals(Object)`	
Description	Compares two different `Font` instances for equivalence.	

getFamily

```
public String getFamily()
```

Returns	Retrieves the actual name of the font.

getName

```
public String getName()
```

Returns Retrieves the logical name of the font.

getPeer

```
public FontPeer getPeer() ★
```

Returns The font's peer.

getSize

```
public int getSize()
```

Returns Retrieves the `size` parameter from creation

getStyle

```
public int getStyle()
```

Returns Retrieves the `style` parameter from creation.

hashCode

```
public int hashCode()
```

Returns A hashcode to use when using the `Font` as a key in a `Hashtable`.
Overrides `Object.hashCode()`
Description Generates a hashcode for the `Font`.

isBold

```
public boolean isBold()
```

Returns `true` if `Font` style is bold, `false` otherwise.

isItalic

```
public boolean isItalic()
```

Returns `true` if `Font` style is italic, `false` otherwise.

isPlain

```
public boolean isPlain()
```

Returns `true` if `Font` style is neither bold nor italic, `false` otherwise.

toString

```
public String toString()
```

Returns	A string representation of the `Font` object.
Overrides	`Object.toString()`

See Also

`FontMetrics`, `Object`, `Properties`, `String`

19.25 FontMetrics

Description

The `FontMetrics` class provides the means to calculate actual width and height of text if drawn on the screen.

Class Definition

```
public abstract class java.awt.FontMetrics
    extends java.lang.Object
    implements java.io.Serializable {

  // Variables
  protected Font font;

  // Constructors
  protected FontMetrics (Font font);

  // Instance Methods
  public int bytesWidth (byte data[], int offset, int length);
  public int charsWidth (char data[], int offset, int length);
  public int charWidth (char character);
  public int charWidth (int character);
  public int getAscent();
  public int getDescent();
  public Font getFont();
  public int getHeight();
  public int getLeading();
  public int getMaxAdvance();
  public int getMaxAscent();
  public int getMaxDecent();
  public int getMaxDescent();
```

```
  public int[] getWidths();
  public int stringWidth (String string);
  public String toString();
}
```

Variables

font

```
protected Font font
```

The `Font` object whose metrics are represented by this object.

Constructors

FontMetrics

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

Parameters	*font*	The `Font` object whose metrics you want.
Description	Constructs a platform specific `FontMetrics` object for the given `font`.	

Instance Methods

bytesWidth

```
public int bytesWidth (byte data[], int offset, int
length)
```

Parameters	*data[]*	Array of characters to lookup.
	offset	Initial character position.
	length	Number of characters to lookup.
Returns	Advance width of characters in the array, starting with `offset` and ending with `offset+length`, in pixels.	
Throws	*ArrayIndexOutOfBoundsException*	If `offset` or `length` is invalid.

charsWidth

```
public int charsWidth (char data[], int offset, int
length)
```

Parameters	*data[]*	Array of characters to lookup.
	offset	Initial character position.
	length	Number of characters to lookup.
Returns	Advance width of characters in the array, starting with `offset` and ending with `offset+length-1`, in pixels.	

Throws *ArrayIndexOutOfBoundsException*
If `offset` or `length` is invalid.

charWidth

```
public int charWidth (char character)
```

Parameters *character* character to lookup
Returns Advanced pixel width of character.

```
public int charWidth (int character)
```

Parameters *character* int value of character to lookup
Returns Advanced pixel width of character.

getAscent

```
public int getAscent()
```

Returns Amount of space above the baseline required for the tallest character in the font.

getDescent

```
public int getDescent()
```

Returns Amount of space below the baseline required for the lowest descender (e.g., the tail on "p") in the font.

getFont

```
public Font getFont()
```

Returns The `Font` whose metrics are represented by this object.

getHeight

```
public int getHeight()
```

Returns The sum of `getDescent()`, `getAscent()`, and `getLeading()`; recommended total space between baselines.

getLeading

```
public int getLeading()
```

Returns Retrieves recommended amount of space between lines of text.

getMaxAdvance

```
public int getMaxAdvance()
```

Returns Retrieves advance pixel width of widest character in the font.

getMaxAscent

```
public int getMaxAscent()
```

Returns Retrieves maximum amount of space above the baseline required for the tallest character within the font's `FontMetrics`. May differ from `getAscent()` for characters with diacritical marks.

getMaxDecent

```
public int getMaxDecent()
```

Returns Retrieves the maximum amount of space below the baseline required for the deepest character for the font.
Description A misspelling of `getMaxDescent()`.

getMaxDescent

```
public int getMaxDescent()
```

Returns Retrieves the maximum amount of space below the baseline required for the deepest character for the font.

getWidths

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

Returns 255 element array of character widths.
Description Retrieves an integer array of the advance widths of the first 255 characters in the `FontMetrics`' font.

stringWidth

```
public int stringWidth (String string)
```

Parameters *string* Character string to lookup.
Returns Advance pixel width of `string`.

toString

```
public String toString()
```

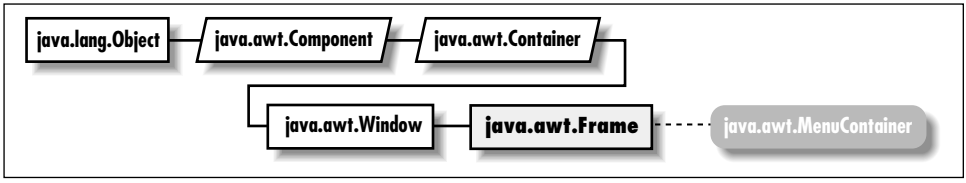
Returns A string representation of the `FontMetrics` object.

Overrides `Object.toString()`

See Also

`Font, Object, String`

19.26 Frame



Description

The `Frame` class is a special type of `Window` that will appear like other high-level programs in your windowing environment. It adds a `MenuBar`, window title, and window gadgets (like resize, maximize, minimize, window menu) to the basic `Window` object. Frames are initially invisible; call `show()` to display them. Frames may also be associated with an `Image` to be used as an icon. The `Frame` class includes many constants to represent different cursor styles. All styles aren't necessarily available on any platform. In 1.1, these constants are defined in `java.awt.Cursor`.

Class Definition

```
public class java.awt.Frame
    extends java.awt.Window
    implements java.awt.MenuContainer {

  // Constants
  public final static int CROSSHAIR_CURSOR;
  public final static int DEFAULT_CURSOR;
  public final static int E_RESIZE_CURSOR;
  public final static int HAND_CURSOR;
  public final static int MOVE_CURSOR;
  public final static int N_RESIZE_CURSOR;
  public final static int NE_RESIZE_CURSOR;
  public final static int NW_RESIZE_CURSOR;
  public final static int S_RESIZE_CURSOR;
  public final static int SE_RESIZE_CURSOR;
  public final static int SW_RESIZE_CURSOR;
```

```
    public final static int TEXT_CURSOR;
    public final static int W_RESIZE_CURSOR;
    public final static int WAIT_CURSOR;

    // Constructors
    public Frame();
    public Frame (String title);

    // Instance Methods
    public void addNotify();
    public synchronized void dispose();
    public int getCursorType(); ☆
    public Image getIconImage();
    public MenuBar getMenuBar();
    public String getTitle();
    public boolean isResizable();
    public synchronized void remove (MenuComponent component);
    public synchronized void setCursor (int cursorType); ☆
    public synchronized void setIconImage (Image image);
    public synchronized void setMenuBar (MenuBar bar);
    public synchronized void setResizable (boolean resizable);
    public synchronized void setTitle (String title);

    // Protected Instance Methods
    protected String paramString();
}
```

Constants

CROSSHAIR_CURSOR

```
public final static int CROSSHAIR_CURSOR
```

Constant representing a cursor that looks like a crosshair.

DEFAULT_CURSOR

```
public final static int DEFAULT_CURSOR
```

Constant representing the platform's default cursor.

E_RESIZE_CURSOR

```
public final static int E_RESIZE_CURSOR
```

Constant representing the cursor for resizing an object on the left.

HAND_CURSOR

```
public final static int HAND_CURSOR
```

Constant representing a cursor that looks like a hand.

MOVE_CURSOR

```
public final static int MOVE_CURSOR
```

Constant representing a cursor used to move an object.

N_RESIZE_CURSOR

```
public final static int N_RESIZE_CURSOR
```

Constant representing a cursor for resizing an object on the top.

NE_RESIZE_CURSOR

```
public final static int NE_RESIZE_CURSOR
```

Constant representing a cursor for resizing an object on the top left corner.

NW_RESIZE_CURSOR

```
public final static int NW_RESIZE_CURSOR
```

Constant representing a cursor for resizing an object on the top right corner.

S_RESIZE_CURSOR

```
public final static int S_RESIZE_CURSOR
```

Constant representing a cursor for resizing an object on the bottom.

SE_RESIZE_CURSOR

```
public final static int SE_RESIZE_CURSOR
```

Constant representing a cursor for resizing an object on the bottom left corner.

SW_RESIZE_CURSOR

```
public final static int SW_RESIZE_CURSOR
```

Constant representing a cursor for resizing an object on the bottom right corner.

TEXT_CURSOR

```
public final static int TEXT_CURSOR
```

Constant representing a cursor used within text.

W_RESIZE_CURSOR

```
public final static int W_RESIZE_CURSOR
```

Constant representing a cursor for resizing an object on the right side.

WAIT_CURSOR

```
public final static int WAIT_CURSOR
```

Constant representing a cursor that indicates the program is busy.

Constructors

Frame

```
public Frame()
```

Description Constructs a `Frame` object, with no title.

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

Parameters *title* Initial title to use for `Frame`.
Description Constructs a `Frame` object, with the given title.

Instance Methods

addNotify

```
public void addNotify()
```

Overrides `Window.addNotify()`
Description Creates `Frame`'s peer and peers of contained components.

dispose

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

Overrides `Window.dispose()`
Description Releases the resources of the `Frame`.

getCursorType

```
public int getCursorType() ☆
```

Returns The constant for the current cursor. Replaced by `Component.getCursor()`

getIconImage

```
public Image getIconImage()
```

Returns The image used as the icon, or `null` if there is no icon for this frame.

getMenuBar

```
public MenuBar getMenuBar()
```

Returns The `Frame`'s current menu bar, or `null` if there is no menu bar for this frame.

getTitle

```
public String getTitle()
```

Returns The current title for the `Frame`, or `null` if there is no title for this frame.

isResizable

```
public boolean isResizable()
```

Returns `true` if resizable, `false` otherwise.

remove

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

Parameters *component* `MenuBar` to remove from `Frame`.
Implements `MenuContainer.remove()`
Description Removes `component` from `Frame` if `component` is the `Frame`'s menu bar.

setCursor

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

Parameters *cursorType* One of `Frame`'s cursor constants.
Throws *IllegalArgumentException*
If `cursorType` invalid.
Description Changes the cursor of the `Frame`. Replaced by `Component.setCursor(Cursor)`.

setIconImage

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

Parameters	*image*	New image to use for the `Frame`'s icon.
Description	Changes the icon's image for the `Frame`.	

setMenuBar

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

Parameters	*bar*	New `MenuBar` to use for the `Frame`.
Description	Changes the menu bar of the `Frame`.	

setResizable

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

Parameters	*resizable*	`true` to make the frame resizable, `false` to prevent resizing.
Description	Changes the resize state of the `Frame`.	

setTitle

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

Parameters	*title*	New title to use for the `Frame`.
Description	Changes the title of the `Frame`.	

Protected Instance Methods

paramString

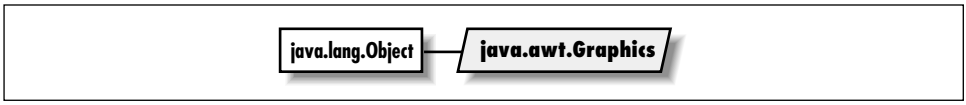
```
protected String paramString()
```

Returns	String with current settings of `Frame`.
Overrides	`Container.paramString()`
Description	Helper method for `toString()` to generate a string of current settings.

See Also

`Container`, `Image`, `MenuBar`, `MenuContainer`, `String`, `Window`

19.27 Graphics



Description

The `Graphics` class is an abstract class that represents an object on which you can draw. The concrete classes that are actually used to represent graphics objects are platform dependent, but because they extend the `Graphics` class, must implement the methods here.

Class Definition

```
public abstract class java.awt.Graphics
    extends java.lang.Object {

  // Constructors
  protected Graphics();

  // Instance Methods
  public abstract void clearRect (int x, int y, int width, int height);
  public abstract void clipRect (int x, int y, int width, int height);
  public abstract void copyArea (int x, int y, int width, int height,
      int deltax, int deltay);
  public abstract Graphics create();
  public Graphics create (int x, int y, int width, int height);
  public abstract void dispose();
  public void draw3DRect (int x, int y, int width, int height,
      boolean raised);
  public abstract void drawArc (int x, int y, int width, int height,
      int startAngle, int arcAngle);
  public void drawBytes (byte text[], int offset, int length,
      int x, int y);
  public void drawChars (char text[], int offset, int length,
      int x, int y);
  public abstract boolean drawImage (Image image, int x, int y,
      ImageObserver observer);
  public abstract boolean drawImage (Image image, int x, int y,
      int width, int height, ImageObserver observer);
  public abstract boolean drawImage (Image image, int x, int y,
      Color backgroundColor, ImageObserver observer);
  public abstract boolean drawImage (Image image, int x, int y,
      int width, int height, Color backgroundColor, ImageObserver observer);
  public abstract boolean drawImage(Image img, int dx1, int dy1,
      int dx2, int dy2, int sx1, int sy1, int sx2, int sy2, ImageObserver
      observer); ★
```

```
  public abstract boolean drawImage(Image img, int dx1, int dy1,
      int dx2, int dy2, int sx1, int sy1, int sx2, int sy2, Color bgcolor,
      ImageObserver observer); ★
  public abstract void drawLine (int x1, int y1, int x2, int y2);
  public abstract void drawOval (int x, int y, int width, int height);
  public abstract void drawPolygon (int xPoints[], int yPoints[],
      int numPoints);
  public void drawPolygon (Polygon p);
  public abstract void drawPolyline(int[ ] xPoints, int[ ] yPoints,
      int nPoints); ★
  public void drawRect (int x, int y, int width, int height);
  public abstract void drawRoundRect (int x, int y, int width,
      int height, int arcWidth, int arcHeight);
  public abstract void drawString (String text, int x, int y);
  public void fill3DRect (int x, int y, int width, int height,
      boolean raised);
  public abstract void fillArc (int x, int y, int width, int height,
      int startAngle, int arcAngle);
  public abstract void fillOval (int x, int y, int width, int height);
  public abstract void fillPolygon (int xPoints[], int yPoints[],
      int numPoints);
  public void fillPolygon (Polygon p);
  public abstract void fillRect (int x, int y, int width, int height);
  public abstract void fillRoundRect (int x, int y, int width,
      int height, int arcWidth, int arcHeight);
  public void finalize();
  public abstract Shape getClip(); ★
  public abstract Rectangle getClipBounds(); ★
  public abstract Rectangle getClipRect();
  public abstract Color getColor();
  public abstract Font getFont();
  public FontMetrics getFontMetrics();
  public abstract FontMetrics getFontMetrics (Font font);
  public abstract void setClip (int x, int y, int width, int height); ★
  public abstract void setClip (Shape clip); ★
  public abstract void setColor (Color color);
  public abstract void setFont (Font font);
  public abstract void setPaintMode();
  public abstract void setXORMode (Color xorColor);
  public String toString();
  public abstract void translate (int x, int y);
}
```

Constructors

Graphics

```
protected Graphics()
```

Description Called by constructors of platform specific subclasses.

Instance Methods

clearRect

```
public abstract void clearRect (int x, int y, int width,
int height)
```

Parameters	*x*	x coordinate of origin of area to clear.
	y	y coordinate of origin of area to clear.
	width	size in horizontal direction to clear.
	height	size in vertical direction to clear.
Description	Resets a rectangular area to the background color.	

clipRect

```
public abstract void clipRect (int x, int y, int width,
int height)
```

Parameters	*x*	x coordinate of origin of clipped area.
	y	y coordinate of origin of clipped area.
	width	size in horizontal direction to clip.
	height	size in vertical direction to clip.
Description	Reduces the drawing area to the intersection of the current drawing area and the rectangular area defined by `x`, `y`, `width`, and `height`.	

copyArea

```
public abstract void copyArea (int x, int y, int width,
int height, int deltax, int deltay)
```

Parameters	*x*	x coordinate of origin of area to copy.
	y	y coordinate of origin of area to copy.
	width	size in horizontal direction to copy.
	height	size in vertical direction to copy.
	deltax	offset in horizontal direction to copy area to.
	deltay	offset in vertical direction to copy area to.
Description	Copies a rectangular area to a new area, whose top left corner is (`x+deltax`, `y+deltay`).	

create

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

Returns	New graphics context.
Description	Creates a second reference to the same graphics context.

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

Parameters	*x*	x coordinate of origin of new graphics context.
	y	y coordinate of origin of new graphics context.
	width	size in horizontal direction.
	height	size in vertical direction.
Returns	New graphics context	
Description	Creates a second reference to a subset of the same graphics context.	

dispose

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

Description	Frees system resources used by graphics context.

draw3DRect

```
public void draw3DRect (int x, int y, int width, int
height, boolean raised)
```

Parameters	*x*	x coordinate of the rectangle origin.
	y	y coordinate of the rectangle origin
	width	Width of the rectangle to draw.
	height	Height of the rectangle to draw.
	raised	Determines if rectangle drawn is raised or not; `true` for a raised rectangle.
Description	Draws an unfilled 3-D rectangle from (`x`, `y`) of size `width` x `height`.	

drawArc

```
public abstract void drawArc (int x, int y, int width, int
height, int startAngle, int arcAngle)
```

Parameters	*x*	x coordinate of the bounding rectangle's origin.
	y	y coordinate of the bounding rectangle's origin
	width	Width of the bounding rectangle for the arc.

	height	Height of the bounding rectangle for the arc.
	startAngle	Angle at which arc begins, in degrees
	arcAngle	length of arc, in degrees
Description	Draws an unfilled arc from `startAngle` to `arcAngle` within bounding rectangle from (`x`, `y`) of size `width` x `height`. Zero degrees is at three o'clock; positive angles are counter clockwise.	

drawBytes

```
public void drawBytes (byte text[], int offset, int
length, int x, int y)
```

Parameters	*text*	Text to draw, as a byte array.
	offset	Starting position within `text` to draw.
	length	Number of bytes to draw.
	x	x coordinate of baseline origin.
	y	y coordinate of baseline origin.
Throws	*ArrayIndexOutOfBoundsException*	
		If `offset` or `length` is invalid.
Description	Draws text on screen, starting with `text[offset]` and ending with `text[offset+length-1]`.	

drawChars

```
public void drawChars (char text[], int offset, int
length, int x, int y)
```

Parameters	*text*	Text to draw, as a char array.
	offset	Starting position within `text` to draw.
	length	Number of bytes to draw.
	x	x coordinate of baseline origin.
	y	y coordinate of baseline origin.
Throws	*ArrayIndexOutOfBoundsException*	
		If `offset` or `length` is invalid.
Description	Draws text on screen, starting with `text[offset]` and ending with `text[offset+length-1]`.	

drawImage

```
public abstract boolean drawImage (Image image, int x, int
y, ImageObserver observer)
```

Parameters	*image*	Image to draw.

	x	x coordinate of image origin.
	y	y coordinate of image origin.
	observer	Object that watches for image information; almost always `this`.
Returns	`true` if the image has fully loaded when the method returns, `false` otherwise.	
Description	Draws image to screen at (`x`, `y`), at its original size. Drawing may be asynchronous. If `image` is not fully loaded when the method returns, `observer` is notified when additional information made available.	

```
public abstract boolean drawImage (Image image, int x, int
y, int width, int height, ImageObserver observer)
```

Parameters	*image*	Image to draw.
	x	x coordinate of image origin.
	y	y coordinate of image origin.
	width	New image size in horizontal direction.
	height	New image size in vertical direction.
	observer	Object that watches for image information; almost always `this`.
Returns	`true` if the image has fully loaded when the method returns, `false` otherwise.	
Description	Draws image to screen at (`x`, `y`), scaled to `width` x `height`. Drawing may be asynchronous. If `image` is not fully loaded when the method returns, `observer` is notified when additional information made available.	

```
public abstract boolean drawImage (Image image, int x, int
y, Color backgroundColor, ImageObserver observer)
```

Parameters	*image*	Image to draw.
	x	x coordinate of image origin.
	y	y coordinate of image origin.
	backgroundColor	Color to show through image where transparent.
	observer	Object that watches for image information; almost always `this`.
Returns	`true` if the image has fully loaded when the method returns, `false` otherwise.	
Description	Draws image to screen at (`x`, `y`), at its original size. Drawing may be asynchronous. If `image` is not fully loaded when the method returns, `observer` is notified when additional information made available. The background color is visible through any transparent pixels.	

```
public abstract boolean drawImage (Image image, int x, int
y, int width, int height, Color backgroundColor,
ImageObserver observer)
```

Parameters	*image*	Image to draw.
	x	x coordinate of image origin.
	y	y coordinate of image origin.
	width	New image size in horizontal direction.
	height	New image size in vertical direction.
	backgroundColor	Color to show through image where transparent.
	observer	Object that watches for image information; almost always `this`.
Returns	`true` if the image has fully loaded when the method returns, `false` otherwise.	
Description	Draws image to screen at (`x`, `y`), scaled to `width` x `height`. Drawing may be asynchronous. If `image` is not fully loaded when the method returns, `observer` is notified when additional information made available. The background color is visible through any transparent pixels.	

```
public abstract boolean drawImage (Image image, int dx1,
int dy1, int dx2, int dy2, int sx1, int sy1, int sx2, int
sy2, ImageObserver observer) ★
```

Parameters	*image*	Image to draw.
	dx1	x coordinate of one corner of destination (device) rectangle.
	dy1	y coordinate of one corner of destination (device) rectangle.
	dx2	x coordinate of the opposite corner of destination (device) rectangle.
	dy2	y coordinate of the opposite corner of destination (device) rectangle.
	sx1	x coordinate of one corner of source (image) rectangle.
	sy1	y coordinate of one corner of source (image) rectangle.
	sx2	x coordinate of the opposite corner of source (image) rectangle.
	sy2	y coordinate of the opposite corner of source (image) rectangle.

	observer	Object that watches for image information; almost always `this`.
Returns	`true` if the image has fully loaded when the method returns, `false` otherwise.	
Description	Draws the part of image described by `dx1`, `dy1`, `dx2`, and `dy2` to the screen into the rectangle described by `sx1`, `sy1`, `sx2`, and `sy2`. Drawing may be asynchronous. If `image` is not fully loaded when the method returns, `observer` is notified when additional information is made available.	

```
public abstract boolean drawImage (Image image, int dx1,
int dy1, int dx2, int dy2, int sx1, int sy1, int sx2, int
sy2, Color backgroundColor, ImageObserver observer) ★
```

Parameters	*image*	Image to draw.
	dx1	x coordinate of one corner of destination (device) rectangle.
	dy1	y coordinate of one corner of destination (device) rectangle.
	dx2	x coordinate of the opposite corner of destination (device) rectangle.
	dy2	y coordinate of the opposite corner of destination (device) rectangle.
	sx1	x coordinate of one corner of source (image) rectangle.
	sy1	y coordinate of one corner of source (image) rectangle.
	sx2	x coordinate of the opposite corner of source (image) rectangle.
	sy2	y coordinate of the opposite corner of source (image) rectangle.
	backgroundColor	Color to show through image where transparent.
	observer	Object that watches for image information; almost always `this`.
Returns	`true` if the image has fully loaded when the method returns, `false` otherwise.	
Description	Draws the part of image described by `dx1`, `dy1`, `dx2`, and `dy2` to the screen into the rectangle described by `sx1`, `sy1`, `sx2`, and `sy2`. Drawing may be asynchronous. If `image` is not fully loaded when the method returns, `observer` is notified when additional information made available. The background color is visible through any transparent pixels.	

drawLine

```
public abstract void drawLine (int x1, int y1, int x2, int
y2)
```

Parameters	*x1*	x coordinate of one point on line.
	y1	y coordinate of one point on line.
	x2	x coordinate of the opposite point on line.
	y2	y coordinate of the opposite point on line.
Description	Draws a line connecting `(x1, y1)` and `(x2, y2)`.	

drawOval

```
public abstract void drawOval (int x, int y, int width,
int height)
```

Parameters	*x*	x coordinate of bounding rectangle origin.
	y	y coordinate of bounding rectangle origin
	width	Width of bounding rectangle to draw in.
	height	Height of bounding rectangle to draw in.
Description	Draws an unfilled oval within bounding rectangle from (`x`, `y`) of size `width` x `height`.	

drawPolygon

```
public abstract void drawPolygon (int xPoints[], int
yPoints[], int numPoints)
```

Parameters	*xPoints[]*	The array of x coordinates for each point.
	yPoints[]	The array of y coordinates for each point.
	numPoints	The number of elements in both `xPoints` and `yPoints` arrays to use.
Description	Draws an unfilled polygon based on first `numPoints` elements in `xPoints` and `yPoints`.	

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

Parameters	*p*	Points of object to draw.
Description	Draws an unfilled polygon based on points within the `Polygon` `p`.	

drawPolyline

```
public abstract void drawPolyline (int xPoints[], int
yPoints[], int nPoints) ★
```

Parameters	*xPoints[]*	The array of x coordinates for each point.
	yPoints[]	The array of y coordinates for each point.
	nPoints	The number of elements in both `xPoints` and `yPoints` arrays to use.
Description	Draws a series of line segments based on first `numPoints` elements in `xPoints` and `yPoints`.	

drawRect

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

Parameters	*x*	x coordinate of rectangle origin.
	y	y coordinate of rectangle origin
	width	Width of rectangle to draw.
	height	Height of rectangle to draw.
Description	Draws an unfilled rectangle from (`x`, `y`) of size `width` x `height`.	

drawRoundRect

```
public abstract void drawRoundRect (int x, int y, int
width, int height, int arcWidth, int arcHeight)
```

Parameters	*x*	x coordinate of bounding rectangle origin.
	y	y coordinate of bounding rectangle origin
	width	Width of rectangle to draw.
	height	Height of rectangle to draw.
	arcWidth	Width of arc of rectangle corner.
	arcHeight	Height of arc of rectangle corner.
Description	Draws an unfilled rectangle from (`x`, `y`) of size `width` x `height` with rounded corners.	

drawString

```
public abstract void drawString (String text, int x, int
y)
```

Parameters	*text*	Text to draw.
	x	x coordinate of baseline origin.
	y	y coordinate of baseline origin.
Description	Draws text on screen.	

fill3DRect

```
public void fill3DRect (int x, int y, int width, int
height, boolean raised)
```

Parameters	*x*	x coordinate of rectangle origin.
	y	y coordinate of rectangle origin
	width	Width of rectangle to draw.
	height	Height of rectangle to draw.
	raised	`true` to draw a rectangle that appears raised; `false` to draw a rectangle that appears depressed.
Description	Draws a filled 3-D rectangle from (`x`, `y`) of size `width` x `height`.	

fillArc

```
public abstract void fillArc (int x, int y, int width, int
height, int startAngle, int arcAngle)
```

Parameters	*x*	x coordinate of bounding rectangle origin.
	y	y coordinate of bounding rectangle origin
	width	Width of bounding rectangle to draw in.
	height	Height of bounding rectangle to draw in.
	startAngle	Starting angle of arc.
	arcAngle	The extent of the arc, measured from startAngle
Description	Draws a filled arc from `startAngle` to `arcAngle` within bounding rectangle from (`x`, `y`) of size `width` x `height`. Zero degrees is at three o'clock; positive angles are counter clockwise.	

fillOval

```
public abstract void fillOval (int x, int y, int width,
int height)
```

Parameters	*x*	x coordinate of bounding rectangle origin.
	y	y coordinate of bounding rectangle origin
	width	Width of bounding rectangle to draw in.
	height	Height of bounding rectangle to draw in.
Description	Draws filled oval within bounding rectangle from (`x`, `y`) of size `width` x `height`.	

fillPolygon

```
public abstract void fillPolygon (int xPoints[], int
yPoints[], int numPoints)
```

Parameters	*xPoints[]*	The array of x coordinates for each point.
	yPoints[]	The array of y coordinates for each point.
	numPoints	The number of elements in both `xPoints` and `yPoints` arrays to use.
Throws	*ArrayIndexOutOfBoundsException*	If `numPoints > xPoints.length` or `numPoints > yPoints.length`.
Description	Draws filled polygon based on first `numPoints` elements in `xPoints` and `yPoints`.	

```
public void fillPolygon (Polygon p)
```

Parameters	*p*	Points of object to draw.
Description	Draws filled polygon based on points within the `Polygon p`.	

fillRect

```
public abstract void fillRect (int x, int y, int width,
int height)
```

Parameters	*x*	x coordinate of rectangle origin.
	y	y coordinate of rectangle origin
	width	Width of rectangle to draw.
	height	Height of rectangle to draw.
Description	Draws filled rectangle from (`x`, `y`) of size `width` x `height`.	

fillRoundRect

```
public abstract void fillRoundRect (int x, int y, int
width, int height, int arcWidth, int arcHeight)
```

Parameters	*x*	x coordinate of bounding rectangle origin.
	y	y coordinate of bounding rectangle origin
	width	Width of rectangle to draw.
	height	Height of rectangle to draw.
	arcWidth	Width of arc of rectangle corner.
	arcHeight	Height of arc of rectangle corner.
Description	Draws a filled rectangle from (`x`, `y`) of size `width` x `height` with rounded corners.	

finalize

```
public void finalize()
```

Overrides `Object.finalize()`
Description Tells the garbage collector to dispose of graphics context.

getClip

```
public abstract Shape getClip () ★
```

Returns Shape describing the clipping are of the graphics context.

getClipBounds

```
public abstract Rectangle getClipBounds() ★
```

Returns Rectangle describing the clipping area of the graphics context.

getClipRect

```
public abstract Rectangle getClipRect() ☆
```

Returns Replaced by `getClipBounds()`.

getColor

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

Returns The current drawing `Color` of the graphics context.

getFont

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

Returns The current `Font` of the graphics context.

getFontMetrics

```
public FontMetrics getFontMetrics()
```

Returns The `FontMetrics` of the current font of the graphics context.

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

Parameters *font* Font to get metrics for.
Returns The `FontMetrics` of the given font for the graphics context.

setClip

```
public abstract void setClip (int x, int y, int width, int
height) ★
```

Parameters	*x*	x coordinate of rectangle
	y	y coordinate of rectangle
	width	width of rectangle
	height	height of rectangle
Description	Changes current clipping region to the specified rectangle.	

```
public abstract void setClip (Shape clip) ★
```

Parameters	*clip*	The new clipping shape.
Description	Changes current clipping region to the specified shape.	

setColor

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

Parameters	*color*	New color.
Description	Changes current drawing color of graphics context.	

setFont

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

Parameters	*font*	New font.
Description	Changes current font of graphics context.	

setPaintMode

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

Description	Changes painting mode to normal mode.

setXORMode

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

Parameters	*xorColor*	XOR mode drawing color.
Description	Changes painting mode to XOR mode; in this mode, drawing the same object in the same color at the same location twice has no net effect.	

toString

```
public String toString()
```

Returns	A string representation of the `Graphics` object.
Overrides	`Object.toString()`

translate

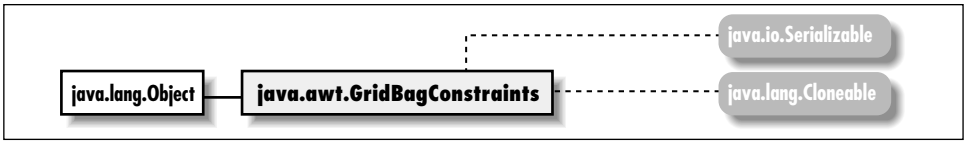
```
public void translate (int x, int y)
```

Parameters	*x*	x coordinate of new drawing origin.
	y	y coordinate of new drawing origin.
Description	Moves the origin of drawing operations to (x, y).	

See Also

Color, Font, FontMetrics, Image, ImageObserver, Object, Polygon, Rectangle, Shape, String

19.28 GridBagConstraints



Description

The GridBagConstraints class provides the means to control the layout of components within a Container whose LayoutManager is GridBagLayout.

Class Definition

```
public class java.awt.GridBagConstraints
    extends java.lang.Object
    implements java.lang.Cloneable, java.io.Serializable {

  // Constants
  public final static int BOTH;
  public final static int CENTER;
  public final static int EAST;
  public final static int HORIZONTAL;
  public final static int NONE;
  public final static int NORTH;
  public final static int NORTHEAST;
  public final static int NORTHWEST;
  public final static int RELATIVE;
  public final static int REMAINDER;
  public final static int SOUTH;
  public final static int SOUTHEAST;
  public final static int SOUTHWEST;
  public final static int VERTICAL;
  public final static int WEST;
```

```
    // Variables
    public int anchor;
    public int fill;
    public int gridheight;
    public int gridwidth;
    public int gridx;
    public int gridy;
    public Insets insets;
    public int ipadx;
    public int ipady;
    public double weightx
    public double weighty

    // Constructors
    public GridBagConstraints();

    // Instance Methods
    public Object clone();
}
```

Constants

BOTH

```
public final static int BOTH
```

Constant for possible `fill` value.

CENTER

```
public final static int CENTER
```

Constant for possible `anchor` value.

EAST

```
public final static int EAST
```

Constant for possible `anchor` value.

HORIZONTAL

```
public final static int HORIZONTAL
```

Constant for possible `fill` value.

NONE

```
public final static int NONE
```

Constant for possible `fill` value.

NORTH

```
public final static int NORTH
```

Constant for possible `anchor` value.

NORTHEAST

```
public final static int NORTHEAST
```

Constant for possible `anchor` value.

NORTHWEST

```
public final static int NORTHWEST
```

Constant for possible `anchor` value.

RELATIVE

```
public final static int RELATIVE
```

Constant for possible `gridx`, `gridy`, `gridwidth`, or `gridheight` value.

REMAINDER

```
public final static int REMAINDER
```

Constant for possible `gridwidth` or `gridheight` value.

SOUTH

```
public final static int SOUTH
```

Constant for possible `anchor` value.

SOUTHEAST

```
public final static int SOUTHEAST
```

Constant for possible `anchor` value.

SOUTHWEST

```
public final static int SOUTHWEST
```

Constant for possible `anchor` value.

VERTICAL

```
public final static int VERTICAL
```

Constant for possible `fill` value.

WEST

```
public final static int WEST
```

Constant for possible `anchor` value.

Variables

anchor

```
public int anchor
```

Specifies the alignment of the component in the event that it is smaller than the space allotted for it by the layout manager; e.g., `CENTER` centers the object within the region.

fill

```
public int fill
```

The component's resize policy if additional space available.

gridheight

```
public int gridheight
```

Number of columns a component occupies.

gridwidth

```
public int gridwidth
```

Number of rows a component occupies.

gridx

```
public int gridx
```

Horizontal grid position at which to add component.

gridy

```
public int gridy
```

Vertical grid position at which to add component.

insets

```
public Insets insets
```

Specifies the outer padding around the component.

ipadx

```
public int ipadx
```

Serves as the internal padding within the component in both the right and left directions.

ipady

```
public int ipady
```

Serves as the internal padding within the component in both the top and bottom directions.

weightx

```
public double weightx
```

Represents the percentage of extra horizontal space that will be given to this component if there is additional space available within the container.

weighty

```
public double weighty
```

Represents the percentage of extra vertical space that will be given to this component if there is additional space available within the container.

Constructors

GridBagConstraints

```
public GridBagConstraints()
```

Description Constructs a `GridBagConstraints` object.

Instance Methods

clone

```
public Object clone()
```

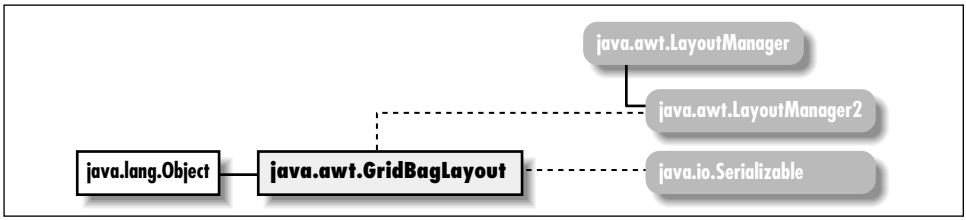
Returns A new instance of `GridBagConstraints` with same values for constraints.

Overrides `Object.clone()`

See Also

`Cloneable`, `GridBagLayout`, `Insets`, `Object`, `Serializable`

19.29 GridBagLayout



Description

The `GridBagLayout` `LayoutManager` provides the means to layout components in a flexible grid-based display model.

Class Definition

```
public class java.awt.GridBagLayout
    extends java.lang.Object
    implements java.awt.LayoutManager2, java.io.Serializable {

  // Protected Constants
  protected static final MAXGRIDSIZE;
  protected static final MINSIZE;
  protected static final PREFERREDSIZE;

  // Variables
  public double columnWeights[];
  public int columnWidths[];
  public int rowHeights[];
  public double rowWeights[];

  // Protected Variables
  protected Hashtable comptable;
  protected GridBagConstraints defaultConstraints;
  protected GridBagLayoutInfo layoutInfo;

  // Constructors
  public GridBagLayout();

  // Instance Methods
  public void addLayoutComponent (Component comp, Object constraints); ★
  public void addLayoutComponent (String name, Component component);
  public GridBagConstraints getConstraints (Component component);
```

```
    public abstract float getLayoutAlignmentX(Container target); ★
    public abstract float getLayoutAlignmentY(Container target); ★
    public int[][] getLayoutDimensions();
    public Point getLayoutOrigin();
    public double[][] getLayoutWeights();
    public abstract void invalidateLayout(Container target); ★
    public void layoutContainer (Container target);
    public Point location (int x, int y);
    public abstract Dimension maximumLayoutSize(Container target); ★
    public Dimension minimumLayoutSize (Container target);
    public Dimension preferredLayoutSize (Container target);
    public void removeLayoutComponent (Component component);
    public void setConstraints (Component component,
        GridBagConstraints constraints);
    public String toString();

    // Protected Instance Methods
    protected void AdjustForGravity (GridBagConstraints constraints,
        Rectangle r);
    protected void ArrangeGrid (Container target);
    protected GridBagLayoutInfo GetLayoutInfo (Container target,
        int sizeFlag);
    protected Dimension GetMinSize (Container target,
        GridBagLayoutInfo info);
    protected GridBagConstraints lookupConstraints (Component comp);
}
```

Protected Constants

MAXGRIDSIZE

```
protected static final MAXGRIDSIZE
```

Maximum number of rows and columns within container managed by `GridBagLayout`.

MINSIZE

```
protected static final MINSIZE
```

Used for internal sizing purposes.

PREFERREDSIZE

```
protected static final PREFERREDSIZE
```

Used for internal sizing purposes.

Variables

columnWeights

```
public double[] columnWeights
```

The `weightx` values of the components in the row with the most elements.

columnWidths

```
public int[] columnWidths
```

The `width` values of the components in the row with the most elements.

rowHeights

```
public int[] rowHeights
```

The `height` values of the components in the column with the most elements.

rowWeights

```
public double[] rowWeights
```

The `weighty` values of the components in the column with the most elements.

Protected Variables

comptable

```
protected Hashtable comptable
```

Internal table to manage components.

defaultConstraints

```
protected GridBagConstraints defaultConstraints
```

Constraints to use for `Components` that have none.

layoutInfo

```
protected GridBagLayoutInfo layoutInfo
```

Internal information about the `GridBagLayout`.

Constructors

GridBagLayout

```
public GridBagLayout()
```

Description Constructs a `GridBagLayout` object.

Instance Methods

addLayoutComponent

```
public void addLayoutComponent (Component comp, Object
constraints) ★
```

Parameters	*comp*	The component being added.
	constraints	An object describing the constraints on this component.
Implements	`LayoutManager2.addLayoutComponent()`	
Description	Adds the component `comp` to container subject to the given `constraints`. This is a more generalized version of `addLayoutComponent(String, Component)`. It corresponds to `java.awt.Container's add(Component, Object)`.	

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

Parameters	*name*	Name of component to add.
	component	Actual component being added.
Implements	`LayoutManager.addLayoutComponent()`	
Description	Does nothing.	

getConstraints

```
public GridBagConstraints getConstraints (Component
component)
```

Parameters	*component*	Component whose constraints are desired
Returns	`GridBagConstraints` for component requested.	

getLayoutAlignmentX

```
public abstract float getLayoutAlignmentX (Container
target) ★
```

Parameters	*target*	The container to inspect.
Returns	The value .5 for all containers.	
Description	This method returns the preferred alignment of the given container `target`. A return value of 0 is left aligned, .5 is centered, and 1 is right aligned.	

getLayoutAlignmentY

```
public abstract float getLayoutAlignmentY (Container
target) ★
```

Parameters	*target*	The container to inspect.
Returns	The value .5 for all containers.	
Description	This method returns the preferred alignment of the given container `target`. A return value of 0 is top aligned, .5 is centered, and 1 is bottom aligned.	

getLayoutDimensions

```
public int[][] getLayoutDimensions()
```

Returns	Returns two single dimension arrays as a multi-dimensional array. Index 0 is an array of widths (`columnWidths` instance variable), while index 1 is an array of heights (`rowHeights` instance variable).

getLayoutOrigin

```
public Point getLayoutOrigin()
```

Returns	Returns the origin of the components within the `Container` whose `LayoutManager` is `GridBagLayout`.

getLayoutWeights

```
public double[][] getLayoutWeights()
```

Returns	Returns two single dimension arrays as a multi-dimensional array. Index 0 is an array of columns weights (`columnWeights` instance variable), while index 1 is an array of row weights (`rowWeights` instance variable).

invalidateLayout

```
public abstract void invalidateLayout (Container target)
★
```

Parameters	*target*	The container to invalidate.
Description	Does nothing.	

layoutContainer

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

Parameters	*target*	The container that needs to be redrawn.

Implements	`LayoutManager.layoutContainer()`
Description	Draws components contained within `target`.

location

```
public Point location (int x, int y)
```

Parameters	*x*	The x coordinate of the grid position to find.
	y	The y coordinate of the grid position to find.
Returns	Returns the grid element under the location provided at position (`x`, `y`) in pixels. Note that the returned Point uses the `GridBagLayout`'s grid for its coordinate space.	
Description	Locates the grid position in the `Container` under the given location.	

maximumLayoutSize

```
public abstract Dimension maximumLayoutSize (Container
target) ★
```

Parameters	*target*	The container to inspect.
Returns	A `Dimension` whose horizontal and vertical components are `Integer.MAX_VALUE`.	
Description	For `GridBagLayout`, a maximal `Dimension` is always returned.	

minimumLayoutSize

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

Parameters	*target*	The container whose size needs to be calculated.
Returns	Minimum `Dimension` of container target.	
Implements	`LayoutManager.minimumLayoutSize()`	
Description	Calculates minimum size of `target` container.	

preferredLayoutSize

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

Parameters	*target*	The container whose size needs to be calculated.
Returns	Preferred `Dimension` of container target	
Implements	`LayoutManager.preferredLayoutSize()`	
Description	Calculates preferred size of `target` container.	

removeLayoutComponent

```
public void removeLayoutComponent (Component component)
```

Parameters	*component*	Component to stop tracking.
Implements	`LayoutManager.removeLayoutComponent()`	
Description	Does nothing.	

setConstraints

```
public void setConstraints (Component component,
GridBagConstraints constraints)
```

Parameters	*component*	Component to set constraints for
	constraints	Constraints for component
Description	Changes the `GridBagConstraints` on `component` to those provided.	

toString

```
public String toString()
```

Returns	A string representation of the `GridBagLayout` object.
Overrides	`Object.toString()`

Protected Instance Methods

AdjustForGravity

```
protected void AdjustForGravity (GridBagConstraints
constraints, Rectangle r)
```

Parameters	*constraints*	Constraints to use for adjustment of Rectangle.
	r	Rectangular area that needs to be adjusted.
Description	Helper routine for laying out a cell of the grid. The routine adjusts the values for `r` based upon the `constraints`.	

ArrangeGrid

```
protected void ArrangeGrid (Container target)
```

Parameters	*target*	`Container` to layout.
Description	Helper routine that does the actual arrangement of components in `target`.	

GetLayoutInfo

```
protected GridBagLayoutInfo GetLayoutInfo (Container
target, int sizeFlag)
```

Parameters	*target*	`Container` to get information about.
	sizeFlag	One of the constants `MINSIZE` or `PREFERRED-SIZE`.
Returns	Returns an internal class used to help size the container.	

`GetMinSize`

```
protected Dimension GetMinSize (Container target,
GridBagLayoutInfo info)
```

Parameters	*target*	`Container` to calculate size.
	info	Specifics about the container's constraints.
Returns	Minimum `Dimension` of container target based on info.	
Description	Helper routine for calculating size of container.	

`lookupConstraints`

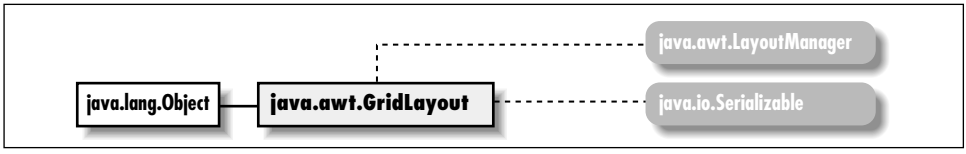
```
protected GridBagConstraints lookupConstraints (Component
comp)
```

Parameters	*comp*	`Component` in question.
Returns	A reference to the `GridBagConstraints` object for this component.	
Description	Helper routine for calculating size of container.	

See Also

`Component`, `Container`, `Dimension`, `GridBagConstraints`, `Hashtable`, `LayoutManager`, `LayoutManager2`, `Object`, `Point`, `Rectangle`, `String`

19.30 GridLayout



Description

The `GridLayout` `LayoutManager` provides the means to layout components in a grid of rows and columns.

Class Definition

```
public class java.awt.GridLayout
        extends java.lang.Object
        implements java.awt.LayoutManager, java.io.Serializable
 {

// Constructors
  public GridLayout(); ★
  public GridLayout (int rows, int cols);
  public GridLayout (int rows, int cols, int hgap, int vgap);

// Instance Methods
  public void addLayoutComponent (String name, Component component);
  public int getColumns(); ★
  public int getHgap(); ★
  public int getRows(); ★

  public int getVgap(); ★
  public void layoutContainer (Container target);
  public Dimension minimumLayoutSize (Container target);
  public Dimension preferredLayoutSize (Container target);
  public void removeLayoutComponent (Component component);
  public int setColumns(int cols); ★
  public int setHgap(int hgap); ★
  public int setRows(int rows); ★

  public int setVgap(int vgap); ★
  public String toString();
}
```

Constructors

GridLayout

```
public GridLayout() ★
```

Description Constructs a `GridLayout` object with a default single row and one column per component.

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

Parameters		
	rows	Requested number of rows in container.
	cols	Requested number of columns in container.

Description Constructs a `GridLayout` object with the requested number of rows and columns. Note that the actual number of rows and columns depends on the number of objects in the layout, not the constructor's parameters.

```
public GridLayout (int rows, int cols, int hgap, int vgap)
```

Parameters	*rows*	Requested number of rows in container.
	cols	Requested number of columns in container.
	hgap	Horizontal space between each component in a row.
	vgap	Vertical space between each row.
Description	Constructs a `GridLayout` object with the requested number of `rows` and `columns` and the values specified as the gaps between each component. Note that the actual number of rows and columns depends on the number of objects in the layout, not the constructor's parameters.	

Instance Methods

addLayoutComponent

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

Parameters	*name*	Name of component to add.
	component	Actual component being added.
Implements	`LayoutManager.addLayoutComponent()`	
Description	Does nothing.	

getColumns

```
public int getColumns() ★
```

Returns The number of columns.

getHgap

```
public int getHgap() ★
```

Returns The horizontal gap for this `GridLayout` instance.

getRows

```
public int getRows() ★
```

Returns The number of rows.

getVgap

```
public int getVgap() ★
```

Returns The vertical gap for this `GridLayout` instance.

layoutContainer

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

Parameters *target* The container that needs to be redrawn.
Implements `LayoutManager.layoutContainer()`
Description Draws the components contained within the `target`.

minimumLayoutSize

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

Parameters *target* The container whose size needs to be calculated.
Returns Minimum `Dimension` of the container `target`.
Implements `LayoutManager.minimumLayoutSize()`
Description Calculates the minimum size of the `target` container.

preferredLayoutSize

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

Parameters *target* The container whose size needs to be calculated.
Returns Preferred `Dimension` of the container `target`.
Implements `LayoutManager.preferredLayoutSize()`
Description Calculates the preferred size of the `target` container.

removeLayoutComponent

```
public void removeLayoutComponent (Component component)
```

Parameters *component* Component to stop tracking.
Implements `LayoutManager.removeLayoutComponent()`
Description Does nothing.

setColumns

```
public void setColumns(int cols) ★
```

Parameters *cols* The new number of columns.
Description Sets the number of columns.

setHgap

```
public void setHgap(int hgap) ★
```

Parameters	*hgap*	The horizontal gap value.
Description	Sets the horizontal gap between components.	

setRows

```
public void setRows(int rows) ★
```

Parameters	*rows*	The new number of rows.
Description	Sets the number of rows.	

setVgap

```
public void setVgap(int vgap) ★
```

Parameters	*vgap*	The vertical gap value.
Description	Sets the vertical gap between components.	

toString

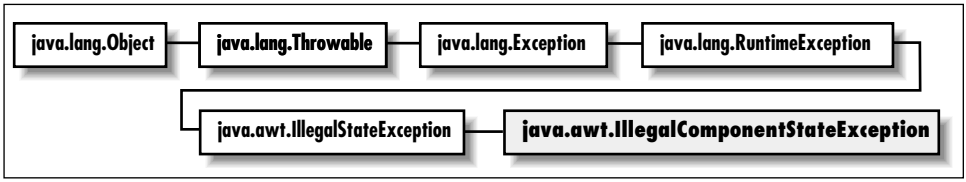
```
public String toString()
```

Returns	A string representation of the `GridLayout` object.
Overrides	`Object.toString()`

See Also

`Component`, `Container`, `Dimension`, `LayoutManager`, `Object`, `String`

19.31 IllegalComponentStateException ★



Description

An `Exception` indicating that a `Component` was not in an appropriate state to perform a requested action.

Class Definition

```
public class java.awt.IllegalComponentStateException
    extends java.lang.IllegalStateException {

  // Constructors
  public IllegalComponentStateException();
  public IllegalComponentStateException (String s);
}
```

Constructors

`IllegalComponentStateException`

```
public IllegalComponentStateException()
```

Description Constructs the exception object with no detail message.

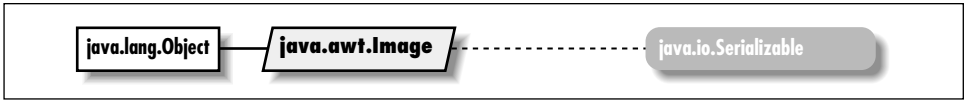
```
public IllegalComponentStateException (String s)
```

Parameters *s* Detail message
Description Constructs the exception object with the given detail message.

See Also

`Exception`, `String`

19.32 Image



Description

The `Image` class represents a displayable object maintained in memory. Because `Image` is an abstract class, you never work with the `Image` class itself, but with a platform specific subclass. However, you should never need to know what that subclass is. To draw on an `Image`, get its graphics context.

Class Definition

```
public abstract class java.awt.Image
    extends java.lang.Object
    implements java.io.Serializable {

  // Constants
  public final static int SCALE_AREA_AVERAGING; ★
  public final static int SCALE_DEFAULT; ★
  public final static int SCALE_FAST; ★
```

```
    public final static int SCALE_REPLICATE; ★
    public final static int SCALE_SMOOTH; ★
    public final static Object UndefinedProperty;

    // Instance Methods
    public abstract void flush();
    public abstract Graphics getGraphics();
    public abstract int getHeight (ImageObserver observer);
    public abstract Object getProperty (String name, ImageObserver observer);
    public Image getScaledInstance (int width, int height, int hints); ★
    public abstract ImageProducer getSource();
    public abstract int getWidth (ImageObserver observer);
}
```

Constants

SCALE_AREA_AVERAGING

```
public final static int SCALE_AREA_AVERAGING ★
```

Flag that requests use of `AreaAveragingScaleFilter`.

SCALE_DEFAULT

```
public final static int SCALE_DEFAULT ★
```

Flag that requests use of the default image scaling algorithm.

SCALE_FAST

```
public final static int SCALE_FAST ★
```

Flag that requests use of an image scaling algorithm that is faster rather than smoother.

SCALE_REPLICATE

```
public final static int SCALE_REPLICATE ★
```

Flag that requests use of ReplicateScaleFilter.

SCALE_SMOOTH

```
public final static int SCALE_SMOOTH ★
```

Flag that requests use of an image scaling algorithm that is smoother rather than faster.

UndefinedProperty

```
public final static Object UndefinedProperty
```

Possible return object from `getProperty()`.

Instance Methods

`flush`

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

Description Resets image to initial state.

`getGraphics`

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

Throws *ClassCastException*
If image created from file or URL.

Returns The graphics context of the image.

Description Gets the graphics context of the image for drawing.

`getHeight`

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

Parameters *observer* An image observer; usually the `Component` on which the image is rendered.

Returns Image height, or -1 if the height is not yet available.

`getProperty`

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

Parameters *name* Name of the property to fetch.
observer An image observer; usually the `Component` on which the image is rendered.

Returns `Object` representing the requested property, `null`, or `UndefinedProperty`.

Throws *ArrayIndexOutOfBoundsException*
If `offset` or `length` is invalid.

Description Retrieves a property from the image's private property list.

`getScaledInstance`

```
public Image getScaledInstance (int width, int height, int
hints) ★
```

Parameters	*width*	The width for the scaled image. Use -1 to preserve the aspect ratio with reference to `height`.
	height	The height for the scaled image. Use -1 to preserve the aspect ratio with reference to `width`.
	hints	One or more of the `SCALE_` constants.
Returns	The scaled image. It may be loaded asynchronously, even if the original image was fully loaded.	
Description	Creates a copy of an image, scaled to `width` x `height` and using an algorithm chosen based on the `hints` given.	

`getSource`

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

Returns	The `ImageProducer` of the image.

`getWidth`

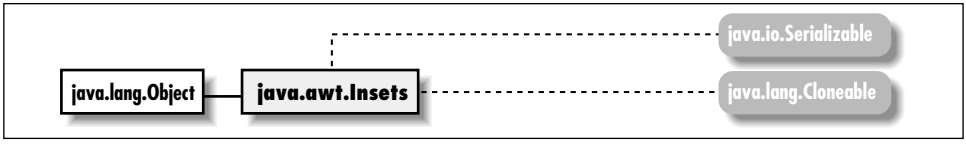
```
public abstract int getWidth (ImageObserver observer)
```

Parameters	*observer*	An image observer; usually the `Component` on which the image is rendered.
Returns	Image width, or -1 if the width is not yet available.	

See Also

`Graphics`, `ImageObserver`, `ImageProducer`, `Object`, `Properties`, `String`

19.33 Insets



Description

The `Insets` class provides a way to encapsulate the layout margins of the four different sides of a `Container`.

Class Definition

```
public class java.awt.Insets
    extends java.lang.Object
    implements java.io.Serializable, java.lang.Cloneable {

  // Variables
```

```
    public int bottom;
    public int left;
    public int right;
    public int top;

    // Constructors
    public Insets (int top, int left, int bottom, int right);

    // Instance Methods
    public Object clone();
    public boolean equals (Object obj); ★
    public String toString();
}
```

Variables

bottom

```
public int bottom
```

The border width for the bottom of a `Container`.

left

```
public int left
```

The border width for the left side of a `Container`.

right

```
public int right
```

The border width for the right side of a `Container`.

top

```
public int top
```

The border width for the top of a `Container`.

Constructors

Insets

```
public Insets (int top, int left, int bottom, int right)
```

Parameters		
	top	The border width for the top of a `Container`.
	left	The border width for the left side of a `Container`.
	bottom	The border width for the bottom of a `Container`.

	right	The border width for the right side of a `Container`.
Description	Constructs an `Insets` object with the appropriate border settings.	

Instance Methods

`clone`

```
public Object clone()
```

Returns	Clone of original object.
Overrides	`Object.clone()`
Description	Creates a copy of the original instance of an object.

`equals`

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

Parameters	*obj*	The object to be tested.
Returns	`true` if the objects are equal; `false` otherwise.	
Overrides	`Object.equals(Object)`	
Description	Tests two `Insets` objects for equality.	

`toString`

```
public String toString()
```

Returns	A string representation of the `Insets` object.
Overrides	`Object.toString()`

See Also

`Cloneable`, `Container`, `Object`, `Serializable`, `String`

19.34 ItemSelectable ★

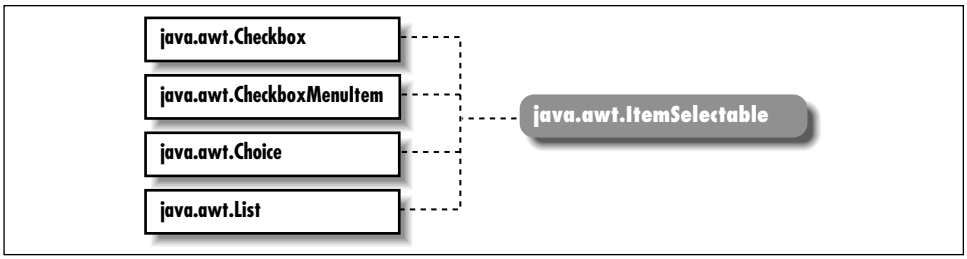
Description

An interface that describes an object that has one or more items that can be selected.

Interface Definition

```
public abstract interface ItemSelectable {

  // Instance Methods
  public abstract void addItemListener (ItemListener l);
  public abstract Object[] getSelectedObjects();
  public abstract void removeItemListener (ItemListener l);
```




```
}
```

Interface Methods

`addItemListener`

```
public abstract void addItemListener (ItemListener l)
```

Parameters *l* The listener to be added.
Description Adds a listener for `ItemEvent` objects.

`getSelectedObjects`

```
public abstract Object[] getSelectedObjects()
```

Description This method returns an array containing `Objects` representing the items that are currently selected. If no items are selected, `null` is returned.

`removeItemListener`

```
public abstract void removeItemListener (ItemListener l)
```

Parameters *l* The listener to be removed.
Description Removes the specified `ItemListener` so it will not receive `ItemEvent` objects.

See Also

`Checkbox`, `CheckboxMenuItem`, `Choice`, `ItemEvent`, `ItemListener`, `List`

19.35 Label

Description

The `Label` is a `Component` that displays a single line of static text.

Class Definition

```
public class java.awt.Label
    extends java.awt.Component {

  // Constants
  public static final int CENTER;
  public static final int LEFT;
  public static final int RIGHT;

  // Constructors
  public Label();
  public Label (String label);
  public Label (String label, int alignment);

  // Instance Methods
  public void addNotify();
  public int getAlignment();
  public String getText();
  public synchronized void setAlignment (int alignment);
  public synchronized void setText (String label);

  // Protected Instance Methods
  protected String paramString();
}
```

Constants

CENTER

```
public static final int CENTER
```

Description Constant to center text within the label.

LEFT

```
public static final int LEFT
```

Description Constant to left justify text within the label.

RIGHT

```
public static final int RIGHT
```

Description Constant to right justify text within the label.

Constructors

Label

```
public Label()
```

Description Constructs a `Label` object with the text centered within the label.

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

Parameters *label* The text for the label
Description Constructs a `Label` object with the text `label` centered within the label.

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

Parameters *label* The text for the label
alignment The alignment for the label; one of the constants `CENTER`, `LEFT`, or `RIGHT`.
Throws *IllegalArgumentException* If alignment is not one of `CENTER`, `LEFT`, or `RIGHT`.
Description Constructs a `Label` object, with a given `alignment` and text of `label`.

Instance Methods

addNotify

```
public void addNotify()
```

Overrides `Component.addNotify()`
Description Creates `Label`'s peer.

getAlignment

```
public int getAlignment()
```

Returns Current alignment.

getText

```
public String getText()
```

Returns Current text of `Label`.

setAlignment

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

Parameters *alignment* New alignment for `Label`; `CENTER`, `LEFT`, or `RIGHT`.
Throws *IllegalArgumentException* If `alignment` is not one of `CENTER`, `LEFT`, or `RIGHT`.
Description Changes the current alignment of `Label`.

setText

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

Parameters *label* New text for `Label`.
Description Changes the current text of `Label`.

Protected Instance Methods

paramString

```
protected String paramString()
```

Returns String with current settings of `Label`.
Overrides `Component.paramString()`
Description Helper method for `toString()` to generate string of current settings.

See Also

`Component`, `String`

19.36 LayoutManager

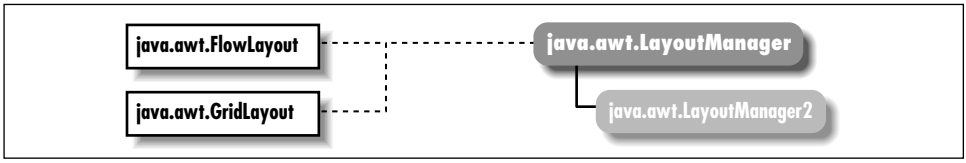
Description

`LayoutManager` is an interface that defines the responsibilities of an object that wants to lay out `Components` to the display in a `Container`.

Interface Definition

```
public abstract interface java.awt.LayoutManager {

  // Interface Methods
  public abstract void addLayoutComponent (String name,
      Component component);
```




```
  public abstract void layoutContainer (Container target);
  public abstract Dimension minimumLayoutSize (Container target);
  public abstract Dimension preferredLayoutSize (Container target);
  public abstract void removeLayoutComponent (Component component);
}
```

Interface Methods

addLayoutComponent

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

Parameters	*name*	Name of component to add.
	component	Actual component being added.
Description	Called when you call `Container.add(String, Component)` to add an object to a container.	

layoutContainer

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

Parameters	*target*	The container who needs to be redrawn.
Description	Called when `target` needs to be redrawn.	

minimumLayoutSize

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

Parameters	*target*	The container whose size needs to be calculated.
Returns	Minimum `Dimension` of the container `target`	
Description	Called when the minimum size of the `target` container needs to be calculated.	

preferredLayoutSize

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

Parameters	*target*	The container whose size needs to be calculated.
Returns	Preferred `Dimension` of the container `target`	
Description	Called when the preferred size of the `target` container needs to be calculated.	

`removeLayoutComponent`

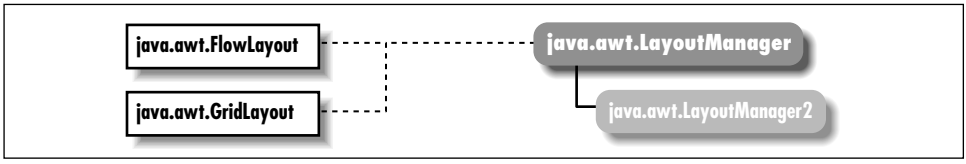
```
public abstract void removeLayoutComponent (Component
component)
```

Parameters	*component*	Component to no longer track.
Description	Called when you call `Container.remove(Component)` to remove a component from the layout.	

See Also

`Component`, `Container`, `FlowLayout`, `GridLayout`, `Object`, `String`

19.37 LayoutManager2 ⋆



Description

`LayoutManager2` is an extension of `LayoutManager`. It provides a more generalized way to add components to a container, as well as more sizing and alignment methods.

Interface Definition

```
public abstract interface java.awt.LayoutManager2
    extends java.awt.LayoutManager {

  // Interface Methods
  public abstract void addLayoutComponent (Component comp,
      Object constraints);
  public abstract float getLayoutAlignmentX(Container target);
  public abstract float getLayoutAlignmentY(Container target);
  public abstract void invalidateLayout(Container target);
  public abstract Dimension maximumLayoutSize(Container target);
}
```

Interface Methods

addLayoutComponent

```
public abstract void addLayoutComponent (Component comp,
Object constraints)
```

Parameters	*comp*	Component to add.
	constraints	Constraints on the component.
Description	Called to add an object to a container. This is slightly more generic than `LayoutManager`'s `addLayoutComponent(String, Component)`.	

getLayoutAlignmentX

```
public abstract float getLayoutAlignmentX (Container
target)
```

Parameters	*target*	The container to inspect.
Returns	A value between 0 and 1.	
Description	This method returns the preferred alignment of the given container `target`. A return value of 0 is left aligned, .5 is centered, and 1 is right aligned.	

getLayoutAlignmentY

```
public abstract float getLayoutAlignmentY (Container
target)
```

Parameters	*target*	The container to inspect.
Returns	A value between 0 and 1.	
Description	This method returns the preferred alignment of the given container `target`. A return value of 0 is top aligned, .5 is centered, and 1 is bottom aligned.	

invalidateLayout

```
public abstract void invalidateLayout (Container target)
```

Parameters	*target*	The container to invalidate.
Description	Sophisticated layout managers may cache information to improve performance. This method can be used to signal the manager to discard any cached information and start fresh.	

maximumLayoutSize

```
public abstract Dimension maximumLayoutSize (Container
target)
```

Returns	The maximum size of `target`.	
Parameters	*target*	The container to inspect.
Description	This method returns the maximum size of `target` using this layout manager.	

See Also

`BorderLayout`, `CardLayout`, `Component`, `Container`, `GridBagLayout`, `Object`, `String`

19.38 List

Description

The `List` is a `Component` that provides a scrollable list of choices to select from. A `List` can be in one of two modes: single selection mode, in which only one item may be selected at a time; and multiple selection mode, in which several items may be selected at one time. A list does not necessarily display all of the choices at one time; one of the constructors lets you specify the number of choices to display simultaneously. Although the changes in 1.1 are extensive, almost all of them can be boiled down to (1) using the 1.1 event model, and (2) standardizing method names (e.g. set/get pairs).

Class Definition

```
public class java.awt.List
    extends java.awt.Component
    implements java.awt.ItemSelectable {

  // Constructors
  public List();
  public List (int rows); ★
  public List (int rows, boolean multipleSelections);

  // Instance Methods
  public void add (String item); ★
  public synchronized void add (String item, int index); ★
  public void addActionListener (ActionListener l); ★
  public void addItem (String item);
```

```
  public synchronized void addItem (String item, int index); ☆
  public void addItemListener (ItemListener l); ★
  public void addNotify();
  public boolean allowsMultipleSelections(); ☆
  public synchronized void clear(); ☆
  public int countItems(); ☆
  public synchronized void delItem (int position);
  public synchronized void delItems (int start, int end); ☆
  public synchronized void deselect (int index);
  public String getItem (int index);
  public int getItemCount(); ★
  public synchronized String[] getItems(); ★
  public Dimension getMinimumSize(); ★
  public Dimension getMinimumSize (int rows); ★
  public Dimension getPreferredSize(); ★
  public Dimension getPreferredSize (int rows); ★
  public int getRows();
  public synchronized int getSelectedIndex();
  public synchronized int[] getSelectedIndexes();
  public synchronized String getSelectedItem();
  public synchronized String[] getSelectedItems();
  public Object[] getSelectedObjects(); ★
  public int getVisibleIndex();
  public boolean isIndexSelected(int index); ★
  public boolean isMultipleMode(); ★
  public boolean isSelected (int index); ☆
  public synchronized void makeVisible (int index);
  public Dimension minimumSize(); ☆
  public Dimension minimumSize (int rows); ☆
  public Dimension preferredSize(); ☆
  public Dimension preferredSize (int rows); ☆
  public synchronized void remove (int position); ★
  public synchronized void remove (String item); ★
  public void removeActionListener (ActionListener l); ★
  public synchronized void removeAll(); ★
  public void removeItemListener (ItemListener l); ★
  public void removeNotify();
  public synchronized void replaceItem (String newItem, int index);
  public synchronized void select (int position);
  public synchronized void setMultipleMode (boolean b); ★
  public synchronized void setMultipleSelections (boolean value); ☆

  // Protected Instance Methods
  protected String paramString();
  protected void processActionEvent (ActionEvent e); ★
  protected void processEvent (AWTEvent e); ★
  protected void processItemEvent (ItemEvent e); ★
}
```

Constructors

List

```
public List()
```

Description Constructs a `List` object in single-selection mode.

```
public List (int rows) ★
```

Parameters *rows* Requested number of rows to display.
Description Constructs a `List` object with the specified number of rows, in single-selection mode.

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

Parameters *rows* Requested number of rows to display.
multipleSelections `true` to allow multiple selections; `false` to select one item at a time.
Description Constructs a `List` object.

Instance Methods

add

```
public void add (String item) ★
```

Parameters *item* Text for entry to add.
Description Adds a new entry to the available choices.

```
public synchronized void add (String item, int index) ★
```

Parameters *item* Text for entry to add.
index Position at which to add entry; the first entry has an `index` of zero.
Description Adds a new entry to the available choices at the designated position.

addActionListener

```
public void addActionListener (ActionListener l) ★
```

Parameters *l* An object that implements the `ActionListener` interface.
Description Add a listener for the action event.

addItem

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

Parameters	*item*	Text for entry to add.
Description	Replaced by `add(String)`.	

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

Parameters	*item*	Text for entry to add.
	index	Position at which to add entry; the first entry has an `index` of zero.
Description	Replaced by `add(String, int)`.	

addItemListener

```
public void addItemListener (ItemListener l)
```
★

Parameters	*l*	The listener to be added.
Implements	`ItemSelectable.addItemListener(ItemListener l)`	
Description	Adds a listener for the `ItemEvent` objects this `List` fires off.	

addNotify

```
public void addNotify()
```

Overrides	`Component.addNotify()`
Description	Creates `List`'s peer.

allowsMultipleSelections

```
public boolean allowsMultipleSelections()
```
☆

Returns	`true` if multi-selection active, `false` otherwise. Replaced by `isMultipleMode()`.

clear

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

Description	Clears all the entries out of the `List`. Replaced by `removeAll()`.

countItems

```
public int countItems()
```
☆

Returns	Number of items in the `List`. Replaced by `getItemCount()`.

delItem

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

Parameters	*position*	Position of item to delete.
Description	Removes a single entry from the `List`. Replaced by `remove(int)` and `remove(String)`.	

delItems

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

Parameters	*start*	Starting position of entries to delete.
	end	Ending position of entries to delete.
Description	Removes a set of entries from the `List`.	

deselect

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

Parameters	*index*	Position to deselect.
Description	Deselects the entry at the designated position, if selected.	

getItem

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

Parameters	*index*	Position of entry to get.
Throws	*ArrayIndexOutOfBoundsException*	If `index` is invalid.
Returns	String for entry at given position.	

getItemCount

```
public int getItemCount() ★
```

Returns	Number of items in the `List`.

getItems

```
public String[] getItems() ★
```

Returns	The string items in the `List`.

getMinimumSize

```
public Dimension getMinimumSize() ★
```

Returns	The minimum dimensions of the `List`.

```
public Dimension getMinimumSize (int rows) ★
```

Parameters *rows* Number of rows within `List` to size.
Returns The minimum dimensions of a `List` of the given size.

getPreferredSize

```
public Dimension getPreferredSize() ★
```

Returns The preferred dimensions of the `List`.

```
public Dimension getPreferredSize (int rows) ★
```

Parameters *rows* Number of rows within `List` to size.
Returns The preferred dimensions of a `List` of the given size.

getRows

```
public int getRows()
```

Returns Returns number of rows requested to be displayed in `List`.

getSelectedIndex

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

Returns Position of currently selected entry, or -1 if nothing is selected, or if multiple entries are selected.

getSelectedIndexes

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

Returns An array whose elements are the indices of the currently selected entries.

getSelectedItem

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

Returns Currently selected entry as a String, or null if nothing is selected, or if multiple entries are selected.

getSelectedItems

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

Returns An array of strings whose elements are the labels of the currently selected entries.

getSelectedObjects

```
public Object[] getSelectedObjects() ★
```

Implements	`ItemSelectable.getSelectedObjects()`
Returns	An array of strings whose elements are the labels of the currently selected entries.

getVisibleIndex

```
public int getVisibleIndex()
```

Returns	The last index from a call to `makeVisible()`.

isIndexSelected

```
public boolean isIndexSelected (int index) ★
```

Parameters	*index*	Position to check.
Returns	`true` if index selected, `false` otherwise.	
Description	Checks to see if a particular entry is currently selected.	

isMultipleMode

```
public boolean isMultipleMode() ★
```

Returns	`true` if multiple selection is allowed, `false` otherwise.

isSelected

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

Parameters	*index*	Position to check.
Returns	`true` if index selected, `false` otherwise.	
Description	Checks to see if a particular entry is currently selected. Replaced by `isIndexSelected(int)`.	

makeVisible

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

Parameters	*index*	Position to make visible on screen.
Description	Ensures an item is displayed on the screen.	

minimumSize

```
public Dimension minimumSize() ☆
```

Returns	The minimum dimensions of the `List`. Replaced by `getMinimumSize()`.

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

Parameters	*rows*	Number of rows within `List` to size.
Returns	The minimum dimensions of a `List` of the given size. Replaced by `getMinimumSize(int)`.	

preferredSize

```
public Dimension preferredSize() ☆
```

Returns	The preferred dimensions of the `List`. Replaced by `getPreferredSize()`.

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

Parameters	*rows*	Number of rows within `List` to size.
Returns	The preferred dimensions of a `List` of the given size. Replaced by `getPreferredSize(int)`.	

remove

```
public synchronized void remove (int position) ★
```

Parameters	*position*	Position of item to remove.
Description	Removes a single entry from the `List`.	

```
public synchronized void remove (String item) ★
```

Parameters	*item*	Item to remove.
Throws	*IllegalArgumentException*	If `item` is not in the `List`.
Description	Removes a single entry from the `List`.	

removeActionListener

```
public void removeActionListener (ActionListener l) ★
```

Parameters	*l*	One of this `List`'s `ActionListeners`.
Description	Remove an action event listener.	

removeAll

```
public synchronized removeAll() ★
```

Description	Removes all items from the `List`.

removeItemListener

```
public void removeItemListener (ItemListener l) ★
```

Parameters	*l*	The listener to be removed.
Implements	`ItemSelectable.removeItemListener (ItemListener l)`	
Description	Removes the specified `ItemListener` so it will not receive `ItemEvent` objects from this `List`.	

removeNotify

```
public void removeNotify()
```

Description Destroys the peer of the `List`.

replaceItem

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

Parameters	*newItem*	Label for entry to add.
	index	Position of entry to replace.
Description	Replaces the contents at a particular position with a new entry.	

select

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

Parameters	*position*	Position to make selected entry.
Description	Makes the given entry the selected one for the `List`.	

setMultipleMode

```
public synchronized void setMultipleMode (boolean b) ★
```

Parameters	*b*	`true` to enable multiple selections; `false` to disable multiple selections.
Description	Changes `List`'s selection mode based upon flag.	

setMultipleSelections

```
public synchronized void setMultipleSelections
(boolean value) ☆
```

Parameters	*value*	`true` to enable multiple selections; `false` to disable multiple selections.
Description	Changes `List`'s selection mode based upon flag. Replaced by `setMultipleMode(boolean)`.	

Protected Instance Methods

paramString

```
protected String paramString()
```

Returns	String with current settings of `List`.
Overrides	`Component.paramString()`
Description	Helper method for `toString()` to generate string of current settings.

processActionEvent

```
protected void processActionEvent (ActionEvent e) ★
```

Parameters	*e*	The action event to process.
Description	Action events are passed to this method for processing. Normally, this method is called by `processEvent()`.	

processEvent

```
protected void processEvent (AWTEvent e) ★
```

Parameters	*e*	The event to process.
Description	Low-level `AWTEvents` are passed to this method for processing.	

processItemEvent

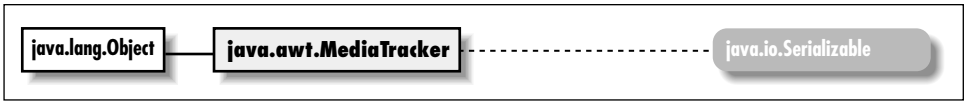
```
protected void processItemEvent(ItemEvent e) ★
```

Parameters	*e*	The item event to process.
Description	Item events are passed to this method for processing. Normally, this method is called by `processEvent()`.	

See Also

`Component`, `Dimension`, `ItemSelectable`, `String`

19.39 MediaTracker



Description

The `MediaTracker` class assists in the loading of multimedia objects across the network. It can be used to wait until an object (or group of objects) has been loaded completely. Tracked objects are assigned to groups; if there is more than one object in a group, you can only track the behavior of the group as a whole (i.e., it isn't possible to track an individual object unless it is the only object in its group). Currently (1.0.2 and 1.1) `MediaTracker` only works for `Image` objects; future releases may extend `MediaTracker` to other multi-media types.

Class Definition

```
public abstract class java.awt.MediaTracker
    extends java.lang.Object
    implements java.io.Serializable {

  // Constants
  public static final int ABORTED;
  public static final int COMPLETE;
  public static final int ERRORED;
  public static final int LOADING;

  // Constructors
  public MediaTracker (Component component);

  // Instance Methods
  public void addImage (Image image, int id);
  public synchronized void addImage (Image image, int id, int width, int height);
  public boolean checkAll();
  public synchronized boolean checkAll (boolean load);
  public boolean checkID (int id);
  public synchronized boolean checkID (int id, boolean load);
  public synchronized Object[] getErrorsAny();
  public synchronized Object[] getErrorsID (int id);
  public synchronized boolean isErrorAny();
  public synchronized boolean isErrorID (int id);
  public synchronized void removeImage(Image image); ★
  public synchronized void removeImage(Image image, int id); ★
  public synchronized void removeImage(Image image, int id, int width, int height);
  public synchronized int statusAll (boolean load);
  public synchronized int statusID (int id, boolean load);
  public void waitForAll() throws InterruptedException;
  public synchronized boolean waitForAll (long ms) throws InterruptedException;
  public void waitForID (int id) throws InterruptedException;
  public synchronized boolean waitForID (int id, long ms) throws InterruptedExceptio
}
```

Constants

ABORTED

```
public static final int ABORTED
```

Flag that indicates that the loading process aborted while loading a particular image.

COMPLETE

```
public static final int COMPLETE
```

Flag that indicates a particular image loaded successfully.

ERRORED

```
public static final int ERRORED
```

Flag that indicates an error occurred while a particular image was loading.

LOADING

```
public static final int LOADING
```

Flag that indicates a particular image is still loading.

Constructors

MediaTracker

```
public MediaTracker (Component component)
```

Parameters	*component*	`Component` that eventually renders objects being tracked.
Description	Constructs an `MediaTracker` object.	

Instance Methods

addImage

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

Parameters	*image*	`Image` to track.
	id	ID of a group.
Description	Tells a MediaTracker to track the loading of `image`, placing the image in the group identified by `id`.	

```
public synchronized void addImage (Image image, int id,
int width, int height)
```

Parameters	*image*	Image to track.
	id	ID of a group.

	width	Eventual rendering width.
	height	Eventual rendering height.
Description	Tells a MediaTracker to track the loading of `image`, which will be scaled to the given `height` and `width`, placing the image in the group identified by `id`.	

checkAll

```
public boolean checkAll()
```

Returns	`true` if images completed loading (successfully or unsuccessfully), `false` otherwise.	
Description	Determines if all images have finished loading.	

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

Parameters	*load*	Flag to force image loading to start.
Returns	`true` if all images have completed loading (successfully or unsuccessfully), `false` otherwise.	
Description	Determines if all images have finished loading; the `load` parameter may be used to force images to start loading.	

checkID

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

Parameters	*id*	ID of a group.
Returns	`true` if all images have completed loading (successfully or unsuccessfully), `false` otherwise.	
Description	Determines if all images with the given ID tag have finished loading.	

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

Parameters	*id*	ID of a group.
	load	Flag to force image loading to start.
Returns	`true` if all images have completed loading (successfully or unsuccessfully), `false` otherwise.	
Description	Determines if all images with the given ID tag have finished loading; the `load` parameter may be used to force images to start loading.	

getErrorsAny

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

Returns	An array of objects managed by this media tracker that encountered a loading error.
Description	Checks to see if any media encountered an error while loading.

getErrorsID

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

Parameters	*id*	ID of a group.
Returns	An array of objects that encountered a loading error.	
Description	Checks to see if any media with the given ID tag encountered an error while loading.	

isErrorAny

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

Returns	`true` if an error occurred, `false` otherwise.
Description	Checks to see if any media monitored by this media tracker encountered an error while loading.

isErrorID

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

Parameters	*id*	ID of a group.
Returns	`true` if error happened, `false` otherwise.	
Description	Checks to see if any media in the given group encountered an error while loading.	

removeImage

```
public synchronized void removeImage (Image image) ★
```

Parameters	*image*	The image to remove.
Description	Removes the specified image from this `MediaTracker`.	

```
public synchronized void removeImage (Image image, int id)
★
```

Parameters	*image*	The image to remove.
	id	ID of a group.
Description	Removes the specified image from this `MediaTracker`. Only instances matching the given `id` will be removed.	

```
public synchronized void removeImage (Image image, int id,
int width, int height) ★
```

Parameters	*image*	The image to remove.
	id	ID of a group.
	width	Width of the scaled image, or -1 for unscaled.
	height	Height of the scaled image, or -1 for unscaled.
Description	Removes the specified image from this `MediaTracker`. Only instances matching the given `id` and scale sizes will be removed.	

`statusAll`

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

Parameters	*load*	Flag to force image loading to start.
Returns	`MediaTracker` status flags ORed together.	
Description	Checks load status of all the images monitored by this media tracker; the `load` parameter may be used to force images to start loading.	

`statusID`

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

Parameters	*id*	ID of a group.
	load	Flag to force image loading to start.
Returns	`MediaTracker` status flags ORed together.	
Description	Checks load status of all the images in the given group; the `load` parameter may be used to force images to start loading.	

`waitForAll`

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

Throws	*InterruptedException*	If waiting interrupted.
Description	Waits for all the images monitored by this media tracker to load.	

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

Parameters	*ms*	Time to wait for loading.
Throws	*InterruptedException*	If waiting interrupted.

Returns	`true` if images fully loaded, `false` otherwise.
Description	Waits at most `ms` milliseconds for all images monitored by this media tracker to load.

`waitForID`

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

Parameters	*id*	ID of a group.
Throws	*InterruptedException*	If waiting interrupted.
Description	Waits for images in the given group to load.	

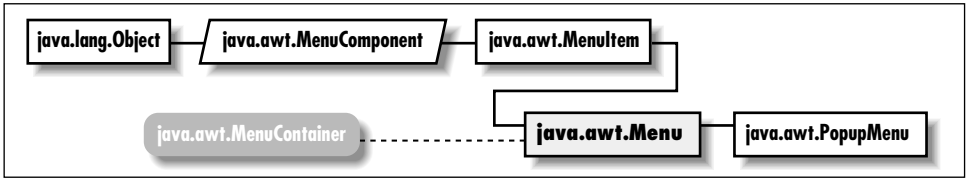
```
public synchronized boolean waitForID (int id, long ms)
throws InterruptedException
```

Parameters	*id*	ID of a group.
	ms	Maximum time to wait for loading.
Throws	*InterruptedException*	If waiting interrupted.
Returns	`true` if images fully loaded, `false` otherwise.	
Description	Waits at most `ms` milliseconds for the images in the given group to load.	

See Also

`Component`, `Image`, `Object`

19.40 Menu



Description

The `Menu` class represents a group of `MenuItem` objects. `Menus` themselves are menu items, allowing you to build multi-level menus. `Menus` are always attached to `MenuBars`, which currently can only belong to frames.

Class Definition

```
public class java.awt.Menu
    extends java.awt.MenuItem
    implements java.awt.MenuContainer {

  // Constructors
  public Menu(); ★
  public Menu (String label);
  public Menu (String label, boolean tearOff);

  // Instance Methods
  public synchronized MenuItem add (MenuItem item);
  public void add (String label);
  public void addNotify();
  public void addSeparator();
  public int countItems(); ☆
  public MenuItem getItem (int index);
  public int getItemCount(); ★
  public void insert (String label, int index); ★
  public synchronized void insert (MenuItem menuitem, int index); ★
  public void insertSeparator (int index); ★
  public boolean isTearOff();
  public String paramString(); ★
  public synchronized void remove (int index);
  public synchronized void remove (MenuComponent component);
  public synchronized void removeAll(); ★
  public void removeNotify();
}
```

Constructors

Menu

`public Menu()` ★

Description Constructs a `Menu` object.

`public Menu (String label)`

Parameters *label* Text that appears on `Menu`.
Description Constructs a `Menu` object with the given label.

`public Menu (String label, boolean tearOff)`

Parameters *label* Text that appears on `Menu`.
tearOff `true` to create a tear-off menu, `false` otherwise.
Description Constructs a `Menu` object; this will be a tear-off menu if `tearOff` is set to `true`.

Instance Methods

add

```
public synchronized MenuItem add (MenuItem item)
```

Parameters	*item*	A `MenuItem` to add to the `Menu`.
Returns	Item just added.	
Description	Adds a new item to a `Menu`.	

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

Parameters	*label*	Text for a `MenuItem`
Description	Constructs a new `MenuItem` object with the given label, and adds it to a `Menu`.	

addNotify

```
public void addNotify()
```

Overrides	`MenuItem.addNotify()`
Description	Creates a `Menu` peer, and peers for all `MenuItem` objects that appear on it.

addSeparator

```
public void addSeparator()
```

Description	Adds a separator bar to the `Menu`.

countItems

```
public int countItems() ☆
```

Returns	The number of items on the menu. Replaced by `getItemCount()`.

getItem

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

Parameters	*index*	The position of the `MenuItem` to fetch; the first item has index 0.
Returns	The `MenuItem` at the designated position.	

getItemCount

```
public int getItemCount() ★
```

Returns	The number of items on the menu.

insert

```
public void insert (String label, int index) ★
```

Parameters	*label*	The label for the new item.
	index	The position for the new item.
Description	Adds a new item to this menu.	

```
public synchronized void insert (MenuItem menuitem, int
index) ★
```

Parameters	*menuitem*	The item to add.
	index	The position for the new item.
Throws	*IllegalArgumentException*	If `index` is less than zero.
Description	Adds a new item to this menu.	

insertSeparator

```
public void insertSeparator (int index) ★
```

Parameters	*index*	The position for the separator.
Throws	*IllegalArgumentException*	If `index` is less than zero.
Description	Adds a separator to this menu.	

isTearOff

```
public boolean isTearOff()
```

Returns	`true` if the menu is a tear-off menu, `false` otherwise.

paramString

```
public String paramString() ★
```

Returns	String with current settings of `Menu`.
Overrides	`MenuItem.paramString()`
Description	Helper method for `toString()` to generate string of current settings.

remove

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

Parameters	*index*	The position of the `MenuItem` to remove.
Description	Removes an item from the `Menu`.	

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

Parameters *component* The element to remove.
Implements `MenuContainer.remove()`
Description Removes an item from the Menu.

`removeAll`

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

Description Removes all items from the `Menu`.

`removeNotify`

```
public void removeNotify()
```

Description Destroys `Menu` peer, and peers for all `MenuItem` objects that appear on it.

See Also

`Frame`, `MenuComponent`, `MenuContainer`, `MenuItem`, `String`

19.41 MenuBar

Description

A `MenuBar` holds menus. `MenuBars` are always attached to frames, and displayed on the top line of the `Frame`. One menu in a `MenuBar` may be designated a "help" menu.

Class Definition

```
public class java.awt.MenuBar
    extends java.awt.MenuComponent
    implements java.awt.MenuContainer {

  // Constructors
  public MenuBar();

  // Instance Methods
  public synchronized Menu add (Menu m);
  public void addNotify();
  public int countMenus(); ☆
  public void deleteShortcut (MenuShortcut s); ★
```

```
  public Menu getHelpMenu();
  public Menu getMenu (int index);
  public int getMenuCount(); ★
  public MenuItem getShortcutMenuItem (MenuShortcut s); ★
  public synchronized void remove (int index);
  public synchronized void remove (MenuComponent component);
  public void removeNotify();
  public synchronized void setHelpMenu (Menu m);
  public synchronized Enumeration shortcuts(); ★
}
```

Constructors

MenuBar

```
public MenuBar()
```

Description Constructs a `MenuBar` object.

Instance Methods

add

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

Parameters *m* A `Menu` to add to `MenuBar`.
Returns Item just added.
Description Adds a new menu to the `MenuBar`.

addNotify

```
public void addNotify()
```

Description Creates `MenuBar`'s peer and peers of contained menus.

countMenus

```
public int countMenus() ☆
```

Returns The number of menus on the menu bar. Replaced by `getMenuCount()`.

deleteShortcut

```
public void deleteShortcut (MenuShortcut s) ★
```

Parameters *s* The shortcut to remove.
Description Removes a menu shortcut.

getHelpMenu

```
public Menu getHelpMenu()
```

Returns The menu that was designated the help menu.

getMenu

```
public Menu getMenu (int index)
```

Parameters *index* The position of the `Menu` to fetch.
Returns The `Menu` at the designated position.

getMenuCount

```
public int getMenuCount() ★
```

Returns The number of menus on the menu bar.

getShortcutMenuItem

```
public MenuItem getShortcutMenuItem (MenuShortcut s) ★
```

Parameters *s* A menu shortcut.
Returns The corresponding menu item.
Description Finds the `MenuItem` corresponding to the given `MenuShortcut`, or `null` if no match is found.

remove

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

Parameters *index* The position of the `Menu` to remove.
Description Removes a `Menu` from the `MenuBar`.

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

Parameters *component* The element of the `MenuBar` to remove.
Implements `MenuContainer.remove()`
Description Removes a `Menu` from the `MenuBar`.

removeNotify

```
public void removeNotify()
```

Description Destroys the `MenuBar` peer, and peers for all `Menu` objects that appear on it.

setHelpMenu

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

Parameters	*m*	`Menu` to designate as the help menu.
Description	Designates a `Menu` as the `MenuBar`'s help menu.	

shortcuts

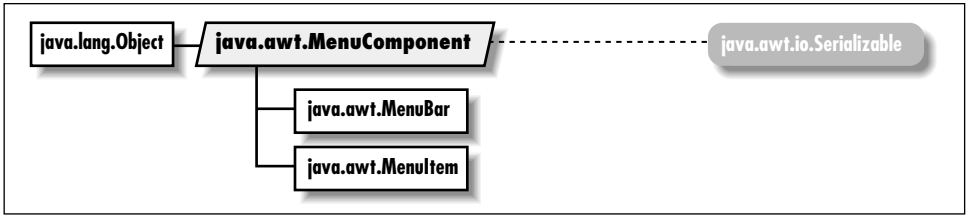
```
public synchronized Enumeration shortcuts() ★
```

Returns	An `Enumeration` of `MenuShortcut` objects.
Description	Returns an `Enumeration` of all `MenuShortcut` objects managed by this `MenuBar`.

See Also

`Frame`, `Menu`, `MenuComponent`, `MenuContainer`

19.42 MenuComponent



Description

The abstract `MenuComponent` class represents the parent of all menu GUI components.

Class Definition

```
public abstract class java.awt.MenuComponent
    extends java.lang.Object
    implements java.io.Serializable {

  // Instance Methods
  public final void dispatchEvent (AWTEvent e); ★
  public Font getFont();
  public String getName(); ★
  public MenuContainer getParent();
  public MenuComponentPeer getPeer(); ☆
  public boolean postEvent (Event e); ☆
  public void removeNotify();
  public void setFont (Font f);
```

```
    public void setName (String name); ★
    public String toString();

    // Protected Instance Methods
    protected String paramString(); ★
    protected void processEvent (AWTEvent e); ★
}
```

Instance Methods

dispatchEvent

```
public final void dispatchEvent (AWTEvent e)
```

Parameters *e* The `AWTEvent` to process.

Description Tells the menu component to deal with the `AWTEvent e`.

getFont

```
public Font getFont()
```

Returns The font for the current `MenuComponent`.

getName

```
public Font getName() ★
```

Returns The name for the current `MenuComponent`.

getParent

```
public MenuContainer getParent()
```

Returns The parent `MenuContainer` for the `MenuComponent`.

getPeer

```
public MenuComponentPeer getPeer() ★
```

Returns A reference to the `MenuComponent`'s peer.

postEvent

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

Parameters *e* `Event` instance to post to component.

Returns Ignored for menus.

Description Tells the `Frame` that contains the `MenuBar` containing the `MenuComponent` to deal with `Event`.

removeNotify

```
public void removeNotify()
```

Description Removes peer of `MenuComponent`'s subclass.

setFont

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

Parameters *f* New font for `MenuComponent`.
Description Changes the font of the label of the `MenuComponent`.

setName

```
public void setName (String name) ★
```

Parameters *name* New name for `MenuComponent`.
Description Changes the name of the `MenuComponent`.

toString

```
public String toString()
```

Returns A string representation of the `MenuComponent` object.

Overrides `Object.toString()`

Protected Instance Methods

paramString

```
protected String paramString() ★
```

Returns String with current settings of `MenuComponent`.
Overrides `Component.paramString()`
Description Helper method for `toString()` to generate string of current settings.

processEvent

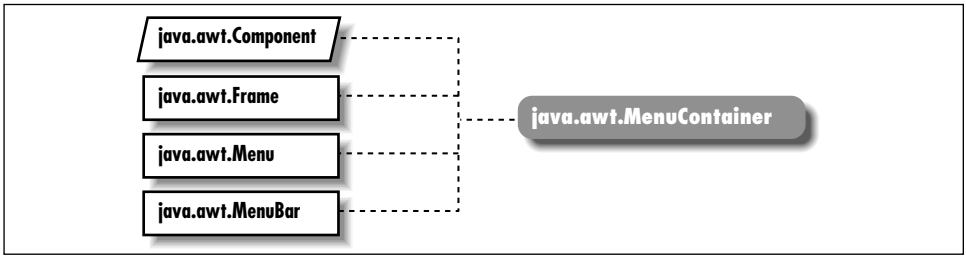
```
protected void processEvent (AWTEvent e) ★
```

Parameters *e* The event to process.
Description Low-level `AWTEvents` are passed to this method for processing.

See Also

`Event`, `Font`, `MenuBar`, `MenuComponentPeer`, `MenuContainer`, `MenuItem`, `Object`, `Serializable`, `String`

19.43 MenuContainer



Description

`MenuContainer` is an interface that defines the responsibilities for objects that can have a menu.

Interface Definition

```
public abstract interface java.awt.MenuContainer
    extends java.lang.Object {

  // Interface Methods
  public abstract Font getFont();
  public abstract boolean postEvent (Event e);  ☆
  public abstract void remove (MenuComponent component);
}
```

Interface Methods

getFont

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

Returns Current font of the object implementing this method.

postEvent

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

Parameters *e* `Event` to post.
Returns Ignores return value.
Description Posts event to the object implementing this method.

remove

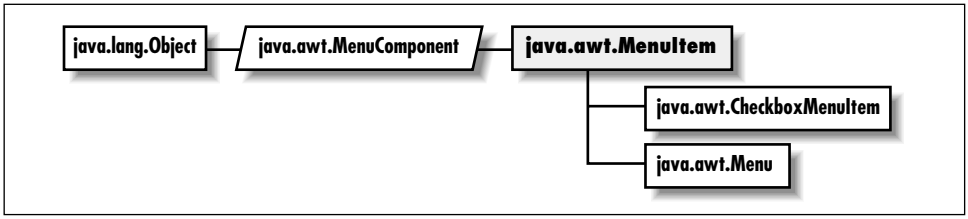
```
public abstract void remove (MenuComponent component)
```

Parameters *component* Menu object to remove
Description Tells the object implementing this method to remove a menu component.

See Also

`Event`, `Font`, `Frame`, `Menu`, `MenuBar`, `MenuComponent`, `Object`

19.44 MenuItem



Description

The `MenuItem` class represents a selectable item on a menu.

Class Definition

```
public class java.awt.MenuItem
    extends java.awt.MenuComponent {

  // Constructors
  public MenuItem(); ★
  public MenuItem (String label);
  public MenuItem (String label, MenuShortcut s); ★

  // Instance Methods
  public void addActionListener (ActionListener l); ★
  public void addNotify();
  public void deleteShortcut(); ★
  public synchronized void disable(); ☆
  public synchronized void enable(); ☆
  public void enable (boolean condition); ☆
  public String getActionCommand(); ★
  public String getLabel();
  public MenuShortcut getShortcut(); ★
  public boolean isEnabled();
  public String paramString();
  public void removeActionListener (ActionListener l); ★
```

```
    public void setActionCommand (String command); ★
    public synchronized void setEnabled (boolean b); ★
    public synchronized void setLabel (String label);
    public void setShortcut (MenuShortcut s); ★

    // Protected Instance Methods
    protected final void disableEvents (long eventsToDisable); ★
    protected final void enableEvents (long eventsToEnable); ★
    protected void processActionEvent (ActionEvent e); ★
    protected void processEvent (AWTEvent e); ★
}
```

Constructors

MenuItem

```
public MenuItem() ★
```

Description Constructs a `MenuItem` object with no label or shortcut.

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

Parameters *label* Text that appears on the `MenuItem`.
Description Constructs a `MenuItem` object.

```
public MenuItem (String label, MenuShortcut s) ★
```

Parameters *label* Text that appears on the `MenuItem`.
s Shortcut for the `MenuItem`.
Description Constructs a `MenuItem` object with the given shortcut.

Instance Methods

addActionListener

```
public void addActionListener(ActionListener l) ★
```

Parameters *l* An object that implements the `ActionListener` interface.
Description Add a listener for the action event.

addNotify

```
public void addNotify()
```

Description Creates the `MenuItem`'s peer.

deleteShortcut

```
public void deleteShortcut() ★
```

Description Removes the shortcut associated with this item.

disable

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

Description Disables the menu component so that it is unresponsive to user interactions. Replaced by `setEnabled(false)`.

enable

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

Description Enables the menu component so that it is responsive to user interactions. Replaced by `setEnabled(true)`.

```
public void enable (boolean condition) ☆
```

Parameters *condition* `true` to enable the menu component; `false` to disable it.

Description Enables or disables the menu component, depending on the `condition` parameter. Replaced by `setEnabled(boolean)`.

getActionCommand

```
public String getActionCommand() ★
```

Returns Current action command string.

Description Returns the string used for the action command.

getLabel

```
public String getLabel()
```

Returns The current text associated with the `MenuItem`.

getShortcut

```
public MenuShortcut getShortcut() ★
```

Returns The current shortcut for this item, or `null` if there is none.

isEnabled

```
public boolean isEnabled()
```

Returns `true` if the menu item is enabled, `false` otherwise.

paramString

```
public String paramString()
```

Returns String with current settings of `MenuItem`.
Description Helper method for `toString()` to generate string of current settings.

removeActionListener

```
public void removeActionListener(ActionListener l) ★
```

Parameters *l* One of this `Button`'s `ActionListeners`.
Description Remove an action event listener.

setActionCommand

```
public void setActionCommand(String command) ★
```

Parameters *command* New action command string.
Description Specify the string used for the action command.

setEnabled

```
public synchronized void setEnabled (boolean b) ★
```

Parameters *b* `true` to enable the item, `false` to disable it.
Description Enables or disables the item. Replaces `enable()`, `enable(boolean)`, and `disable()`.

setLabel

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

Parameters *label* New text to appear on `MenuItem`.
Description Changes the label of the `MenuItem`.

setShortcut

```
public void setShortcut (MenuShortcut s) ★
```

Parameters *s* New shortcut for the `MenuItem`.
Description Changes the shortcut of the `MenuItem`.

Protected Instance Methods

disableEvents

```
protected final void disableEvents (long eventsToDisable)
```
★

Parameters *eventsToDisable*
A value representing certain kinds of events. This can be constructed by ORing the event mask constants defined in `java.awt.AWTEvent`.

Description By default, a menu item receives events corresponding to the event listeners that have registered. If a menu item should not receive events of a certain type, even if there is a listener registered for that type of event, this method can be used to disable that event type.

enableEvents

```
protected final void enableEvents (long eventsToEnable)
```
★

Parameters *eventsToDisable*
A value representing certain kinds of events. This can be constructed by ORing the event mask constants defined in `java.awt.AWTEvent`.

Description By default, a menu item receives events corresponding to the event listeners that have registered. If a menu item should receive other types of events as well, this method can be used to get them.

processActionEvent

```
protected void processActionEvent (ActionEvent e)
```
★

Parameters *e* The action event to process.

Description Action events are passed to this method for processing. Normally, this method is called by `processEvent()`.

processEvent

```
protected void processEvent (AWTEvent e)
```
★

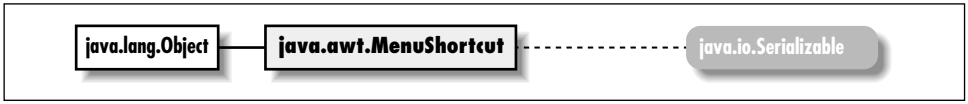
Parameters *e* The event to process.

Description Low-level `AWTEvents` are passed to this method for processing.

See Also

`CheckboxMenuItem`, `Menu`, `MenuComponent`, `MenuShortcut`, `String`

19.45 MenuShortcut ★



Description

A `MenuShortcut` is used to associate a keystroke with a menu item. `MenuShortcuts` are constructed using their corresponding key; they are associated with menu items via `MenuItem.setShortcut(MenuShortcut)`.

Class Definition

```
public class java.awt.MenuShortcut
    extends java.awt.Event {

  // Constructors
  public MenuShortcut (int key);
  public MenuShortcut (int key, boolean useShiftModifier);

  // Instance Methods
  public boolean equals (MenuShortcut s);
  public int getKey();
  public String toString();
  public boolean usesShiftModifier();

  // Protected Instance Methods
  protected String paramString();
}
```

Constructors

`MenuShortcut`

```
public MenuShortcut (int key)
```

Parameters	*key*	A keycode like those returned with key press `Event` objects.
Description	Constructs a `MenuShortcut` object for the given `key`.	

```
public MenuShortcut (int key, boolean useShiftModifier)
```

Parameters	*key*	A keycode like those returned with key press `Event` objects.
	useShiftModifier	`true` if the Shift key must be used, `false` otherwise.
Description	Constructs a `MenuShortcut` object with the given values.	

Instance Methods

equals

```
public boolean equals (MenuShortcut s)
```

Parameters	*s*	The `MenuShortcut` to compare.
Returns	`true` if `s` is equal to this `MenuShortcut`, `false` otherwise.	

getKey

```
public int getKey()
```

Returns	The key for this `MenuShortcut`.

toString

```
public String toString()
```

Returns	A string representation of the `MenuShortcut` object.
Overrides	`Event.toString()`

usesShiftModifier

```
public boolean usesShiftModifier()
```

Returns	`true` if this `MenuShortcut` must be invoked with the Shift key pressed, `false` otherwise.

Protected Instance Methods

paramString

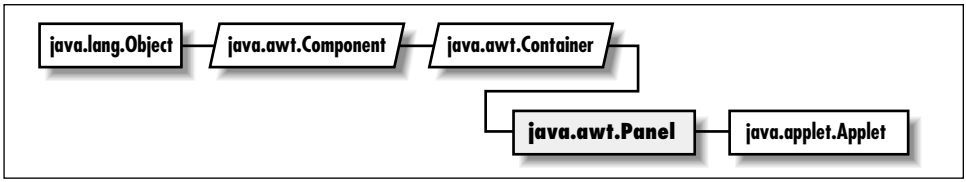
```
protected String paramString()
```

Returns	String with current settings of `MenuShortcut`.
Overrides	`Event.paramString()`
Description	Helper method for `toString()` to generate string of current settings.

See Also

`Event`, `MenuItem`

19.46 Panel



Description

The `Panel` class provides a generic `Container` within an existing display area.

Class Definition

```
public class java.awt.Panel
    extends java.awt.Container {

  // Constructors
  public Panel();
  public Panel(LayoutManager layout); ★

  // Instance Methods
  public void addNotify();
}
```

Constructors

`Panel`

```
public Panel()
```

Description Constructs a `Panel` object.

```
public Panel (LayoutManager layout) ★
```

Description Constructs a `Panel` object with the specified layout manager.

Instance Methods

addNotify

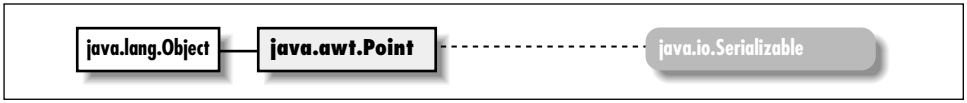
```
public void addNotify()
```

Overrides `Container.addNotify()`
Description Creates `Panel`'s peer and peers of contained components.

See Also

`Applet`, `Container`

19.47 Point



Description

The `Point` class encapsulates a pair of x and y coordinates within a single object.

Class Definition

```
public class java.awt.Point
    extends java.lang.Object
    implements java.io.Serializable {

  // Variables
  public int x;
  public int y;

  // Constructors
  public Point(); ★
  public Point (int width, int height);
  public Point (Point p); ★

  // Instance Methods
  public boolean equals (Object object);
  public Point getLocation(); ★
  public int hashCode();
  public void move (int x, int y);
  public void setLocation (int x, int y); ★
  public void setLocation (Point p); ★
  public String toString();
  public void translate (int deltax, int deltay);
}
```

Variables

x

```
public int x
```

The coordinate that represents the horizontal position.

y

```
public int y
```

The coordinate that represents the vertical position.

Constructors

Point

```
public Point() ★
```

Description Constructs a `Point` object initialized to (0, 0).

```
public Point (int x, int y)
```

Parameters *x* Coordinate that represents the horizontal position.
y Coordinate that represents the vertical position.
Description Constructs a `Point` object with an initial position of (`x`, `y`).

```
public Point (Point p) ★
```

Parameters *p* Initial position.
Description Constructs a `Point` object with the same position as `p`.

Instance Methods

equals

```
public boolean equals (Object object)
```

Parameters *object* The object to compare.
Returns `true` if both points have the same x and y coordinates, `false` otherwise.
Overrides `Object.equals()`
Description Compares two different `Point` instances for equivalence.

getLocation

```
public Point getLocation() ★
```

Returns Position of this point.
Description Gets the current position of this `Point`.

hashCode

```
public int hashCode()
```

Returns	A hashcode to use the `Point` is used as a key in a `Hashtable`.
Overrides	`Object.hashCode()`
Description	Generates a hashcode for the `Point`.

move

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

Parameters	*x*	The new x coordinate.
	y	The new y coordinate.
Description	Changes the `Point`'s location to (`x`, `y`).	

setLocation

```
public void setLocation (int x, int y) ★
```

Parameters	*x*	The new x coordinate.
	y	The new y coordinate.
Description	Changes the `Point`'s location to (`x`, `y`).	

```
public void setLocation (Point p) ★
```

Parameters	*p*	The new location.
Description	Changes the `Point`'s location to `p`.	

toString

```
public String toString()
```

Returns	A string representation of the `Point` object.
Overrides	`Object.toString()`

translate

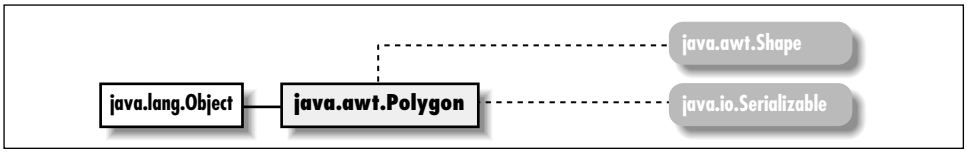
```
public void translate (int deltax, int deltay)
```

Parameters	*deltax*	Amount to move horizontally.
	deltay	Amount to move vertically.
Description	Moves the `Point` to the location (`x+deltax`, `y+deltay`).	

See Also

`Object`, `String`

19.48 Polygon



Description

The `Polygon` class encapsulates a collection of points used to create a series of line segments.

Class Definition

```
public class java.awt.Polygon
    extends java.lang.Object
    implements java.awt.Shape, java.io.Serializable {

  // Variables
  protected Rectangle bounds; ★
  public int npoints;
  public int xpoints[];
  public int ypoints[];

  // Constructors
  public Polygon();
  public Polygon (int xpoints[], int ypoints, int npoints);

  // Instance Methods
  public void addPoint (int x, int y);
  public boolean contains (int x, int y); ★
  public boolean contains (Point p); ★
  public Rectangle getBoundingBox(); ☆
  public Rectangle getBounds(); ★
  public boolean inside (int x,int y); ☆
  public void translate (int deltaX, int deltaY); ★
}
```

Variables

bounds

```
protected Rectangle bounds ★
```

The rectangle that describes the boundaries of the `Polygon`.

npoints

```
public int npoints
```

The number of elements to use in the `xpoints` and `ypoints` arrays.

xpoints

```
public int xpoints[]
```

The array of x coordinates for each point.

ypoints

```
public int ypoints[]
```

The array of y coordinates for each point.

Constructors

Polygon

```
public Polygon()
```

Description Constructs an empty `Polygon` object with no points.

```
public Polygon (int xPoints[], int yPoints[], int
numPoints)
```

Parameters	*xPoints[]*	The initial array of x coordinates for each point.
	yPoints[]	The initial array of y coordinates for each point.
	numPoints	The number of elements in both `xPoints` and `yPoints` arrays to use.
Throws	`ArrayIndexOutOfBoundsException`	If `numPoints > xPoints.length` or `numPoints > yPoints.length`.

Description Constructs a `Polygon` object with the set of points provided.

Instance Methods

addPoint

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

Parameters	*x*	The x coordinate of the point to be added.
	y	The y coordinate of the point to be added.
Description	Adds the point (x, y) to the end of the list of points for the Polygon.	

contains

```
public boolean contains (int x, int y) ★
```

Parameters	*x*	The x coordinate to test.
	y	The y coordinate to test.
Returns	true if the Polygon contains the point; false otherwise.	

```
public boolean contains (Point p) ★
```

Parameters	*p*	The point to be tested.
Returns	true if the Polygon contains the point; false otherwise.	

getBoundingBox

```
public Rectangle getBoundingBox() ☆
```

Returns	Bounding Rectangle of the points within the Polygon.
Description	Returns the smallest Rectangle that contains all the points within the Polygon. Replaced by getBounds().

getBounds

```
public Rectangle getBounds() ★
```

Implements	Shape.getBounds()
Returns	Bounding Rectangle of the points within the Polygon.
Description	Returns the smallest Rectangle that contains all the points within the Polygon.

inside

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

Parameters	*x*	The x coordinate of the point to be checked.
	y	The y coordinate of the point to be checked.
Returns	true if (x, y) within Polygon, false otherwise.	
Description	Checks to see if the (x, y) point is within an area that would be filled if the Polygon was drawn with Graphics.fillPolygon(). Replaced by contains(int, int).	

translate

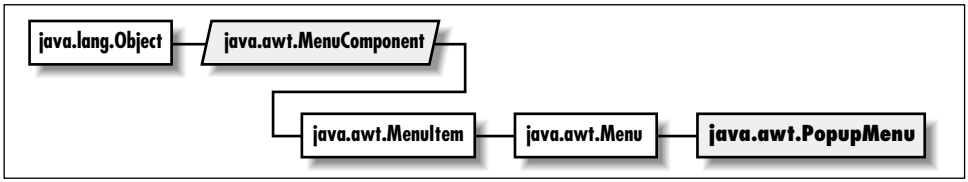
```
public void translate (int deltaX, int deltaY) ★
```

Parameters	*deltaX*	Amount to move horizontally.
	deltaY	Amount to move vertically.
Description	Moves the `Polygon` to the location (`x+deltaX`, `y+deltaY`).	

See Also

`Graphics`, `Object`, `Rectangle`

19.49 PopupMenu ★



Description

A `PopupMenu` is a menu that can be popped up on a `Component`.

Class Definition

```
public class java.awt.PopupMenu
    extends java.awt.Menu {

  // Constructors
  public PopupMenu();
  public PopupMenu (String label);

  // Instance Methods
  public synchronized void addNotify();
  public void show (Component origin, int x, int y);
}
```

Constructors

PopupMenu

```
public PopupMenu()
```

Description Constructs a PopupMenu object.

```
public PopupMenu (String label)
```

Parameters	*label*	Text that appears on `Menu`.
Description	Constructs a PopupMenu object with the given label.	

Instance Methods

`addNotify`

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

Overrides	`Menu.addNotify()`
Description	Creates a PopupMenu peer.

`show`

```
public void show (Component origin, int x, int y)
```

Parameters	*origin*	The `Component` upon which the `PopupMenu` will be displayed.
	x	The `PopupMenu`'s horizontal position on the component.
	y	The `PopupMenu`'s vertical position on the component.
Description	Shows the menu on the given `Component`. The origin specified must be contained in the hierarchy of the `PopupMenu`'s parent component, which is determined by the call to `Component.add(PopupMenu)`.	

19.50 PrintGraphics ★

Description

`PrintGraphics` is an interface for classes that provide a printing graphics context.

Interface Definition

```
public abstract interface java.awt.PrintGraphics {

  // Interface Methods
  public abstract PrintJob getPrintJob();
}
```

Interface Methods

`getPrintJob`

```
public abstract PrintJob getPrintJob()
```

Returns The `PrintJob` from which the `PrintGraphics` object originated.

See Also

`PrintJob`

19.51 PrintJob ★

Description

`PrintJob` encapsulates printing information. When you call `Toolkit.getPrintJob()`, this is the object that is returned. From the `PrintJob`, you can access a `Graphics` object, which can be used for drawing to the printer.

Class Definition

```
public abstract class jav.awt.PrintJob
    extends java.lang.Object {

  // Instance Methods
  public abstract void end();
  public void finalize();
  public abstract Graphics getGraphics();
  public abstract Dimension getPageDimension();
  public abstract int getPageResolution();
  public abstract boolean lastPageFirst();
}
```

Instance Methods

end

```
public abstract void end()
```

Description Ends printing and cleans up.

finalize

```
public void finalize()
```

Overrides `Object.finalize()`
Description Cleans up when this object is garbage collected.

getGraphics

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

Returns A `Graphics` object representing the next page. The object returned will also implement the `PrintGraphics` interface.
Description Returns a `Graphics` object for printing.

getPageDimension

```
public abstract Dimension getPageDimension()
```

Returns The page dimensions in pixels.

getPageResolution

```
public abstract int getPageResolution
```

Returns The page resolution, in pixels per inch.

lastPageFirst

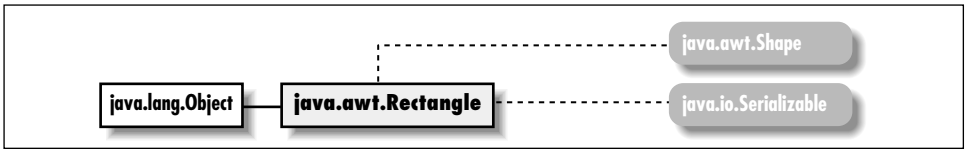
```
public abstract boolean lastPageFirst()
```

Returns `true` if pages are printed in reverse order; `false` otherwise.

See Also

`Dimension`, `Graphics`, `PrintGraphics`, `Toolkit`

19.52 Rectangle



Description

The `Rectangle` class represents a rectangle by combining its origin (a pair of x and y coordinates) with its size (a width and a height).

Class Definition

```
public class java.awt.Rectangle
    extends java.lang.Object
    implements java.awt.Shape, java.io.Serializable {

  // Variables
  pubic int height;
  public int width;
  public int x;
  public int y;

  // Constructors
  public Rectangle();
  public Rectangle (int width, int height);
  public Rectangle (int x, int y, int width, int height);
  public Rectangle (Dimension d);
  public Rectangle (Point p);
  public Rectangle (Point p, Dimension d);
  public Rectangle (Rectangle r); ★

  // Instance Methods
  public void add (int newX, int newY);
  public void add (Point p);
  public void add (Rectangle r);
  public boolean contains (int x, int y); ★
  public boolean contains (Point p); ★
  public boolean equals (Object object);
  public Rectangle getBounds(); ★
  public Point getLocation(); ★
  public Dimension getSize(); ★
  public void grow (int horizontal, int vertical);
  public int hashCode();
  public boolean inside (int x, int y); ☆
  public Rectangle intersection (Rectangle r);
  public boolean intersects (Rectangle r);
```

```
  public boolean isEmpty();
  public void move (int x, int y); ☆
  public void reshape (int x, int y, int width, int height); ☆
  public void resize (int width, int height); ☆
  public void setBounds (Rectangle r); ★
  public void setBounds (int x, int y, int width, int height); ★
  public void setLocation (int x, int y); ★
  public void setLocation (Point p); ★
  public void setSize (int width, int height); ★
  public void setSize (Dimension d); ★
  public String toString();
  public void translate (int x, int y);
  public Rectangle union (Rectangle r);
}
```

Variables

height

```
public int height
```

The height of the `Rectangle`.

width

```
public int width
```

The width of the `Rectangle`.

x

```
public int x
```

The x coordinate of the `Rectangle`'s upper left corner (its origin).

y

```
public int y
```

The y coordinate of the `Rectangle`'s upper left corner (its origin).

Constructors

Rectangle

```
public Rectangle()
```

Description Constructs an empty `Rectangle` object with an origin of (0, 0) and dimensions of 0 x 0.

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

Parameters	*width*	width of `Rectangle`
	height	height of `Rectangle`
Description	Constructs a `Rectangle` object with an origin of (0, 0) and dimensions of `width` x `height`.	

```
public Rectangle (int x, int y, int width, int height)
```

Parameters	*x*	x coordinate of the `Rectangle`'s origin
	y	y coordinate of the `Rectangle`'s origin
	width	width of `Rectangle`
	height	height of `Rectangle`
Description	Constructs a `Rectangle` object with an origin of (`x`, `y`) and dimensions of `width` x `height`.	

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

Parameters	*d*	dimensions of `Rectangle`
Description	Constructs a `Rectangle` object with an origin of (0, 0) and dimensions of `d.width` x `d.height`.	

```
public Rectangle (Point p)
```

Parameters	*p*	origin of `Rectangle`
Description	Constructs an empty `Rectangle` object with an origin of (`p.x`, `p.y`) and dimensions of 0 x 0.	

```
public Rectangle (Point p, Dimension d)
```

Parameters	*p*	origin of `Rectangle`
	d	dimensions of `Rectangle`
Description	Constructs a `Rectangle` object with an origin of (`p.x`, `p.y`) and dimensions of `d.width` x `d.height`.	

```
public Rectangle (Rectangle r) ★
```

Parameters	*r*	original `Rectangle`
Description	Constructs copy of the given `Rectangle`.	

Instance Methods

`add`

```
public void add (int newX, int newY)
```

Parameters	*newX*	The x-coordinate of a point to incorporate within the `Rectangle`.

	newY	The y-coordinate of a point to incorporate within the `Rectangle`.
Description	Extends the `Rectangle` so that the point (`newX`, `newY`) is within it.	

```
public void add (Point p)
```

Parameters	*p*	The new `Point` to add to the `Rectangle`.
Description	Extends the `Rectangle` so that the point `p` is within it.	

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

Parameters	*r*	The `Rectangle` being added to the current `Rectangle`.
Description	Extends the `Rectangle` to include the `Rectangle` `r`.	

`contains`

```
public boolean contains (int x, int y) ★
```

Parameters	*x*	The x coordinate to test.
	y	The y coordinate to test.
Returns	`true` if the `Rectangle` contains the point; `false` otherwise.	

```
public boolean contains (Point p) ★
```

Parameters	*p*	The point to be tested.
Returns	`true` if the `Rectangle` contains the point; `false` otherwise.	

`equals`

```
public boolean equals (Object object)
```

Parameters	*object*	The object to compare.
Returns	`true` if both `Rectangles` have the same origin, width, and height; `false` otherwise.	
Overrides	`Object.equals(Object)`	
Description	Compares two different `Rectangle` instances for equivalence.	

`getBounds`

```
public Rectangle getBounds() ★
```

Implements	`Shape.getBounds()`
Returns	Bounding `Rectangle`.

getLocation

```
public Point getLocation() ★
```

Returns	Position of the rectangle.
Description	Gets the current position of this `Rectangle`.

getSize

```
public Dimension getSize() ★
```

Returns	Dimensions of the rectangle.
Description	Gets width and height of the rectangle.

grow

```
public void grow (int horizontal, int vertical)
```

Parameters	*horizontal*	Amount to extend `Rectangle` in horizontal direction on both the left and right sides.
	vertical	Amount to extend `Rectangle` in vertical direction on both the top and the bottom.
Description	Increases the rectangle's dimensions.	

hashCode

```
public int hashCode()
```

Returns	A hashcode to use when using the `Rectangle` as a key in a `Hashtable`.
Overrides	`Object.hashCode()`
Description	Generates a hashcode for the `Rectangle`.

inside

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

Parameters	*x*	The x coordinate to check.
	y	The y coordinate to check.
Returns	`true` if (`x`, `y`) falls within the `Rectangle`, `false` otherwise.	
Description	Checks to see if the point (`x`, `y`) is within the `Rectangle`. Replaced by `contains(int, int)`.	

intersection

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

Parameters	*r*	`Rectangle` to add to the current `Rectangle`.
Returns	A new `Rectangle` consisting of all points in both the current `Rectangle` and `r`.	
Description	Generates a new `Rectangle` that is the intersection of `r` and the current `Rectangle`.	

intersects

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

Parameters	*r*	`Rectangle` to check.
Returns	`true` if any points in `r` are also in the current `Rectangle`, `false` otherwise.	
Description	Checks to see if `r` crosses the `Rectangle`.	

isEmpty

```
public boolean isEmpty()
```

Returns	`true` if the `Rectangle` is empty, `false` otherwise.
Description	Determines if the rectangle is dimensionless (i.e., width or height are less than or equal to 0).

move

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

Parameters	*x*	The new x coordinate of the `Rectangle`'s upper left corner.
	y	The new y coordinate of the `Rectangle`'s upper left corner.
Description	Changes the `Rectangle`'s origin to (`x`, `y`). Replaced by `setLocation(int, int)`.	

reshape

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

Parameters	*x*	The new x coordinate of the `Rectangle`'s upper left corner.
	y	The new y coordinate of the `Rectangle`'s upper left corner.

	width	The new width.
	height	The new height.
Description	Changes `Rectangle`'s origin and dimensions. Replaced by `setBounds(int, int, int, int)`.	

resize

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

Parameters	*width*	The new width.
	height	The new height.
Description	Changes `Rectangle`'s dimensions. Replaced by `setSize(int, int)`.	

setBounds

```
public void setBounds (Rectangle r) ★
```

Parameters	*r*	A `Rectangle` describing the new bounds.
Description	Changes `Rectangle`'s location and size.	

```
public void setBounds (int x, int y, int width, int
height) [New in 1.1]
```

Parameters	*x*	The new x coordinate of the `Rectangle`'s upper left corner.
	y	The new y coordinate of the `Rectangle`'s upper left corner.
	width	The new width.
	height	The new height.
Description	Changes `Rectangle`'s location and size.	

setLocation

```
public void setLocation (int x, int y) ★
```

Parameters	*x*	New horizontal position.
	y	New vertical position.
Description	Relocates the rectangle.	

```
public void setLocation (Point p) ★
```

Parameters	*p*	New position for component.
Description	Relocates the rectangle.	

setSize

```
public void setSize (int width, int height) ★
```

Parameters	*width*	New width.
	height	New height.
Description	Resizes the rectangle.	

```
public void setSize (Dimension d) ★
```

Parameters	*d*	New dimensions.
Description	Resizes the rectangle.	

toString

```
public String toString()
```

Returns	A string representation of the `Rectangle` object.
Overrides	`Object.toString()`

translate

```
public void translate (int deltax, int deltay)
```

Parameters	*deltax*	Amount to move `Rectangle` horizontally.
	deltay	Amount to move `Rectangle` vertically.
Description	Moves the `Rectangle`'s origin to (`x+deltax`, `y+deltay`).	

union

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

Parameters	*r*	Rectangle to determine union with.
Returns	The smallest `Rectangle` containing both `r` and the current `Rectangle`.	
Description	Generates a new `Rectangle` by combining `r` and the current `Rectangle`.	

See Also

`Dimension`, `Object`, `Point`, `String`

19.53 ScrollPane ⋆

Description

The `ScrollPane` class provides automatic scrolling of a child component.

Class Definition

```
public class java.awt.ScrollPane
    extends java.awt.Container {

  // Constants
  public final static int SCROLLBARS_ALWAYS;
  public final static int SCROLLBARS_AS_NEEDED;
  public final static int SCROLLBARS_NEVER;

  // Constructors
  public ScrollPane();
  public ScrollPane (int scrollbarDisplayPolicy);

  // Public Instance Methods
  public void addNotify();
  public void doLayout();
  public Adjustable getHAdjustable();
  public int getHScrollbarHeight();
  public Point getScrollPosition();
  public int getScrollbarDisplayPolicy();
  public Adjustable getVAdjustable();
  public int getVScrollbarWidth();
  public Dimension getViewportSize();
  public void layout(); ☆
  public String paramString();
  public void printComponents (Graphics g);
  public final void setLayout (LayoutManager mgr);
  public void setScrollPosition (int x, int y);
  public void setScrollPosition (Point p);

  //Protected Instance Methods
  protected final void addImpl (Component comp, Object constraints,
    int index);
}
```

Constants

SCROLLBARS_ALWAYS

```
public final static int SCROLLBARS_ALWAYS
```

Always show the scrollbars.

SCROLLBARS_AS_NEEDED

```
public final static int SCROLLBARS_AS_NEEDED
```

Only show the scrollbars if the contents of the `ScrollPane` are larger than what is visible.

SCROLLBARS_NEVER

```
public final static int SCROLLBARS_NEVER
```

Don't ever show the scrollbars. The `ScrollPane` can still be scrolled programmatically.

Constructors

ScrollPane

```
public ScrollPane()
```

Description Constructs a `ScrollPane` object with `SCROLLBARS_AS_NEEDED`.

```
public ScrollPane (int scrollbarDisplayPolicy)
```

Parameters *scrollbarDisplayPolicy*
One of the `SCROLLBARS_` constants.

Description Constructs a `ScrollPane` object with the specified scrollbar display policy.

Instance Methods

addImpl

```
protected final void addImpl (Component comp, Object
constraints, int index)
```

Parameters *comp* The component to add to the `Scrollpane`.
constraints Layout constraints; ignored.
index The position at which to add the component; should always be less than or equal to 0.

Returns The component that was added.

Overrides `Container.addImpl (Component, Object, int)`

Throws	*IllegalArgumentException* If `pos` is greater than 0.
Description	Adds a child component to the `Scrollpane`. If there already was a child component, it is replaced by the new component.

addNotify

```
public void addNotify()
```

Overrides	`Container.addNotify()`
Description	Creates `ScrollPane`'s peer.

doLayout

```
public void doLayout()
```

Overrides	`Container.doLayout()`
Description	Lays out the `ScrollPane`. Resizes the child component to its preferred size.

getHAdjustable

```
public Adjustable getHAdjustable()
```

Returns	The object implementing the `Adjustable` interface that is used to adjust the `ScrollPane` horizontally. Usually this is a `Scrollbar`.

getHScrollbarHeight

```
public int getHScrollbarHeight()
```

Returns	The height a horizontal scrollbar would occupy, regardless of whether it's shown or not.

getScrollPosition

```
public Point getScrollPosition()
```

Returns	Returns the position within the child component that is displayed at 0, 0 in the `ScrollPane`.

getScrollbarDisplayPolicy

```
public int getScrollbarDisplayPolicy()
```

Returns	The display policy for the scrollbars (one of the `SCROLLBARS_` constants).

getVAdjustable

```
public Adjustable getVAdjustable()
```

Returns The object implementing the `Adjustable` interface that is used to adjust the `ScrollPane` vertically. Usually this is a `Scrollbar`.

getVScrollbarWidth

```
public int getVScrollbarWidth()
```

Returns The width a vertical scrollbar would occupy, regardless of whether it's shown or not.

getViewportSize

```
public Dimension getViewportSize()
```

Returns The size of the `ScrollPane`'s port (the area of the child component that is shown).

layout

```
public void layout() ☆
```

Overrides `Container.layout()`
Description Lays out component. Replaced by `doLayout()`.

paramString

```
public String paramString()
```

Returns String with current settings of `ScrollPane`.
Overrides `Container.paramString()`
Description Helper method for `toString()` to generate string of current settings.

printComponents

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

Parameters *g* Graphics context.
Overrides `Container.printComponents(Graphics)`
Description Prints the `ScrollPane`'s child component.

setLayout

```
public void setLayout (LayoutManager manager)
```

Parameters	*manager*	Ignored.
Overrides	`Container.setLayout(LayoutManager)`	
Description	Does nothing. No layout manager is needed because there is only one child component.	

`setScrollPosition`

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

Parameters	*x*	New horizontal position.
	y	New vertical position.
Throws	*IllegalArgumentException*	If the point given is not valid.
Description	Scroll to the given position in the child component.	

```
public void setScrollPosition (Point p)
```

Parameters	*p*	New position.
Throws	*IllegalArgumentException*	If the point given is not valid.
Description	Scroll to the given position in the child component.	

See Also

`Adjustable`, `Container`, `Point`, `Scrollbar`

19.54 Scrollbar

Description

The `Scrollbar` is a `Component` that provides the means to get and set values within a predetermined range. For example, a scrollbar could be used for a volume control. Scrollbars are most frequently used to help users manipulate areas too large to be displayed on the screen (pre version 1.1) or to set a value within an integer range.

Class Definition

```
public class java.awt.Scrollbar
    extends java.awt.Component
    implements java.awt.Adjustable {

  // Constants
  public final static int HORIZONTAL;
  public final static int VERTICAL;

  // Constructors
  public Scrollbar();
  public Scrollbar (int orientation);
  public Scrollbar (int orientation, int value, int visible, int minimum,
    int maximum);

  // Instance Methods
  public void addAdjustmentListener (AdjustmentListener l); ★
  public void addNotify();
  public int getBlockIncrement(); ★
  public int getLineIncrement(); ☆
  public int getMaximum();
  public int getMinimum();
  public int getOrientation();
  public int getPageIncrement(); ☆
  public int getUnitIncrement(); ★
  public int getValue();
  public int getVisible(); ☆
  public int getVisibleAmount(); ★
  public void removeAdjustmentListener (AdjustmentListener l); ★
  public synchronized void setBlockIncrement (int v); ★
  public void setLineIncrement (int amount); ☆
  public synchronized void setMaximum (int newMaximum); ★
  public synchronized void setMinimum (int newMinimum); ★
  public synchronized void setOrientation (int orientation); ★
  public void setPageIncrement (int amount); ☆
  public synchronized void setUnitIncrement(int v); ★
  public synchronized void setValue (int value);
  public synchronized void setValues (int value, int visible,
    int minimum, int maximum);
  public synchronized void setVisibleAmount (int newAmount); ★

  // Protected Instance Methods
  protected String paramString();
  protected void processAdjustmentEvent (AdjustmentEvent e); ★
  protected void processEvent (AWTEvent e); ★
}
```

Constants

HORIZONTAL

```
public final static int HORIZONTAL
```

Constant used for a `Scrollbar` with a horizontal orientation.

VERTICAL

```
public final static int VERTICAL
```

Constant used for a `Scrollbar` with a vertical orientation.

Constructors

Scrollbar

```
public Scrollbar()
```

Description Constructs a vertical `Scrollbar` object; slider size, minimum value, maximum value, and initial value are all zero.

```
public Scrollbar (int orientation)
```

Parameters *orientation* `Scrollbar` constant designating direction.
Throws *IllegalArgumentException* If `orientation` is invalid.
Description Constructs a `Scrollbar` object, in the designated direction; slider size, minimum value, maximum value, and initial value are all zero.

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

Parameters *orientation* `Scrollbar` constant designating direction.
value Initial value of `Scrollbar`.
visible Initial slider size.
minimum Initial minimum value.
maximum Initial maximum value.
Throws *IllegalArgumentException* If `orientation` is invalid.
Description Constructs a `Scrollbar` object with the given values.

Instance Methods

addAdjustmentListener

```
public void addAdjustmentListener (AdjustmentListener l)
```
★

Parameters	*l*	An object that implements the `AdjustmentListener` interface.
Implements	`Adjustable.addAdjustmentListener()`	
Description	Add a listener for adjustment event.	

addNotify

```
public void addNotify()
```

Overrides	`Component.addNotify()`
Description	Creates `Scrollbar`'s peer.

getBlockIncrement

```
public int getBlockIncrement()
```
★

Implements	`Adjustable.getBlockIncrement()`
Returns	The amount to scroll when a paging area is selected.

getLineIncrement

```
public int getLineIncrement()
```
☆

Returns	The amount to scroll when one of the arrows at the ends of the scrollbar is selected. Replaced by `getUnitIncrement()`.

getMaximum

```
public int getMaximum()
```

Implements	`Adjustable.getMaximum()`
Returns	The maximum value that the `Scrollbar` can take.

getMinimum

```
public int getMinimum()
```

Implements	`Adjustable.getMinimum()`
Returns	The minimum value that the `Scrollbar` can take.

getOrientation

```
public int getOrientation()
```

Implements	`Adjustable.getOrientation()`
Returns	A constant representing the direction of the `Scrollbar`.

getPageIncrement

```
public int getPageIncrement() ☆
```

Returns The amount to scroll when a paging area is selected. Replaced with `getBlockIncrement()`.

getUnitIncrement

```
public int getUnitIncrement() ★
```

Implements `Adjustable.getUnitIncrement()`
Returns The amount to scroll when one of the arrows at the ends of the scrollbar is selected.

getValue

```
public int getValue()
```

Implements `Adjustable.getValue()`
Returns The current setting for the `Scrollbar`.

getVisible

```
public int getVisible() ☆
```

Returns The current visible setting (i.e., size) for the slider. Replaced by `getVisibleAmount()`.

getVisibleAmount

```
public int getVisibleAmount() ★
```

Implements `Adjustable.getVisibleAmount()`
Returns The current visible setting (i.e., size) for the slider.

removeAdjustmentListener

```
public void removeAdjustmentListener (AdjustmentListener
l) ★
```

Parameters *l* One of this `Scrollbar`'s `AdjustmentListeners`.
Implements `Adjustable.removeAdjustmentListener()`
Description Remove an adjustment event listener.

setBlockIncrement

```
public synchronized void setBlockIncrement (int amount) ★
```

Parameters	*amount*	New paging increment amount.
Implements	`Adjustable.setBlockIncrement()`	
Description	Changes the block increment amount for the `Scrollbar`; the default block increment is 10.	

setLineIncrement

```
public void setLineIncrement (int amount) ☆
```

Parameters	*amount*	New line increment amount.
Description	Changes the line increment amount for the `Scrollbar`. The default line increment is 1. Replaced by `setUnitIncrement(int)`.	

setMaximum

```
public synchronized void setMaximum (int newMaximum) ★
```

Parameters	*newMaximum*	New maximum value.
Implements	`Adjustable.setMaximum()`	
Description	Changes the maximum value for the `Scrollbar`.	

setMinimum

```
public synchronized void setMinimum (int newMinimum) ★
```

Parameters	*newMinimum*	New minimum value.
Implements	`Adjustable.setMinimum()`	
Description	Changes the minimum value for the `Scrollbar`.	

setOrientation

```
public synchronized void setOrientation (int orientation)
★
```

Parameters	*orientation*	One of the orientation constants `HORIZONTAL` or `VERTICAL`.
Description	Changes the orientation of the `Scrollbar`.	

setPageIncrement

```
public void setPageIncrement (int amount) ☆
```

Parameters	*amount*	New paging increment amount.
Description	Changes the paging increment amount for the `Scrollbar`; the default page increment is 10. Replaced by `setBlockIncrement(int)`.	

setUnitIncrement

```
public synchronized void setUnitIncrement (int amount) ★
```

Parameters	*amount*	New line increment amount.
Implements	`Adjustable.setUnitIncrement()`	
Description	Changes the unit increment amount for the `Scrollbar`. The default unit increment is 1.	

setValue

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

Parameters	*value*	New `Scrollbar` value.
Implements	`Adjustable.setValue()`	
Description	Changes the current value of the `Scrollbar`.	

setValues

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

Parameters	*value*	New `Scrollbar` value.
	visible	New slider width.
	minimum	New minimum value for `Scrollbar`.
	maximum	New maximum value for `Scrollbar`.
Description	Changes the settings of the `Scrollbar` to the given amounts.	

setVisibleAmount

```
public synchronized void setVisibleAmount (int newAmount)
★
```

Parameters	*newAmount*	New amount visible.
Implements	`Adjustable.setVisibleAmount()`	
Description	Changes the current visible amount of the `Scrollbar`.	

Protected Instance Methods

paramString

```
protected String paramString()
```

Returns	String with current settings of `Scrollbar`.
Overrides	`Component.paramString()`
Description	Helper method for `toString()` to generate string of current settings.

processAdjustmentEvent

```
protected void processAdjustmentEvent (AdjustmentEvent e)
```
★

Parameters	*e*	The adjustment event to process.
Description	Adjustment events are passed to this method for processing. Normally, this method is called by `processEvent()`.	

processEvent

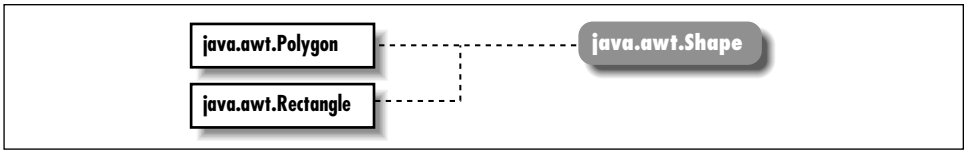
```
protected void processEvent (AWTEvent e)
```
★

Parameters	*e*	The event to process.
Description	Low level `AWTEvents` are passed to this method for processing.	

See Also

`Adjustable`, `Component`, `String`

19.55 Shape ★



Description

`Shape` is an interface describing a two-dimensional geometric shape.

Interface Definition

```
public abstract interface java.awt.Shape {

  // Interface Methods
  public abstract Rectangle getBounds();
}
```

Interface Methods

getBounds

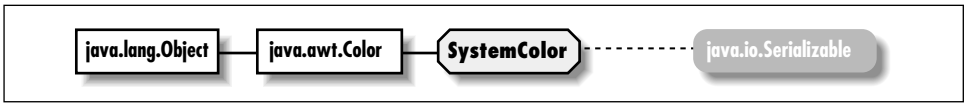
```
public abstract Rectangle getBounds()
```

Returns	A `Rectangle` that completely encloses the shape.

See Also

`Polygon`, `Rectangle`

19.56 SystemColor ★



Description

`SystemColor` provides information on the colors that the windowing system uses to display windows and other graphic components. Most windowing systems allow the user to choose different color schemes; `SystemColor` enables programs to find out what colors are in use in order to paint themselves in a consistent manner.

Class Definition

```
public final class java.awt.SystemColor
    extends java.awt.Color
    implements java.io.Serializable {

  // Constants
  public final static int ACTIVE_CAPTION;
  public final static int ACTIVE_CAPTION_BORDER;
  public final static int ACTIVE_CAPTION_TEXT;
  public final static int CONTROL;
  public final static int CONTROL_DK_SHADOW;
  public final static int CONTROL_HIGHLIGHT;
  public final static int CONTROL_LT_HIGHLIGHT;
  public final static int CONTROL_SHADOW;
  public final static int CONTROL_TEXT;
  public final static int DESKTOP;
  public final static int INACTIVE_CAPTION;
  public final static int INACTIVE_CAPTION_BORDER;
  public final static int INACTIVE_CAPTION_TEXT;
  public final static int INFO;
  public final static int INFO_TEXT;
  public final static int MENU;
  public final static int MENU_TEXT;
  public final static int NUM_COLORS;
  public final static int SCROLLBAR;
  public final static int TEXT;
  public final static int TEXT_HIGHLIGHT;
  public final static int TEXT_HIGHLIGHT_TEXT;
  public final static int TEXT_INACTIVE_TEXT;
  public final static int TEXT_TEXT;
```

```
    public final static int WINDOW;
    public final static int WINDOW_BORDER;
    public final static int WINDOW_TEXT;
    public final static SystemColor activeCaption;
    public final static SystemColor activeCaptionBorder;
    public final static SystemColor activeCaptionText;
    public final static SystemColor control;
    public final static SystemColor controlDkShadow;
    public final static SystemColor controlHighlight;
    public final static SystemColor controlLtHighlight;
    public final static SystemColor controlShadow;
    public final static SystemColor controlText;
    public final static SystemColor desktop;
    public final static SystemColor inactiveCaption;
    public final static SystemColor inactiveCaptionBorder;
    public final static SystemColor inactiveCaptionText;
    public final static SystemColor info;
    public final static SystemColor infoText;
    public final static SystemColor menu;
    public final static SystemColor menuText;
    public final static SystemColor scrollbar;
    public final static SystemColor text;
    public final static SystemColor textHighlight;
    public final static SystemColor textHighlightText;
    public final static SystemColor textInactiveText;
    public final static SystemColor textText;
    public final static SystemColor window;
    public final static SystemColor windowBorder;
    public final static SystemColor windowText;

    // Public Instance Methods
    public int getRGB();
    public String toString();
}
```

Constants

ACTIVE_CAPTION

```
public static final int ACTIVE_CAPTION
```

ACTIVE_CAPTION_BORDER

```
public static final int ACTIVE_CAPTION_BORDER
```

ACTIVE_CAPTION_TEXT

```
public static final int ACTIVE_CAPTION_TEXT
```

CONTROL

```
public static final int CONTROL
```

CONTROL_DK_SHADOW

```
public static final int CONTROL_DK_SHADOW
```

CONTROL_HIGHLIGHT

```
public static final int CONTROL_HIGHLIGHT
```

CONTROL_LT_HIGHLIGHT

```
public static final int CONTROL_LT_HIGHLIGHT
```

CONTROL_SHADOW

```
public static final int CONTROL_SHADOW
```

CONTROL_TEXT

```
public static final int CONTROL_TEXT
```

DESKTOP

```
public static final int DESKTOP
```

INACTIVE_CAPTION

```
public static final int INACTIVE_CAPTION
```

INACTIVE_CAPTION_BORDER

```
public static final int INACTIVE_CAPTION_BORDER
```

INACTIVE_CAPTION_TEXT

```
public static final int INACTIVE_CAPTION_TEXT
```

INFO

```
public static final int INFO
```

INFO_TEXT

```
public static final int INFO_TEXT
```

MENU

```
public static final int MENU
```

MENU_TEXT

```
public static final int MENU_TEXT
```

NUM_COLORS

```
public static final int NUM_COLORS
```

SCROLLBAR

```
public static final int SCROLLBAR
```

TEXT

```
public static final int TEXT
```

TEXT_HIGHLIGHT

```
public static final int TEXT_HIGHLIGHT
```

TEXT_HIGHLIGHT_TEXT

```
public static final int TEXT_HIGHLIGHT_TEXT
```

TEXT_INACTIVE_TEXT

```
public static final int TEXT_INACTIVE_TEXT
```

TEXT_TEXT

```
public static final int TEXT_TEXT
```

WINDOW

```
public static final int WINDOW
```

WINDOW_BORDER

```
public static final int WINDOW_BORDER
```

WINDOW_TEXT

```
public static final int WINDOW_TEXT
```

activeCaption

```
public static final SystemColor activeCaption
```

Background color for captions in window borders.

activeCaptionBorder

```
public static final SystemColor activeCaptionBorder
```

Border color for captions in window borders.

activeCaptionText

```
public static final SystemColor activeCaptionText
```

Text color for captions in window borders.

control

```
public static final SystemColor control
```

Background color for controls.

controlDkShadow

```
public static final SystemColor controlDkShadow
```

Dark shadow color for controls.

controlHighlight

```
public static final SystemColor controlHighlight
```

Highlight color for controls.

controlLtHighlight

```
public static final SystemColor controlLtHighlight
```

Light highlight color for controls.

controlShadow

```
public static final SystemColor controlShadow
```

Shadow color for controls.

controlText

```
public static final SystemColor controlText
```

Text color for controls.

desktop

```
public static final SystemColor desktop
```

Desktop background color.

inactiveCaption

```
public static final SystemColor inactiveCaption
```

Background color for inactive captions in window borders.

inactiveCaptionBorder

```
public static final SystemColor inactiveCaptionBorder
```

Border color for inactive captions in window borders.

inactiveCaptionText

```
public static final SystemColor inactiveCaptionText
```

Text color for inactive captions in window borders.

info

```
public static final SystemColor info
```

Background color for informational text.

infoText

```
public static final SystemColor infoText
```

Text color for informational text.

menu

```
public static final SystemColor menu
```

Background color for menus.

menuText

```
public static final SystemColor menuText
```

Text color for menus.

scrollbar

```
public static final SystemColor scrollbar
```

Background color for scrollbars.

text

```
public static final SystemColor text
```

Background color for text components.

textHighlight

```
public static final SystemColor textHighlight
```

Background color for highlighted text.

textHighlightText

```
public static final SystemColor textHighlightText
```

Text color for highlighted text.

textInactiveText

```
public static final SystemColor textInactiveText
```

Text color for inactive text.

textText

```
public static final SystemColor textText
```

Text color for text components.

window

```
public static final SystemColor window
```

Background color for windows.

windowBorder

```
public static final SystemColor windowBorder
```

Border color for windows.

windowText

```
public static final SystemColor windowText
```

Text color for windows.

Instance Methods

getRGB

```
public int getRGB()
```

Returns	Current color as a composite value
Overrides	`Color.getRGB()`
Description	Gets integer value of current system color.

toString

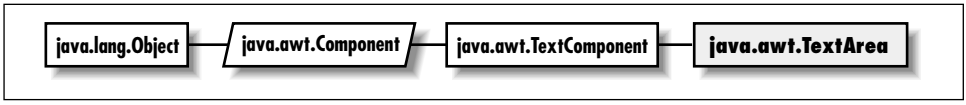
```
public String toString()
```

Returns	A string representation of the `SystemColor` object.
Overrides	`Color.toString()`

See Also

`Color`, `Serializable`, `String`

19.57 TextArea



Description

The `TextArea` class provides a multi-line `Component` for textual user input.

Class Definition

```
public class java.awt.TextArea
    extends java.awt.TextComponent {

  // Constants
  public final static int SCROLLBARS_BOTH; ★
  public final static int SCROLLBARS_HORIZONTAL_ONLY; ★
  public final static int SCROLLBARS_NONE; ★
  public final static int SCROLLBARS_VERTICAL_ONLY; ★

  // Constructors
  public TextArea();
  public TextArea (int rows, int columns);
```

```
    public TextArea (String text);
    public TextArea (String text, int rows, int columns);
    public TextArea (String text, int rows, int columns, int scrollbars); ★

    // Instance Methods
    public void addNotify();
    public synchronized void append (String string); ★
    public void appendText (String string); ☆
    public int getColumns();
    public Dimension getMinimumSize(); ★
    public Dimension getMinimumSize (int rows, int columns); ★
    public Dimension getPreferredSize(); ★
    public Dimension getPreferredSize (int rows, int columns); ★
    public int getRows();
    public int getScrollbarVisibility(); ★
    public synchronized void insert (String string, int position); ★
    public void insertText (String string, int position); ☆
    public Dimension minimumSize(); ☆
    public Dimension minimumSize (int rows, int columns); ☆
    public Dimension preferredSize(); ☆
    public Dimension preferredSize (int rows, int columns); ☆
    public synchronized void replaceRange (String str, int start, int end); ★
    public void replaceText (String string, int startPosition, int endPosition); ☆
    public void setColumns (int columns); ★
    public void setRows (int rows); ★

    // Protected Instance Methods
    protected String paramString();
}
```

Constants

SCROLLBARS_BOTH

```
public final static int SCROLLBARS_BOTH ★
```

Show both the horizontal and vertical scrollbars.

SCROLLBARS_HORIZONTAL_ONLY

```
public final static int SCROLLBARS_HORIZONTAL_ONLY ★
```

Show the horizontal scrollbar.

SCROLLBARS_NONE

```
public final static int SCROLLBARS_NONE ★
```

Show no scrollbars.

SCROLLBARS_VERTICAL_ONLY

```
public final static int SCROLLBARS_VERTICAL_ONLY ★
```

Show the vertical scrollbar.

Constructors

TextArea

```
public TextArea()
```

Description Constructs a `TextArea` object with the default size and no initial content. The default size of a text area varies widely from platform to platform, so it's best to avoid this constructor.

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

Parameters *rows* Requested number of displayed rows.
columns Requested number of displayed columns.
Description Constructs a `TextArea` object of the given size and no initial content.

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

Parameters *text* Initial text for `TextArea`.
Description Constructs a `TextArea` object with the given initial content.

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

Parameters *text* Initial text for `TextArea`.
rows Requested number of displayed rows.
columns Requested number of displayed columns.
Description Constructs a `TextArea` object with the given content and size.

```
public TextArea (String text, int rows, int columns, int
scrollbars) ★
```

Parameters *text* Initial text for `TextArea`.
rows Requested number of displayed rows.
columns Requested number of displayed columns.
scrollbars Requested scrollbar visibility. Use one of the constants defined.
Description Constructs a `TextArea` object with the given content, size, and scrollbar visibility.

Instance Methods

addNotify

```
public void addNotify()
```

Overrides `Component.addNotify()`
Description Creates `TextArea`'s peer.

append

```
public synchronized void append (String string) ★
```

Parameters *string* Content to append to the end of the `TextArea`.
Description Appends the given text string to the text already displayed in the `TextArea`.

appendText

```
public void appendText (String string) ☆
```

Parameters *string* Content to append to end of `TextArea`.
Description Replaced by `append(String)`.

getColumns

```
public int getColumns()
```

Returns The width of the `TextArea` in columns.

getMinimumSize

```
public Dimension getMinimumSize() ★
```

Returns The minimum dimensions of the `TextArea`.

```
public Dimension getMinimumSize (int rows, int columns) ★
```

Parameters *rows* Number of rows within `TextArea` to size.
columns Number of columns within `TextArea` to size.
Returns The minimum dimensions of a `TextArea` of the given size.

getPreferredSize

```
public Dimension getPreferredSize() ★
```

Returns The preferred dimensions of the `TextArea`.

```
public Dimension getPreferredSize (int rows, int columns)
★
```

Parameters	*rows*	Number of rows within `TextArea` to size.
	columns	Number of columns within `TextArea` to size.
Returns	The preferred dimensions of a `TextArea` of the given size.	

getRows

```
public int getRows()
```

Returns The height of the `TextArea` in rows.

getScrollbarVisibility

```
public int getScrollbarVisibility() ★
```

Returns One of the `SCROLLBAR_` constants indicating which scrollbars are visible.

insert

```
public synchronized void insert (String string, int
position) ★
```

Parameters	*string*	Content to place within `TextArea` content.
	position	Location to insert content.
Description	Places additional text within the `TextArea` at the given position.	

insertText

```
public void insertText (String string, int position) ☆
```

Parameters	*string*	Content to place within `TextArea` content.
	position	Location to insert content.
Description	Places additional text within the `TextArea` at the given position. Replaced by `insert(String, int)`.	

minimumSize

```
public Dimension minimumSize() ☆
```

Returns The minimum dimensions of the `TextArea`. Replaced by `getMinimumSize()`.

```
public Dimension minimumSize (int rows, int columns) ☆
```

Parameters	*rows*	Number of rows within `TextArea` to size.
	columns	Number of columns within `TextArea` to size.
Returns	The minimum dimensions of a `TextArea` of the given size. Replaced by `getMinimumSize(int, int)`.	

preferredSize

```
public Dimension preferredSize() ☆
```

Returns The preferred dimensions of the `TextArea`. Replaced by `getPreferredSize()`.

```
public Dimension preferredSize (int rows, int columns) ☆
```

Parameters *rows* Number of rows within `TextArea` to size.
columns Number of columns within `TextArea` to size.
Returns The preferred dimensions of a `TextArea` of the given size. Replaced by `getPreferredSize(int, int)`.

replaceRange

```
public synchronized void replaceRange (String str, int
start, int end) ★
```

Parameters *str* New content to place in `TextArea`.
start Starting position of content to replace.
end Ending position of content to replace.
Description Replaces a portion of the `TextArea`'s content with the given text.

replaceText

```
public void replaceText (String string, int startPosition,
int endPosition) ☆
```

Parameters *string* New content to place in `TextArea`.
startPosition Starting position of content to replace.
endPosition Ending position of content to replace.
Description Replaces a portion of the `TextArea`'s content with the given text. Replaced by `replaceRange(String, int, int)`.

setColumns

```
public void setColumns (int columns) ★
```

Parameters *columns* New number of columns.
Throws *IllegalArgumentException*
If `columns` is less than zero.
Description Changes the number of columns.

setRows

```
public void setRows (int rows) ★
```

Parameters	*rows*	New number of columns.
Throws	*IllegalArgumentException*	If `rows` is less than zero.
Description	Changes the number of rows.	

Protected Instance Methods

paramString

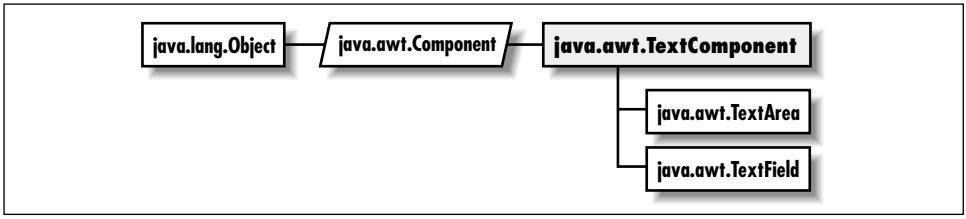
```
protected String paramString()
```

Returns	String with current settings of `TextArea`.
Overrides	`TextComponent.paramString()`
Description	Helper method for `toString()` to generate string of current settings.

See Also

`Dimension`, `TextComponent`, `String`

19.58 TextComponent



Description

The abstract `TextComponent` class provides the base class for the text input components, `TextArea` and `TextField`.

Class Definition

```
public abstract class java.awt.TextComponent
    extends java.awt.Component {

  // Instance Methods
  public synchronized void addTextListener (TextListener l); ★
  public int getCaretPosition(); ★
  public synchronized String getSelectedText();
  public synchronized int getSelectionEnd();
```

```
  public synchronized int getSelectionStart();
  public synchronized String getText();
  public boolean isEditable();
  public void removeNotify();
  public void removeTextListener (TextListener l); ★
  public synchronized void select (int selectionStart, int selectionEnd);
  public synchronized void selectAll();
  public void setCaretPosition (int position); ★
  public synchronized void setEditable (boolean state);
  public synchronized void setSelectionEnd (int selectionEnd); ★
  public synchronized void setSelectionStart (int selectionStart); ★
  public synchronized void setText (String text);

  // Protected Instance Methods
  protected String paramString();
  protected void processEvent (AWTEvent e); ★
  protected void processTextEvent (TextEvent e); ★
}
```

Instance Methods

addTextListener

```
public synchronized void addTextListener (TextListener l)
★
```

Parameters *l* An object that implements the `TextListener` interface.

Description Add a listener for the text events.

getCaretPosition

```
public int getCaretPosition() ★
```

Returns The position, in characters, of the caret (text cursor).

getSelectedText

```
public synchronized String getSelectedText()
```

Returns The currently selected text of the `TextComponent`.

getSelectionEnd

```
public synchronized int getSelectionEnd()
```

Returns The ending cursor position of any selected text.

getSelectionStart

```
public synchronized int getSelectionStart()
```

Returns The initial position of any selected text.

getText

```
public synchronized String getText()
```

Returns Current contents of the `TextComponent`.

isEditable

```
public boolean isEditable()
```

Returns `true` if editable, `false` otherwise.

removeNotify

```
public void removeNotify()
```

Description Destroys the peer of the `TextComponent`.

removeTextListener

```
public void removeTextListener (TextListener l) ★
```

Parameters *l* One of this `TextComponent`'s `TextListeners`.
Description Remove a text event listener.

select

```
public synchronized void select (int selectionStart, int
selectionEnd)
```

Parameters *selectionStart* Beginning position of text to select.
selectionEnd Ending position of text to select.
Description Selects text in the `TextComponent`.

selectAll

```
public synchronized void selectAll()
```

Description Selects all the text in the `TextComponent`.

setCaretPosition

```
public void setCaretPosition (int position) ★
```

Parameters *position* The new character position for the caret.

Throws	*IllegalArgumentException*	If `position` is less than zero.
Description	Allows you to change the location of the caret.	

setEditable

```
public synchronized void setEditable (boolean state)
```

Parameters	*state*	`true` to allow the user to edit the text in the `TextComponent`; `false` to prevent editing.
Description	Allows you to make the `TextComponent` editable or read-only.	

setSelectionEnd

```
public synchronized void setSelectionEnd (int
selectionEnd) ★
```

Parameters	*selectionEnd*	The character position of the end of the selection.
Description	Allows you to change the location of the end of the selected text.	

setSelectionStart

```
public synchronized void setSelectionStart (int
selectionStart) ★
```

Parameters	*selectionStart*	The character position of the start of the selection.
Description	Allows you to change the location of the start of the selected text.	

setText

```
public synchronized void setText (String text)
```

Parameters	*text*	New text for `TextComponent`.
Description	Sets the content of the `TextComponent`.	

Protected Instance Methods

paramString

```
protected String paramString()
```

Returns	String with current settings of `TextComponent`.
Overrides	`Component.paramString()`
Description	Helper method for `toString()` to generate string of current settings.

processEvent

```
protected void processEvent (AWTEvent e) ★
```

Parameters	*e*	The event to process.
Description	Low-level `AWTEvent`s are passed to this method for processing.	

processTextEvent

```
protected void processTextEvent (TextEvent e) ★
```

Parameters	*e*	The event to process.
Description	Text events are passed to this method for processing. Normally, this method is called by `processEvent()`.	

See Also

`Component`, `TextArea`, `TextField`, `String`

19.59 TextField

Description

The `TextField` class provides a single line `Component` for user input.

Class Definition

```
public class java.awt.TextField
    extends java.awt.TextComponent {

  // Constructors
  public TextField();
  public TextField (int columns);
  public TextField (String text);
  public TextField (String text, int columns);

  // Instance Methods
  public void addActionListener (ActionListener l); ★
  public void addNotify();
  public boolean echoCharIsSet();
  public int getColumns();
  public char getEchoChar();
  public Dimension getMinimumSize(); ★
  public Dimension getMinimumSize (int columns); ★
  public Dimension getPreferredSize(); ★
```

```
  public Dimension getPreferredSize (int columns); ★
  public Dimension minimumSize(); ☆
  public Dimension minimumSize (int columns); ☆
  public Dimension preferredSize(); ☆
  public Dimension preferredSize (int columns); ☆
  public void removeActionListener (ActionListener l); ★
  public void setColumns(int columns); ★
  public void setEchoChar(char c); ★
  public void setEchoCharacter (char c); ☆

  // Protected Instance Methods
  protected String paramString();
  protected void processActionEvent (ActionEvent e); ★
  protected void processEvent (AWTEvent e); ★
}
```

Constructors

TextField

```
public TextField()
```

Description Constructs a `TextField` object of the default size.

```
public TextField (int columns)
```

Parameters *columns* Requested number of displayed columns.
Description Constructs a `TextField` object of the given size.

```
public TextField (String text)
```

Parameters *text* Initial text for `TextField`.
Description Constructs a `TextField` object with the given content.

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

Parameters *text* Initial text for `TextField`.
columns Requested number of displayed columns.
Description Constructs a `TextField` object with the given content and size.

Instance Methods

addActionListener

```
public void addActionListener (ActionListener l) ★
```

Parameters *l* An object that implements the `ActionListener` interface.
Description Add a listener for the action event.

addNotify

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

Overrides `Component.addNotify()`
Description Creates `TextField`'s peer.

echoCharIsSet

```
public boolean echoCharIsSet()
```

Returns `true` if the `TextField` has an echo character used as a response to any input character; `false` otherwise. An echo character can be used to create a `TextField` for hidden input, like a password; the same character (e.g., "x") is used to echo all input.

getColumns

```
public int getColumns()
```

Returns The width of the `TextField` in columns.

getEchoChar

```
public char getEchoChar()
```

Returns The current echo character.

getMinimumSize

```
public Dimension getMinimumSize() ★
```

Returns The minimum dimensions of the `TextField`.

```
public Dimension getMinimumSize (int columns) ★
```

Parameters columns Number of columns within `TextField` to size.
Returns The minimum dimensions of a `TextField` of the given size.

getPreferredSize

```
public Dimension getPreferredSize() ★
```

Returns The preferred dimensions of the `TextField`.

```
public Dimension getPreferredSize (int columns) ★
```

Parameters columns Number of columns within `TextField` to size.
Returns The preferred dimensions of a `TextField` of the given size.

minimumSize

```
public Dimension minimumSize() ☆
```

Returns The minimum dimensions of the `TextField`. Replaced by `getMinimumSize()`.

```
public Dimension minimumSize (int columns) ☆
```

Parameters *columns* Number of columns within `TextField` to size.
Returns The minimum dimensions of a `TextField` of the given size. Replaced by `getMinimumSize(int)`.

preferredSize

```
public Dimension preferredSize() ☆
```

Returns The preferred dimensions of the `TextField`. Replaced by `getPreferredSize()`.

```
public Dimension preferredSize (int columns) ☆
```

Parameters *columns* Number of columns within `TextField` to size.
Returns The preferred dimensions of a `TextField` of the given size. Replaced by `getPreferredSize(int)`.

removeActionListener

```
public void removeActionListener (ActionListener l) ★
```

Parameters *l* One of this `TextField`'s `ActionListeners`.
Description Remove an action event listener.

setColumns

```
public void setColumns (int columns) ★
```

Parameters *columns* New number of columns.
Throws *IllegalArgumentException* If `columns` is less than zero.
Description Changes the number of columns.

setEchoChar

```
public void setEchoChar (char c) ★
```

Parameters *c* The character to echo for all input. To echo the characters that the user types (the default), set the echo character to 0 (zero).

Description Changes the character that is used to echo all user input in the `TextField`.

`setEchoCharacter`

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

Parameters *c* The character to echo for all input. To echo the characters that the user types (the default), set the echo character to 0 (zero).

Description Replaced by `setEchoChar(char)` for consistency with `getEchoChar()`.

Protected Instance Methods

`paramString`

```
protected String paramString()
```

Returns String with current settings of `TextField`.

Overrides `TextComponent.paramString()`

Description Helper method for `toString()` to generate string of current settings.

`processActionEvent`

```
protected void processActionEvent (ActionEvent e) ★
```

Parameters *e* The action event to process.

Description Action events are passed to this method for processing. Normally, this method is called by `processEvent()`.

`processEvent`

```
protected void processEvent (AWTEvent e) ★
```

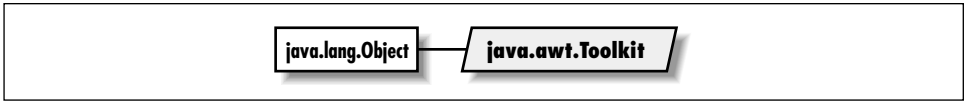
Parameters *e* The event to process.

Description Low-level `AWTEvents` are passed to this method for processing.

See Also

`Dimension`, `TextComponent`, `String`

19.60 Toolkit



Description

The abstract `Toolkit` class provides access to platform-specific details like window size and available fonts. It also deals with creating all the components' peer objects when you call `addNotify()`.

Class Definition

```
public abstract class java.awt.Toolkit
    extends java.lang.Object {

  // Class Methods
  public static synchronized Toolkit getDefaultToolkit();
  protected static Container getNativeContainer (Component c); ★
  public static String getProperty (String key, String defaultValue); ★

  // Instance Methods
  public abstract void beep(); ★
  public abstract int checkImage (Image image, int width, int height,
    ImageObserver observer);
  public abstract Image createImage (ImageProducer producer);
  public Image createImage (byte[] imagedata); ★
  public abstract Image createImage (byte[ ] imagedata, int imageoffset,
    int imagelength); ★
  public abstract ColorModel getColorModel();
  public abstract String[] getFontList();
  public abstract FontMetrics getFontMetrics (Font font);
  public abstract Image getImage (String filename);
  public abstract Image getImage (URL url);
  public int getMenuShortcutKeyMask(); ★
  public abstract PrintJob getPrintJob (Frame frame, String jobtitle,
    Properties props); ★
  public abstract int getScreenResolution();
  public abstract Dimension getScreenSize();
  public abstract Clipboard getSystemClipboard(); ★
  public final EventQueue getSystemEventQueue(); ★
  public abstract boolean prepareImage (Image image, int width, int height,
    ImageObserver observer);
  public abstract void sync();

  // Protected Instance Methods
  protected abstract ButtonPeer createButton (Button b);
```

```
  protected abstract CanvasPeer createCanvas (Canvas c);
  protected abstract CheckboxPeer createCheckbox (Checkbox cb);
  protected abstract CheckboxMenuItemPeer createCheckboxMenuItem
    (CheckboxMenuItem cmi);
  protected abstract ChoicePeer createChoice (Choice c);
  protected LightweightPeer createComponent(Component target); ★
  protected abstract DialogPeer createDialog (Dialog d);
  protected abstract FileDialogPeer createFileDialog (FileDialog fd);
  protected abstract FramePeer createFrame (Frame f);
  protected abstract LabelPeer createLabel (Label l);
  protected abstract ListPeer createList (List l);
  protected abstract MenuPeer createMenu (Menu m);
  protected abstract MenuBarPeer createMenuBar (MenuBar mb);
  protected abstract MenuItemPeer createMenuItem (MenuItem mi);
  protected abstract PanelPeer createPanel (Panel p);
  protected abstract PopupMenuPeer createPopupMenu (PopupMenu target); ★
  protected abstract ScrollPanePeer createScrollPane (ScrollPane target); ★
  protected abstract ScrollbarPeer createScrollbar (Scrollbar sb);
  protected abstract TextAreaPeer createTextArea (TextArea ta);
  protected abstract TextFieldPeer createTextField (TextField tf);
  protected abstract WindowPeer createWindow (Window w);
  protected abstract FontPeer getFontPeer (String name, int style); ★
  protected abstract EventQueue getSystemEventQueueImpl(); ★
  protected void loadSystemColors (int[] systemColors); ★
}
```

Class Methods

getDefaultToolkit

```
public static synchronized Toolkit getDefaultToolkit()
```

Throws	*AWTError*	If the toolkit for the current platform cannot be found.
Returns	The system's default `Toolkit`.	

getNativeContainer

```
protected static Container getNativeContainer (Component
c) ★
```

Returns	The native container for the given component. The component's immediate parent may be a lightweight component.

getProperty

```
public static String getProperty (String key, String
defaultValue) ★
```

Parameters	*key*	The name of a property.
	defaultValue	A default value to return if the property is not found.
Returns	The value of the property described by `key`, or `defaultValue` if it is not found.	

Instance Methods

beep

```
public abstract void beep() ★
```

Description Produces an audible beep.

checkImage

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

Parameters	*image*	Image to check.
	width	Width of the scaled image; -1 if image will be rendered unscaled.
	height	Height of the scaled image; -1 if image will be rendered unscaled.
	observer	The `Component` that `image` will be rendered on.
Returns	The `ImageObserver` flags ORed together for the data that is now available.	
Description	Checks on the status of the construction of a screen representation of `image` on `observer`.	

createImage

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

Parameters	*producer*	An `ImageProducer` that generates data for the desired image.
Returns	Newly created `Image`.	
Description	Creates a new `Image` from an `ImageProducer`.	

```
public abstract Image createImage (byte[] imagedata) ★
```

Parameters	*imagedata*	Raw data representing an image.

Returns Newly created `Image`.
Description Creates a new `Image` from the `imagedata` provided.

```
public abstract Image createImage (byte[] imagedata,
int imageoffset, int imagelength) ★
```

Parameters *imagedata* Raw data representing one or more images.
imageoffset An offset into the data given.
imagelength The length of data to use.
Returns Newly created `Image`.
Description Creates a new `Image` from the `imagedata` provided, starting at `imageoffset` bytes and reading `imagelength` bytes.

`getColorModel`

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

Returns The current `ColorModel` used by the system.

`getFontList`

```
public abstract String[] getFontList()
```

Returns A `String` array of the set of Java fonts available with this `Toolkit`.

`getFontMetrics`

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

Parameters *font* A `Font` whose metrics are desired
Returns The current `FontMetrics` for the `font` on the user's system.

`getImage`

```
public abstract Image getImage (String filename)
```

Parameters *filename* Location of `Image` on local filesystem
Returns The `Image` that needs to be fetched.
Description Fetches an image from the local file system.

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

Parameters *url* Location of `Image`.
Returns The `Image` that needs to be fetched.
Description Fetches an image from a `URL`.

getMenuShortcutKeyMask

```
public int getMenuShortcutKeyMask() ★
```

Returns The modifier key mask used for menu shortcuts. This will be one of the mask constants defined in `java.awt.Event`.

getPrintJob

```
public abstract PrintJob getPrintJob (Frame frame,
String jobtitle, Properties props) ★
```

Parameters
- *frame* — The `frame` to be used as the parent of a platform-specific printing dialog.
- *jobtitle* — The name of the job.
- *props* — Properties for this print job.

Returns A `PrintJob` object. If the user canceled the printing operation, `null` is returned.

getScreenResolution

```
public abstract int getScreenResolution()
```

Returns The current resolution of the user's screen, in dots-per-inch.

getScreenSize

```
public abstract Dimension getScreenSize()
```

Returns The size of the screen available to the `Toolkit`, in pixels, as a `Dimension` object.

getSystemClipboard

```
public abstract Clipboard getSystemClipboard() ★
```

Returns A `Clipboard` object that can be used for cut, copy, and paste operations.

getSystemEventQueue

```
public final EventQueue getSystemEventQueue() ★
```

Returns A reference to the system's event queue, allowing the program to post new events or inspect the queue.

prepareImage

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

Parameters	*image*	Image to check.
	width	Width of the scaled image; -1 if image will be rendered unscaled.
	height	Height of the scaled image; -1 if image will be rendered unscaled.
	observer	The `Component` that `image` will be rendered on.
Returns	`true` if image fully loaded, `false` otherwise.	
Description	Forces the system to start loading the `image`.	

sync

```
public abstract void sync()
```

Description Flushes the display of the underlying graphics context.

Protected Instance Methods

createButton

```
protected abstract ButtonPeer createButton (Button b)
```

Parameters	*b*	`Component` whose peer needs to be created.
Returns	Newly created peer.	
Description	Creates a peer for the `Button`.	

createCanvas

```
protected abstract CanvasPeer createCanvas (Canvas c)
```

Parameters	*c*	`Component` whose peer needs to be created.
Returns	Newly created peer.	
Description	Creates a peer for the `Canvas`.	

createCheckbox

```
protected abstract CheckboxPeer createCheckbox (Checkbox
cb)
```

Parameters	*cb*	`Component` whose peer needs to be created.
Returns	Newly created peer.	
Description	Creates a peer for the `Checkbox`.	

createCheckboxMenuItem

```
protected abstract CheckboxMenuItemPeer
createCheckboxMenuItem (CheckboxMenuItem cmi)
```

Parameters	*cmi*	`Component` whose peer needs to be created.
Returns	Newly created peer.	
Description	Creates a peer for the `CheckboxMenuItem`.	

createChoice

```
protected abstract ChoicePeer createChoice (Choice c)
```

Parameters	*c*	`Component` whose peer needs to be created.
Returns	Newly created peer.	
Description	Creates a peer for the `Choice`.	

createComponent

```
protected LightweightPeer createComponent (Component
target) ★
```

Parameters	*target*	`Component` whose peer needs to be created.
Returns	Newly created peer.	
Description	Creates a peer for the `Component`.	

createDialog

```
protected abstract DialogPeer createDialog (Dialog d)
```

Parameters	*d*	`Component` whose peer needs to be created.
Returns	Newly created peer.	
Description	Creates a peer for the `Dialog`.	

createFileDialog

```
protected abstract FileDialogPeer createFileDialog
(FileDialog fd)
```

Parameters	*fd*	`Component` whose peer needs to be created.
Returns	Newly created peer.	
Description	Creates a peer for the `FileDialog`.	

createFrame

```
protected abstract FramePeer createFrame (Frame f)
```

Parameters	*f*	`Component` whose peer needs to be created.

Returns Newly created peer.
Description Creates a peer for the `Frame`.

createLabel

```
protected abstract LabelPeer createLabel (Label l)
```

Parameters *l* `Component` whose peer needs to be created.
Returns Newly created peer.
Description Creates a peer for the `Label`.

createList

```
protected abstract ListPeer createList (List l)
```

Parameters *l* `Component` whose peer needs to be created.
Returns Newly created peer.
Description Creates a peer for the `List`.

createMenu

```
protected abstract MenuPeer createMenu (Menu m)
```

Parameters *m* `Menu` whose peer needs to be created.
Returns Newly created peer.
Description Creates a peer for the given `Menu`.

createMenuBar

```
protected abstract MenuBarPeer createMenuBar (MenuBar mb)
```

Parameters *mb* `MenuBar` whose peer needs to be created.
Returns Newly created peer.
Description Creates a peer for the `MenuBar`.

createMenuItem

```
protected abstract MenuItemPeer createMenuItem (MenuItem
mi)
```

Parameters *mi* `MenuItem` whose peer needs to be created.
Returns Newly created peer.
Description Creates a peer for the `MenuItem`.

createPanel

```
protected abstract PanelPeer createPanel (Panel p)
```

Parameters	*p*	`Component` whose peer needs to be created.
Returns	Newly created peer.	
Description	Creates a peer for the `Panel`.	

createPopupMenu

```
protected abstract PopupMenuPeer createPopupMenu
(PopupMenu target) ★
```

Parameters	*target*	`Component` whose peer needs to be created.
Returns	Newly created peer.	
Description	Creates a peer for the `PopupMenu`.	

createScrollPane

```
protected abstract ScrollPanePeer createScrollPane
(ScrollPane target) ★
```

Parameters	*target*	`Component` whose peer needs to be created.
Returns	Newly created peer.	
Description	Creates a peer for the `ScrollPane`.	

createScrollbar

```
protected abstract ScrollbarPeer createScrollbar
(Scrollbar sb)
```

Parameters	*sb*	`Component` whose peer needs to be created.
Returns	Newly created peer.	
Description	Creates a peer for the `Scrollbar`.	

createTextArea

```
protected abstract TextAreaPeer createTextArea (TextArea
ta)
```

Parameters	*ta*	`Component` whose peer needs to be created.
Returns	Newly created peer.	
Description	Creates a peer for the `TextArea`.	

createTextField

```
protected abstract TextFieldPeer createTextField
(TextField tf)
```

Parameters	*tf*	`Component` whose peer needs to be created.
Returns	Newly created peer.	
Description	Creates a peer for the `TextField`.	

`createWindow`

```
protected abstract WindowPeer createWindow (Window w)
```

Parameters	*w*	`Component` whose peer needs to be created.
Returns	Newly created peer.	
Description	Creates a peer for the `Window`.	

`getFontPeer`

```
protected abstract FontPeer getFontPeer (String name, int
style) ★
```

Parameters	*name*	Name of the font to be created.
	style	Style of the font to be created.
Returns	Newly created peer.	
Description	Creates a `FontPeer`.	

`getSystemEventQueueImpl`

```
protected abstract getSystemEventQueueImpl()★
```

Returns	A toolkit-specific `EventQueue` object.

`loadSystemColors`

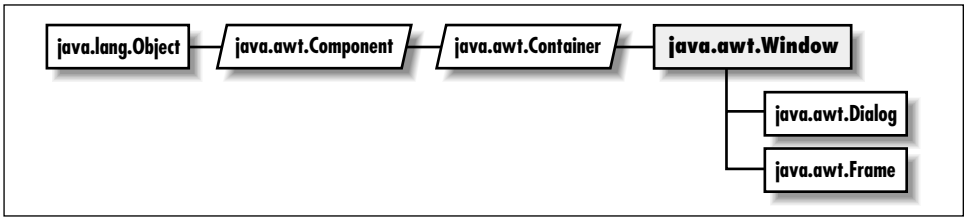
```
protected abstract void loadSystemColors
(int[] systemColors) ★
```

Description	Fills the given integer array with the current system colors.

See Also

`Button`, `ButtonPeer`, `Canvas`, `CanvasPeer`, `Checkbox`, `CheckboxMenuItem`, `CheckboxMenuItemPeer`, `CheckboxPeer`, `Choice`, `ChoicePeer`, `Clipboard`, `ColorModel`, `Component`, `Container`, `Dialog`, `DialogPeer`, `Dimension`, `FileDialog`, `FileDialogPeer`, `Font`, `FontMetrics`, `FontPeer`, `Frame`, `FramePeer`, `Image`, `ImageObserver`, `ImageProducer`, `Label`, `LabelPeer`, `LightweightPeer`, `List`, `ListPeer`, `Menu`, `MenuBar`, `MenuBarPeer`, `MenuItem`, `MenuItemPeer`, `MenuPeer`, `Panel`, `PanelPeer`, `PrintJob`, `Scrollbar`, `ScrollbarPeer`, `ScrollPane`, `ScrollPanePeer`, `String`, `TextArea`, `TextAreaPeer`, `TextField`, `TextFieldPeer`, `Window`, `WindowPeer`

19.61 Window



Description

The `Window` class serves as a top-level display area that exists outside the browser or applet area you may be working in. A window must have a parent `Frame`.

Class Definition

```
public class java.awt.Window
    extends java.awt.Container {

  // Constructors
  public Window (Frame parent);

  // Instance Methods
  public void addNotify();
  public synchronized void addWindowListener (WindowListener l); ★
  public void dispose();
  public Component getFocusOwner(); ★
  public Locale getLocale(); ★
  public Toolkit getToolkit();
  public final String getWarningString();
  public boolean isShowing(); ★
  public void pack();
  public boolean postEvent (Event e); ☆
  public synchronized void remove WindowListener (WindowListener l); ★
  public void show();
  public void toBack();
  public void toFront();
//Protected Instance Methods
  protected void processEvent (AWTEvent e); ★
  protected void processWindowEvent (WindowEvent e); ★
}
```

Constructors

Window

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

Parameters *parent* `Frame` that is to act as the parent of `Window`.
Description Constructs a `Window` object.

Instance Methods

addNotify

```
public void addNotify()
```

Overrides `Container.addNotify()`
Description Creates `Window`'s peer and peers of contained components.

removeWindowListener

```
public synchronized void
removeWindowListener(WindowListener l) ★
```

Parameters *l* One of this `Frame`'s `WindowListeners`.
Description Remove an event listener.

addWindowListener

```
public synchronized void addWindowListener (WindowListener
l) ★
```

Parameters *l* An object that implements the `WindowListener` interface.

Description Add a listener for windowing events.

dispose

```
public void dispose()
```

Returns Releases the resources of the `Window`.

getFocusOwner

```
public Component getFocusOwner() ★
```

Returns The child component that currently has the input focus.

getLocale

```
public Locale getLocale() ★
```

Returns The locale for this `Window`.
Overrides `Window.getLocale()`

getToolkit

```
public Toolkit getToolkit()
```

Returns `Toolkit` of `Window`.
Overrides `Component.getToolkit()`

getWarningString

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

Returns String that will be displayed on the bottom of insecure `Window` instances.

isShowing

```
public boolean isShowing()
```

Returns `true` if the `Window` is showing on the screen, `false` otherwise.

pack

```
public void pack()
```

Description Resizes `Window` to `getPreferredSize()` of contained components.

postEvent

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

Parameters *e* `Event` instance to post to window.
Returns If `Event` is handled, `true` is returned. Otherwise, `false` is returned.
Description Tells the `Window` to deal with `Event`.

removeWindowListener

```
public synchronized void removeWindowListener
(WindowListener l) ★
```

Parameters *l* One of this `Frame`'s `WindowListeners`.

Description Remove an event listener.

show

```
public void show()
```

Description Show the Window and validate its components.
Overrides Component.show()

toBack

```
public void toBack()
```

Description Puts the Window in the background of the display.

toFront

```
public void toFront()
```

Description Brings the Window to the foreground of the display.

Protected Instance Methods

processEvent

```
protected void processEvent (AWTEvent e) ★
```

Parameters *e* The event to process.
Description Low level AWTEvents are passed to this method for processing.

processWindowEvent

```
protected void processWindowEvent (WindowEvent e) ★
```

Parameters *e* The event to process.
Description Window events are passed to this method for processing. Normally, this method is called by processEvent().

See Also

Component, Container, Dialog, Frame, String, Toolkit