| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/JComponent.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/javax/swing/JComboBox.KeySelectionManager.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/JComponent.AccessibleJComponent.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/JComponent.html)    [**NO FRAMES**](http://docs.google.com/JComponent.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | [FIELD](#3dy6vkm) | [CONSTR](#2s8eyo1) | [METHOD](#17dp8vu) | DETAIL: [FIELD](#35nkun2) | [CONSTR](#1ci93xb) | [METHOD](#2bn6wsx) |

## **javax.swing**

Class JComponent

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 [java.awt.Component](http://docs.google.com/java/awt/Component.html)  
 [java.awt.Container](http://docs.google.com/java/awt/Container.html)  
 **javax.swing.JComponent**

**All Implemented Interfaces:** [ImageObserver](http://docs.google.com/java/awt/image/ImageObserver.html), [MenuContainer](http://docs.google.com/java/awt/MenuContainer.html), [Serializable](http://docs.google.com/java/io/Serializable.html) **Direct Known Subclasses:** [AbstractButton](http://docs.google.com/javax/swing/AbstractButton.html), [BasicInternalFrameTitlePane](http://docs.google.com/javax/swing/plaf/basic/BasicInternalFrameTitlePane.html), [Box](http://docs.google.com/javax/swing/Box.html), [Box.Filler](http://docs.google.com/javax/swing/Box.Filler.html), [JColorChooser](http://docs.google.com/javax/swing/JColorChooser.html), [JComboBox](http://docs.google.com/javax/swing/JComboBox.html), [JFileChooser](http://docs.google.com/javax/swing/JFileChooser.html), [JInternalFrame](http://docs.google.com/javax/swing/JInternalFrame.html), [JInternalFrame.JDesktopIcon](http://docs.google.com/javax/swing/JInternalFrame.JDesktopIcon.html), [JLabel](http://docs.google.com/javax/swing/JLabel.html), [JLayeredPane](http://docs.google.com/javax/swing/JLayeredPane.html), [JList](http://docs.google.com/javax/swing/JList.html), [JMenuBar](http://docs.google.com/javax/swing/JMenuBar.html), [JOptionPane](http://docs.google.com/javax/swing/JOptionPane.html), [JPanel](http://docs.google.com/javax/swing/JPanel.html), [JPopupMenu](http://docs.google.com/javax/swing/JPopupMenu.html), [JProgressBar](http://docs.google.com/javax/swing/JProgressBar.html), [JRootPane](http://docs.google.com/javax/swing/JRootPane.html), [JScrollBar](http://docs.google.com/javax/swing/JScrollBar.html), [JScrollPane](http://docs.google.com/javax/swing/JScrollPane.html), [JSeparator](http://docs.google.com/javax/swing/JSeparator.html), [JSlider](http://docs.google.com/javax/swing/JSlider.html), [JSpinner](http://docs.google.com/javax/swing/JSpinner.html), [JSplitPane](http://docs.google.com/javax/swing/JSplitPane.html), [JTabbedPane](http://docs.google.com/javax/swing/JTabbedPane.html), [JTable](http://docs.google.com/javax/swing/JTable.html), [JTableHeader](http://docs.google.com/javax/swing/table/JTableHeader.html), [JTextComponent](http://docs.google.com/javax/swing/text/JTextComponent.html), [JToolBar](http://docs.google.com/javax/swing/JToolBar.html), [JToolTip](http://docs.google.com/javax/swing/JToolTip.html), [JTree](http://docs.google.com/javax/swing/JTree.html), [JViewport](http://docs.google.com/javax/swing/JViewport.html)

public abstract class **JComponent**extends [Container](http://docs.google.com/java/awt/Container.html)implements [Serializable](http://docs.google.com/java/io/Serializable.html)

The base class for all Swing components except top-level containers. To use a component that inherits from JComponent, you must place the component in a containment hierarchy whose root is a top-level Swing container. Top-level Swing containers -- such as JFrame, JDialog, and JApplet -- are specialized components that provide a place for other Swing components to paint themselves. For an explanation of containment hierarchies, see [Swing Components and the Containment Hierarchy](http://java.sun.com/docs/books/tutorial/uiswing/overview/hierarchy.html), a section in *The Java Tutorial*.

The JComponent class provides:

* The base class for both standard and custom components that use the Swing architecture.
* A "pluggable look and feel" (L&F) that can be specified by the programmer or (optionally) selected by the user at runtime. The look and feel for each component is provided by a *UI delegate* -- an object that descends from [ComponentUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html). See [How to Set the Look and Feel](http://java.sun.com/docs/books/tutorial/uiswing/misc/plaf.html) in *The Java Tutorial* for more information.
* Comprehensive keystroke handling. See the document [Keyboard Bindings in Swing](http://java.sun.com/products/jfc/tsc/special_report/kestrel/keybindings.html), an article in *The Swing Connection*, for more information.
* Support for tool tips -- short descriptions that pop up when the cursor lingers over a component. See [How to Use Tool Tips](http://java.sun.com/docs/books/tutorial/uiswing/components/tooltip.html) in *The Java Tutorial* for more information.
* Support for accessibility. JComponent contains all of the methods in the Accessible interface, but it doesn't actually implement the interface. That is the responsibility of the individual classes that extend JComponent.
* Support for component-specific properties. With the [putClientProperty(java.lang.Object, java.lang.Object)](http://docs.google.com/javax/swing/JComponent.html#putClientProperty(java.lang.Object,%20java.lang.Object)) and [getClientProperty(java.lang.Object)](http://docs.google.com/javax/swing/JComponent.html#getClientProperty(java.lang.Object)) methods, you can associate name-object pairs with any object that descends from JComponent.
* An infrastructure for painting that includes double buffering and support for borders. For more information see [Painting](http://java.sun.com/docs/books/tutorial/uiswing/overview/draw.html) and [How to Use Borders](http://java.sun.com/docs/books/tutorial/uiswing/misc/border.html), both of which are sections in *The Java Tutorial*.

For more information on these subjects, see the [Swing package description](http://docs.google.com/package-summary.html#package_description) and *The Java Tutorial* section [The JComponent Class](http://java.sun.com/docs/books/tutorial/uiswing/components/jcomponent.html).

JComponent and its subclasses document default values for certain properties. For example, JTable documents the default row height as 16. Each JComponent subclass that has a ComponentUI will create the ComponentUI as part of its constructor. In order to provide a particular look and feel each ComponentUI may set properties back on the JComponent that created it. For example, a custom look and feel may require JTables to have a row height of 24. The documented defaults are the value of a property BEFORE the ComponentUI has been installed. If you need a specific value for a particular property you should explicitly set it.

In release 1.4, the focus subsystem was rearchitected. For more information, see  [How to Use the Focus Subsystem](http://java.sun.com/docs/books/tutorial/uiswing/misc/focus.html), a section in *The Java Tutorial*.

**Warning:** Swing is not thread safe. For more information see [Swing's Threading Policy](http://docs.google.com/package-summary.html#threading).

**Warning:** Serialized objects of this class will not be compatible with future Swing releases. The current serialization support is appropriate for short term storage or RMI between applications running the same version of Swing. As of 1.4, support for long term storage of all JavaBeansTM has been added to the java.beans package. Please see [XMLEncoder](http://docs.google.com/java/beans/XMLEncoder.html).

**See Also:**[KeyStroke](http://docs.google.com/javax/swing/KeyStroke.html), [Action](http://docs.google.com/javax/swing/Action.html), [setBorder(javax.swing.border.Border)](http://docs.google.com/javax/swing/JComponent.html#setBorder(javax.swing.border.Border)), [registerKeyboardAction(java.awt.event.ActionListener, java.lang.String, javax.swing.KeyStroke, int)](http://docs.google.com/javax/swing/JComponent.html#registerKeyboardAction(java.awt.event.ActionListener,%20java.lang.String,%20javax.swing.KeyStroke,%20int)), [JOptionPane](http://docs.google.com/javax/swing/JOptionPane.html), [setDebugGraphicsOptions(int)](http://docs.google.com/javax/swing/JComponent.html#setDebugGraphicsOptions(int)), [setToolTipText(java.lang.String)](http://docs.google.com/javax/swing/JComponent.html#setToolTipText(java.lang.String)), [setAutoscrolls(boolean)](http://docs.google.com/javax/swing/JComponent.html#setAutoscrolls(boolean))

| **Nested Class Summary** | |
| --- | --- |
| class | [**JComponent.AccessibleJComponent**](http://docs.google.com/javax/swing/JComponent.AccessibleJComponent.html)            Inner class of JComponent used to provide default support for accessibility. |

| **Nested classes/interfaces inherited from class java.awt.**[**Container**](http://docs.google.com/java/awt/Container.html) |
| --- |
| [Container.AccessibleAWTContainer](http://docs.google.com/java/awt/Container.AccessibleAWTContainer.html) |

| **Nested classes/interfaces inherited from class java.awt.**[**Component**](http://docs.google.com/java/awt/Component.html) |
| --- |
| [Component.AccessibleAWTComponent](http://docs.google.com/java/awt/Component.AccessibleAWTComponent.html), [Component.BaselineResizeBehavior](http://docs.google.com/java/awt/Component.BaselineResizeBehavior.html), [Component.BltBufferStrategy](http://docs.google.com/java/awt/Component.BltBufferStrategy.html), [Component.FlipBufferStrategy](http://docs.google.com/java/awt/Component.FlipBufferStrategy.html) |

| **Field Summary** | |
| --- | --- |
| protected  [AccessibleContext](http://docs.google.com/javax/accessibility/AccessibleContext.html) | [**accessibleContext**](http://docs.google.com/javax/swing/JComponent.html#accessibleContext)            The AccessibleContext associated with this JComponent. |
| protected  [EventListenerList](http://docs.google.com/javax/swing/event/EventListenerList.html) | [**listenerList**](http://docs.google.com/javax/swing/JComponent.html#listenerList)            A list of event listeners for this component. |
| static [String](http://docs.google.com/java/lang/String.html) | [**TOOL\_TIP\_TEXT\_KEY**](http://docs.google.com/javax/swing/JComponent.html#TOOL_TIP_TEXT_KEY)            The comment to display when the cursor is over the component, also known as a "value tip", "flyover help", or "flyover label". |
| protected  [ComponentUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html) | [**ui**](http://docs.google.com/javax/swing/JComponent.html#ui)            The look and feel delegate for this component. |
| static int | [**UNDEFINED\_CONDITION**](http://docs.google.com/javax/swing/JComponent.html#UNDEFINED_CONDITION)            Constant used by some of the APIs to mean that no condition is defined. |
| static int | [**WHEN\_ANCESTOR\_OF\_FOCUSED\_COMPONENT**](http://docs.google.com/javax/swing/JComponent.html#WHEN_ANCESTOR_OF_FOCUSED_COMPONENT)            Constant used for registerKeyboardAction that means that the command should be invoked when the receiving component is an ancestor of the focused component or is itself the focused component. |
| static int | [**WHEN\_FOCUSED**](http://docs.google.com/javax/swing/JComponent.html#WHEN_FOCUSED)            Constant used for registerKeyboardAction that means that the command should be invoked when the component has the focus. |
| static int | [**WHEN\_IN\_FOCUSED\_WINDOW**](http://docs.google.com/javax/swing/JComponent.html#WHEN_IN_FOCUSED_WINDOW)            Constant used for registerKeyboardAction that means that the command should be invoked when the receiving component is in the window that has the focus or is itself the focused component. |

| **Fields inherited from class java.awt.**[**Component**](http://docs.google.com/java/awt/Component.html) |
| --- |
| [BOTTOM\_ALIGNMENT](http://docs.google.com/java/awt/Component.html#BOTTOM_ALIGNMENT), [CENTER\_ALIGNMENT](http://docs.google.com/java/awt/Component.html#CENTER_ALIGNMENT), [LEFT\_ALIGNMENT](http://docs.google.com/java/awt/Component.html#LEFT_ALIGNMENT), [RIGHT\_ALIGNMENT](http://docs.google.com/java/awt/Component.html#RIGHT_ALIGNMENT), [TOP\_ALIGNMENT](http://docs.google.com/java/awt/Component.html#TOP_ALIGNMENT) |

| **Fields inherited from interface java.awt.image.**[**ImageObserver**](http://docs.google.com/java/awt/image/ImageObserver.html) |
| --- |
| [ABORT](http://docs.google.com/java/awt/image/ImageObserver.html#ABORT), [ALLBITS](http://docs.google.com/java/awt/image/ImageObserver.html#ALLBITS), [ERROR](http://docs.google.com/java/awt/image/ImageObserver.html#ERROR), [FRAMEBITS](http://docs.google.com/java/awt/image/ImageObserver.html#FRAMEBITS), [HEIGHT](http://docs.google.com/java/awt/image/ImageObserver.html#HEIGHT), [PROPERTIES](http://docs.google.com/java/awt/image/ImageObserver.html#PROPERTIES), [SOMEBITS](http://docs.google.com/java/awt/image/ImageObserver.html#SOMEBITS), [WIDTH](http://docs.google.com/java/awt/image/ImageObserver.html#WIDTH) |

| **Constructor Summary** | |
| --- | --- |
| [**JComponent**](http://docs.google.com/javax/swing/JComponent.html#JComponent())()            Default JComponent constructor. |

| **Method Summary** | |
| --- | --- |
| void | [**addAncestorListener**](http://docs.google.com/javax/swing/JComponent.html#addAncestorListener(javax.swing.event.AncestorListener))([AncestorListener](http://docs.google.com/javax/swing/event/AncestorListener.html) listener)            Registers listener so that it will receive AncestorEvents when it or any of its ancestors move or are made visible or invisible. |
| void | [**addNotify**](http://docs.google.com/javax/swing/JComponent.html#addNotify())()            Notifies this component that it now has a parent component. |
| void | [**addVetoableChangeListener**](http://docs.google.com/javax/swing/JComponent.html#addVetoableChangeListener(java.beans.VetoableChangeListener))([VetoableChangeListener](http://docs.google.com/java/beans/VetoableChangeListener.html) listener)            Adds a VetoableChangeListener to the listener list. |
| void | [**computeVisibleRect**](http://docs.google.com/javax/swing/JComponent.html#computeVisibleRect(java.awt.Rectangle))([Rectangle](http://docs.google.com/java/awt/Rectangle.html) visibleRect)            Returns the Component's "visible rect rectangle" - the intersection of the visible rectangles for this component and all of its ancestors. |
| boolean | [**contains**](http://docs.google.com/javax/swing/JComponent.html#contains(int,%20int))(int x, int y)            Gives the UI delegate an opportunity to define the precise shape of this component for the sake of mouse processing. |
| [JToolTip](http://docs.google.com/javax/swing/JToolTip.html) | [**createToolTip**](http://docs.google.com/javax/swing/JComponent.html#createToolTip())()            Returns the instance of JToolTip that should be used to display the tooltip. |
| void | [**disable**](http://docs.google.com/javax/swing/JComponent.html#disable())()  **Deprecated.** *As of JDK version 1.1, replaced by java.awt.Component.setEnabled(boolean).* |
| void | [**enable**](http://docs.google.com/javax/swing/JComponent.html#enable())()  **Deprecated.** *As of JDK version 1.1, replaced by java.awt.Component.setEnabled(boolean).* |
| void | [**firePropertyChange**](http://docs.google.com/javax/swing/JComponent.html#firePropertyChange(java.lang.String,%20boolean,%20boolean))([String](http://docs.google.com/java/lang/String.html) propertyName, boolean oldValue, boolean newValue)            Support for reporting bound property changes for boolean properties. |
| void | [**firePropertyChange**](http://docs.google.com/javax/swing/JComponent.html#firePropertyChange(java.lang.String,%20char,%20char))([String](http://docs.google.com/java/lang/String.html) propertyName, char oldValue, char newValue)            Reports a bound property change. |
| void | [**firePropertyChange**](http://docs.google.com/javax/swing/JComponent.html#firePropertyChange(java.lang.String,%20int,%20int))([String](http://docs.google.com/java/lang/String.html) propertyName, int oldValue, int newValue)            Support for reporting bound property changes for integer properties. |
| protected  void | [**fireVetoableChange**](http://docs.google.com/javax/swing/JComponent.html#fireVetoableChange(java.lang.String,%20java.lang.Object,%20java.lang.Object))([String](http://docs.google.com/java/lang/String.html) propertyName, [Object](http://docs.google.com/java/lang/Object.html) oldValue, [Object](http://docs.google.com/java/lang/Object.html) newValue)            Supports reporting constrained property changes. |
| [AccessibleContext](http://docs.google.com/javax/accessibility/AccessibleContext.html) | [**getAccessibleContext**](http://docs.google.com/javax/swing/JComponent.html#getAccessibleContext())()            Returns the AccessibleContext associated with this JComponent. |
| [ActionListener](http://docs.google.com/java/awt/event/ActionListener.html) | [**getActionForKeyStroke**](http://docs.google.com/javax/swing/JComponent.html#getActionForKeyStroke(javax.swing.KeyStroke))([KeyStroke](http://docs.google.com/javax/swing/KeyStroke.html) aKeyStroke)            Returns the object that will perform the action registered for a given keystroke. |
| [ActionMap](http://docs.google.com/javax/swing/ActionMap.html) | [**getActionMap**](http://docs.google.com/javax/swing/JComponent.html#getActionMap())()            Returns the ActionMap used to determine what Action to fire for particular KeyStroke binding. |
| float | [**getAlignmentX**](http://docs.google.com/javax/swing/JComponent.html#getAlignmentX())()            Overrides Container.getAlignmentX to return the vertical alignment. |
| float | [**getAlignmentY**](http://docs.google.com/javax/swing/JComponent.html#getAlignmentY())()            Overrides Container.getAlignmentY to return the horizontal alignment. |
| [AncestorListener](http://docs.google.com/javax/swing/event/AncestorListener.html)[] | [**getAncestorListeners**](http://docs.google.com/javax/swing/JComponent.html#getAncestorListeners())()            Returns an array of all the ancestor listeners registered on this component. |
| boolean | [**getAutoscrolls**](http://docs.google.com/javax/swing/JComponent.html#getAutoscrolls())()            Gets the autoscrolls property. |
| int | [**getBaseline**](http://docs.google.com/javax/swing/JComponent.html#getBaseline(int,%20int))(int width, int height)            Returns the baseline. |
| [Component.BaselineResizeBehavior](http://docs.google.com/java/awt/Component.BaselineResizeBehavior.html) | [**getBaselineResizeBehavior**](http://docs.google.com/javax/swing/JComponent.html#getBaselineResizeBehavior())()            Returns an enum indicating how the baseline of the component changes as the size changes. |
| [Border](http://docs.google.com/javax/swing/border/Border.html) | [**getBorder**](http://docs.google.com/javax/swing/JComponent.html#getBorder())()            Returns the border of this component or null if no border is currently set. |
| [Rectangle](http://docs.google.com/java/awt/Rectangle.html) | [**getBounds**](http://docs.google.com/javax/swing/JComponent.html#getBounds(java.awt.Rectangle))([Rectangle](http://docs.google.com/java/awt/Rectangle.html) rv)            Stores the bounds of this component into "return value" rv and returns rv. |
| [Object](http://docs.google.com/java/lang/Object.html) | [**getClientProperty**](http://docs.google.com/javax/swing/JComponent.html#getClientProperty(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) key)            Returns the value of the property with the specified key. |
| protected  [Graphics](http://docs.google.com/java/awt/Graphics.html) | [**getComponentGraphics**](http://docs.google.com/javax/swing/JComponent.html#getComponentGraphics(java.awt.Graphics))([Graphics](http://docs.google.com/java/awt/Graphics.html) g)            Returns the graphics object used to paint this component. |
| [JPopupMenu](http://docs.google.com/javax/swing/JPopupMenu.html) | [**getComponentPopupMenu**](http://docs.google.com/javax/swing/JComponent.html#getComponentPopupMenu())()            Returns JPopupMenu that assigned for this component. |
| int | [**getConditionForKeyStroke**](http://docs.google.com/javax/swing/JComponent.html#getConditionForKeyStroke(javax.swing.KeyStroke))([KeyStroke](http://docs.google.com/javax/swing/KeyStroke.html) aKeyStroke)            Returns the condition that determines whether a registered action occurs in response to the specified keystroke. |
| int | [**getDebugGraphicsOptions**](http://docs.google.com/javax/swing/JComponent.html#getDebugGraphicsOptions())()            Returns the state of graphics debugging. |
| static [Locale](http://docs.google.com/java/util/Locale.html) | [**getDefaultLocale**](http://docs.google.com/javax/swing/JComponent.html#getDefaultLocale())()            Returns the default locale used to initialize each JComponent's locale property upon creation. |
| [FontMetrics](http://docs.google.com/java/awt/FontMetrics.html) | [**getFontMetrics**](http://docs.google.com/javax/swing/JComponent.html#getFontMetrics(java.awt.Font))([Font](http://docs.google.com/java/awt/Font.html) font)            Gets the FontMetrics for the specified Font. |
| [Graphics](http://docs.google.com/java/awt/Graphics.html) | [**getGraphics**](http://docs.google.com/javax/swing/JComponent.html#getGraphics())()            Returns this component's graphics context, which lets you draw on a component. |
| int | [**getHeight**](http://docs.google.com/javax/swing/JComponent.html#getHeight())()            Returns the current height of this component. |
| boolean | [**getInheritsPopupMenu**](http://docs.google.com/javax/swing/JComponent.html#getInheritsPopupMenu())()            Returns true if the JPopupMenu should be inherited from the parent. |
| [InputMap](http://docs.google.com/javax/swing/InputMap.html) | [**getInputMap**](http://docs.google.com/javax/swing/JComponent.html#getInputMap())()            Returns the InputMap that is used when the component has focus. |
| [InputMap](http://docs.google.com/javax/swing/InputMap.html) | [**getInputMap**](http://docs.google.com/javax/swing/JComponent.html#getInputMap(int))(int condition)            Returns the InputMap that is used during condition. |
| [InputVerifier](http://docs.google.com/javax/swing/InputVerifier.html) | [**getInputVerifier**](http://docs.google.com/javax/swing/JComponent.html#getInputVerifier())()            Returns the input verifier for this component. |
| [Insets](http://docs.google.com/java/awt/Insets.html) | [**getInsets**](http://docs.google.com/javax/swing/JComponent.html#getInsets())()            If a border has been set on this component, returns the border's insets; otherwise calls super.getInsets. |
| [Insets](http://docs.google.com/java/awt/Insets.html) | [**getInsets**](http://docs.google.com/javax/swing/JComponent.html#getInsets(java.awt.Insets))([Insets](http://docs.google.com/java/awt/Insets.html) insets)            Returns an Insets object containing this component's inset values. |
| | <T extends [EventListener](http://docs.google.com/java/util/EventListener.html)>  T[] | | --- | | [**getListeners**](http://docs.google.com/javax/swing/JComponent.html#getListeners(java.lang.Class))([Class](http://docs.google.com/java/lang/Class.html)<T> listenerType)            Returns an array of all the objects currently registered as *Foo*Listeners upon this JComponent. |
| [Point](http://docs.google.com/java/awt/Point.html) | [**getLocation**](http://docs.google.com/javax/swing/JComponent.html#getLocation(java.awt.Point))([Point](http://docs.google.com/java/awt/Point.html) rv)            Stores the x,y origin of this component into "return value" rv and returns rv. |
| [Dimension](http://docs.google.com/java/awt/Dimension.html) | [**getMaximumSize**](http://docs.google.com/javax/swing/JComponent.html#getMaximumSize())()            If the maximum size has been set to a non-null value just returns it. |
| [Dimension](http://docs.google.com/java/awt/Dimension.html) | [**getMinimumSize**](http://docs.google.com/javax/swing/JComponent.html#getMinimumSize())()            If the minimum size has been set to a non-null value just returns it. |
| [Component](http://docs.google.com/java/awt/Component.html) | [**getNextFocusableComponent**](http://docs.google.com/javax/swing/JComponent.html#getNextFocusableComponent())()  **Deprecated.** *As of 1.4, replaced by FocusTraversalPolicy.* |
| [Point](http://docs.google.com/java/awt/Point.html) | [**getPopupLocation**](http://docs.google.com/javax/swing/JComponent.html#getPopupLocation(java.awt.event.MouseEvent))([MouseEvent](http://docs.google.com/java/awt/event/MouseEvent.html) event)            Returns the preferred location to display the popup menu in this component's coordinate system. |
| [Dimension](http://docs.google.com/java/awt/Dimension.html) | [**getPreferredSize**](http://docs.google.com/javax/swing/JComponent.html#getPreferredSize())()            If the preferredSize has been set to a non-null value just returns it. |
| [KeyStroke](http://docs.google.com/javax/swing/KeyStroke.html)[] | [**getRegisteredKeyStrokes**](http://docs.google.com/javax/swing/JComponent.html#getRegisteredKeyStrokes())()            Returns the KeyStrokes that will initiate registered actions. |
| [JRootPane](http://docs.google.com/javax/swing/JRootPane.html) | [**getRootPane**](http://docs.google.com/javax/swing/JComponent.html#getRootPane())()            Returns the JRootPane ancestor for this component. |
| [Dimension](http://docs.google.com/java/awt/Dimension.html) | [**getSize**](http://docs.google.com/javax/swing/JComponent.html#getSize(java.awt.Dimension))([Dimension](http://docs.google.com/java/awt/Dimension.html) rv)            Stores the width/height of this component into "return value" rv and returns rv. |
| [Point](http://docs.google.com/java/awt/Point.html) | [**getToolTipLocation**](http://docs.google.com/javax/swing/JComponent.html#getToolTipLocation(java.awt.event.MouseEvent))([MouseEvent](http://docs.google.com/java/awt/event/MouseEvent.html) event)            Returns the tooltip location in this component's coordinate system. |
| [String](http://docs.google.com/java/lang/String.html) | [**getToolTipText**](http://docs.google.com/javax/swing/JComponent.html#getToolTipText())()            Returns the tooltip string that has been set with setToolTipText. |
| [String](http://docs.google.com/java/lang/String.html) | [**getToolTipText**](http://docs.google.com/javax/swing/JComponent.html#getToolTipText(java.awt.event.MouseEvent))([MouseEvent](http://docs.google.com/java/awt/event/MouseEvent.html) event)            Returns the string to be used as the tooltip for *event*. |
| [Container](http://docs.google.com/java/awt/Container.html) | [**getTopLevelAncestor**](http://docs.google.com/javax/swing/JComponent.html#getTopLevelAncestor())()            Returns the top-level ancestor of this component (either the containing Window or Applet), or null if this component has not been added to any container. |
| [TransferHandler](http://docs.google.com/javax/swing/TransferHandler.html) | [**getTransferHandler**](http://docs.google.com/javax/swing/JComponent.html#getTransferHandler())()            Gets the transferHandler property. |
| [String](http://docs.google.com/java/lang/String.html) | [**getUIClassID**](http://docs.google.com/javax/swing/JComponent.html#getUIClassID())()            Returns the UIDefaults key used to look up the name of the swing.plaf.ComponentUI class that defines the look and feel for this component. |
| boolean | [**getVerifyInputWhenFocusTarget**](http://docs.google.com/javax/swing/JComponent.html#getVerifyInputWhenFocusTarget())()            Returns the value that indicates whether the input verifier for the current focus owner will be called before this component requests focus. |
| [VetoableChangeListener](http://docs.google.com/java/beans/VetoableChangeListener.html)[] | [**getVetoableChangeListeners**](http://docs.google.com/javax/swing/JComponent.html#getVetoableChangeListeners())()            Returns an array of all the vetoable change listeners registered on this component. |
| [Rectangle](http://docs.google.com/java/awt/Rectangle.html) | [**getVisibleRect**](http://docs.google.com/javax/swing/JComponent.html#getVisibleRect())()            Returns the Component's "visible rectangle" - the intersection of this component's visible rectangle, new Rectangle(0, 0, getWidth(), getHeight()), and all of its ancestors' visible rectangles. |
| int | [**getWidth**](http://docs.google.com/javax/swing/JComponent.html#getWidth())()            Returns the current width of this component. |
| int | [**getX**](http://docs.google.com/javax/swing/JComponent.html#getX())()            Returns the current x coordinate of the component's origin. |
| int | [**getY**](http://docs.google.com/javax/swing/JComponent.html#getY())()            Returns the current y coordinate of the component's origin. |
| void | [**grabFocus**](http://docs.google.com/javax/swing/JComponent.html#grabFocus())()            Requests that this Component get the input focus, and that this Component's top-level ancestor become the focused Window. |
| boolean | [**isDoubleBuffered**](http://docs.google.com/javax/swing/JComponent.html#isDoubleBuffered())()            Returns whether this component should use a buffer to paint. |
| static boolean | [**isLightweightComponent**](http://docs.google.com/javax/swing/JComponent.html#isLightweightComponent(java.awt.Component))([Component](http://docs.google.com/java/awt/Component.html) c)            Returns true if this component is lightweight, that is, if it doesn't have a native window system peer. |
| boolean | [**isManagingFocus**](http://docs.google.com/javax/swing/JComponent.html#isManagingFocus())()  **Deprecated.** *As of 1.4, replaced by Component.setFocusTraversalKeys(int, Set) and Container.setFocusCycleRoot(boolean).* |
| boolean | [**isOpaque**](http://docs.google.com/javax/swing/JComponent.html#isOpaque())()            Returns true if this component is completely opaque. |
| boolean | [**isOptimizedDrawingEnabled**](http://docs.google.com/javax/swing/JComponent.html#isOptimizedDrawingEnabled())()            Returns true if this component tiles its children -- that is, if it can guarantee that the children will not overlap. |
| boolean | [**isPaintingForPrint**](http://docs.google.com/javax/swing/JComponent.html#isPaintingForPrint())()            Returns true if the current painting operation on this component is part of a print operation. |
| boolean | [**isPaintingTile**](http://docs.google.com/javax/swing/JComponent.html#isPaintingTile())()            Returns true if the component is currently painting a tile. |
| boolean | [**isRequestFocusEnabled**](http://docs.google.com/javax/swing/JComponent.html#isRequestFocusEnabled())()            Returns true if this JComponent should get focus; otherwise returns false. |
| boolean | [**isValidateRoot**](http://docs.google.com/javax/swing/JComponent.html#isValidateRoot())()            If this method returns true, revalidate calls by descendants of this component will cause the entire tree beginning with this root to be validated. |
| void | [**paint**](http://docs.google.com/javax/swing/JComponent.html#paint(java.awt.Graphics))([Graphics](http://docs.google.com/java/awt/Graphics.html) g)            Invoked by Swing to draw components. |
| protected  void | [**paintBorder**](http://docs.google.com/javax/swing/JComponent.html#paintBorder(java.awt.Graphics))([Graphics](http://docs.google.com/java/awt/Graphics.html) g)            Paints the component's border. |
| protected  void | [**paintChildren**](http://docs.google.com/javax/swing/JComponent.html#paintChildren(java.awt.Graphics))([Graphics](http://docs.google.com/java/awt/Graphics.html) g)            Paints this component's children. |
| protected  void | [**paintComponent**](http://docs.google.com/javax/swing/JComponent.html#paintComponent(java.awt.Graphics))([Graphics](http://docs.google.com/java/awt/Graphics.html) g)            Calls the UI delegate's paint method, if the UI delegate is non-null. |
| void | [**paintImmediately**](http://docs.google.com/javax/swing/JComponent.html#paintImmediately(int,%20int,%20int,%20int))(int x, int y, int w, int h)            Paints the specified region in this component and all of its descendants that overlap the region, immediately. |
| void | [**paintImmediately**](http://docs.google.com/javax/swing/JComponent.html#paintImmediately(java.awt.Rectangle))([Rectangle](http://docs.google.com/java/awt/Rectangle.html) r)            Paints the specified region now. |
| protected  [String](http://docs.google.com/java/lang/String.html) | [**paramString**](http://docs.google.com/javax/swing/JComponent.html#paramString())()            Returns a string representation of this JComponent. |
| void | [**print**](http://docs.google.com/javax/swing/JComponent.html#print(java.awt.Graphics))([Graphics](http://docs.google.com/java/awt/Graphics.html) g)            Invoke this method to print the component to the specified Graphics. |
| void | [**printAll**](http://docs.google.com/javax/swing/JComponent.html#printAll(java.awt.Graphics))([Graphics](http://docs.google.com/java/awt/Graphics.html) g)            Invoke this method to print the component. |
| protected  void | [**printBorder**](http://docs.google.com/javax/swing/JComponent.html#printBorder(java.awt.Graphics))([Graphics](http://docs.google.com/java/awt/Graphics.html) g)            Prints the component's border. |
| protected  void | [**printChildren**](http://docs.google.com/javax/swing/JComponent.html#printChildren(java.awt.Graphics))([Graphics](http://docs.google.com/java/awt/Graphics.html) g)            Prints this component's children. |
| protected  void | [**printComponent**](http://docs.google.com/javax/swing/JComponent.html#printComponent(java.awt.Graphics))([Graphics](http://docs.google.com/java/awt/Graphics.html) g)            This is invoked during a printing operation. |
| protected  void | [**processComponentKeyEvent**](http://docs.google.com/javax/swing/JComponent.html#processComponentKeyEvent(java.awt.event.KeyEvent))([KeyEvent](http://docs.google.com/java/awt/event/KeyEvent.html) e)            Processes any key events that the component itself recognizes. |
| protected  boolean | [**processKeyBinding**](http://docs.google.com/javax/swing/JComponent.html#processKeyBinding(javax.swing.KeyStroke,%20java.awt.event.KeyEvent,%20int,%20boolean))([KeyStroke](http://docs.google.com/javax/swing/KeyStroke.html) ks, [KeyEvent](http://docs.google.com/java/awt/event/KeyEvent.html) e, int condition, boolean pressed)            Invoked to process the key bindings for ks as the result of the KeyEvent e. |
| protected  void | [**processKeyEvent**](http://docs.google.com/javax/swing/JComponent.html#processKeyEvent(java.awt.event.KeyEvent))([KeyEvent](http://docs.google.com/java/awt/event/KeyEvent.html) e)            Overrides processKeyEvent to process events. |
| protected  void | [**processMouseEvent**](http://docs.google.com/javax/swing/JComponent.html#processMouseEvent(java.awt.event.MouseEvent))([MouseEvent](http://docs.google.com/java/awt/event/MouseEvent.html) e)            Processes mouse events occurring on this component by dispatching them to any registered MouseListener objects, refer to [Component.processMouseEvent(MouseEvent)](http://docs.google.com/java/awt/Component.html#processMouseEvent(java.awt.event.MouseEvent)) for a complete description of this method. |
| protected  void | [**processMouseMotionEvent**](http://docs.google.com/javax/swing/JComponent.html#processMouseMotionEvent(java.awt.event.MouseEvent))([MouseEvent](http://docs.google.com/java/awt/event/MouseEvent.html) e)            Processes mouse motion events, such as MouseEvent.MOUSE\_DRAGGED. |
| void | [**putClientProperty**](http://docs.google.com/javax/swing/JComponent.html#putClientProperty(java.lang.Object,%20java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) key, [Object](http://docs.google.com/java/lang/Object.html) value)            Adds an arbitrary key/value "client property" to this component. |
| void | [**registerKeyboardAction**](http://docs.google.com/javax/swing/JComponent.html#registerKeyboardAction(java.awt.event.ActionListener,%20javax.swing.KeyStroke,%20int))([ActionListener](http://docs.google.com/java/awt/event/ActionListener.html) anAction, [KeyStroke](http://docs.google.com/javax/swing/KeyStroke.html) aKeyStroke, int aCondition)            This method is now obsolete, please use a combination of getActionMap() and getInputMap() for similiar behavior. |
| void | [**registerKeyboardAction**](http://docs.google.com/javax/swing/JComponent.html#registerKeyboardAction(java.awt.event.ActionListener,%20java.lang.String,%20javax.swing.KeyStroke,%20int))([ActionListener](http://docs.google.com/java/awt/event/ActionListener.html) anAction, [String](http://docs.google.com/java/lang/String.html) aCommand, [KeyStroke](http://docs.google.com/javax/swing/KeyStroke.html) aKeyStroke, int aCondition)            This method is now obsolete, please use a combination of getActionMap() and getInputMap() for similiar behavior. |
| void | [**removeAncestorListener**](http://docs.google.com/javax/swing/JComponent.html#removeAncestorListener(javax.swing.event.AncestorListener))([AncestorListener](http://docs.google.com/javax/swing/event/AncestorListener.html) listener)            Unregisters listener so that it will no longer receive AncestorEvents. |
| void | [**removeNotify**](http://docs.google.com/javax/swing/JComponent.html#removeNotify())()            Notifies this component that it no longer has a parent component. |
| void | [**removeVetoableChangeListener**](http://docs.google.com/javax/swing/JComponent.html#removeVetoableChangeListener(java.beans.VetoableChangeListener))([VetoableChangeListener](http://docs.google.com/java/beans/VetoableChangeListener.html) listener)            Removes a VetoableChangeListener from the listener list. |
| void | [**repaint**](http://docs.google.com/javax/swing/JComponent.html#repaint(long,%20int,%20int,%20int,%20int))(long tm, int x, int y, int width, int height)            Adds the specified region to the dirty region list if the component is showing. |
| void | [**repaint**](http://docs.google.com/javax/swing/JComponent.html#repaint(java.awt.Rectangle))([Rectangle](http://docs.google.com/java/awt/Rectangle.html) r)            Adds the specified region to the dirty region list if the component is showing. |
| boolean | [**requestDefaultFocus**](http://docs.google.com/javax/swing/JComponent.html#requestDefaultFocus())()  **Deprecated.** *As of 1.4, replaced by FocusTraversalPolicy.getDefaultComponent(Container).requestFocus()* |
| void | [**requestFocus**](http://docs.google.com/javax/swing/JComponent.html#requestFocus())()            Requests that this Component gets the input focus. |
| boolean | [**requestFocus**](http://docs.google.com/javax/swing/JComponent.html#requestFocus(boolean))(boolean temporary)            Requests that this Component gets the input focus. |
| boolean | [**requestFocusInWindow**](http://docs.google.com/javax/swing/JComponent.html#requestFocusInWindow())()            Requests that this Component gets the input focus. |
| protected  boolean | [**requestFocusInWindow**](http://docs.google.com/javax/swing/JComponent.html#requestFocusInWindow(boolean))(boolean temporary)            Requests that this Component gets the input focus. |
| void | [**resetKeyboardActions**](http://docs.google.com/javax/swing/JComponent.html#resetKeyboardActions())()            Unregisters all the bindings in the first tier InputMaps and ActionMap. |
| void | [**reshape**](http://docs.google.com/javax/swing/JComponent.html#reshape(int,%20int,%20int,%20int))(int x, int y, int w, int h)  **Deprecated.** *As of JDK 5, replaced by Component.setBounds(int, int, int, int).*  *Moves and resizes this component.* |
| void | [**revalidate**](http://docs.google.com/javax/swing/JComponent.html#revalidate())()            Supports deferred automatic layout. |
| void | [**scrollRectToVisible**](http://docs.google.com/javax/swing/JComponent.html#scrollRectToVisible(java.awt.Rectangle))([Rectangle](http://docs.google.com/java/awt/Rectangle.html) aRect)            Forwards the scrollRectToVisible() message to the JComponent's parent. |
| void | [**setActionMap**](http://docs.google.com/javax/swing/JComponent.html#setActionMap(javax.swing.ActionMap))([ActionMap](http://docs.google.com/javax/swing/ActionMap.html) am)            Sets the ActionMap to am. |
| void | [**setAlignmentX**](http://docs.google.com/javax/swing/JComponent.html#setAlignmentX(float))(float alignmentX)            Sets the the vertical alignment. |
| void | [**setAlignmentY**](http://docs.google.com/javax/swing/JComponent.html#setAlignmentY(float))(float alignmentY)            Sets the the horizontal alignment. |
| void | [**setAutoscrolls**](http://docs.google.com/javax/swing/JComponent.html#setAutoscrolls(boolean))(boolean autoscrolls)            Sets the autoscrolls property. |
| void | [**setBackground**](http://docs.google.com/javax/swing/JComponent.html#setBackground(java.awt.Color))([Color](http://docs.google.com/java/awt/Color.html) bg)            Sets the background color of this component. |
| void | [**setBorder**](http://docs.google.com/javax/swing/JComponent.html#setBorder(javax.swing.border.Border))([Border](http://docs.google.com/javax/swing/border/Border.html) border)            Sets the border of this component. |
| void | [**setComponentPopupMenu**](http://docs.google.com/javax/swing/JComponent.html#setComponentPopupMenu(javax.swing.JPopupMenu))([JPopupMenu](http://docs.google.com/javax/swing/JPopupMenu.html) popup)            Sets the JPopupMenu for this JComponent. |
| void | [**setDebugGraphicsOptions**](http://docs.google.com/javax/swing/JComponent.html#setDebugGraphicsOptions(int))(int debugOptions)            Enables or disables diagnostic information about every graphics operation performed within the component or one of its children. |
| static void | [**setDefaultLocale**](http://docs.google.com/javax/swing/JComponent.html#setDefaultLocale(java.util.Locale))([Locale](http://docs.google.com/java/util/Locale.html) l)            Sets the default locale used to initialize each JComponent's locale property upon creation. |
| void | [**setDoubleBuffered**](http://docs.google.com/javax/swing/JComponent.html#setDoubleBuffered(boolean))(boolean aFlag)            Sets whether this component should use a buffer to paint. |
| void | [**setEnabled**](http://docs.google.com/javax/swing/JComponent.html#setEnabled(boolean))(boolean enabled)            Sets whether or not this component is enabled. |
| void | [**setFocusTraversalKeys**](http://docs.google.com/javax/swing/JComponent.html#setFocusTraversalKeys(int,%20java.util.Set))(int id, [Set](http://docs.google.com/java/util/Set.html)<? extends [AWTKeyStroke](http://docs.google.com/java/awt/AWTKeyStroke.html)> keystrokes)            Sets the focus traversal keys for a given traversal operation for this Component. |
| void | [**setFont**](http://docs.google.com/javax/swing/JComponent.html#setFont(java.awt.Font))([Font](http://docs.google.com/java/awt/Font.html) font)            Sets the font for this component. |
| void | [**setForeground**](http://docs.google.com/javax/swing/JComponent.html#setForeground(java.awt.Color))([Color](http://docs.google.com/java/awt/Color.html) fg)            Sets the foreground color of this component. |
| void | [**setInheritsPopupMenu**](http://docs.google.com/javax/swing/JComponent.html#setInheritsPopupMenu(boolean))(boolean value)            Sets whether or not getComponentPopupMenu should delegate to the parent if this component does not have a JPopupMenu assigned to it. |
| void | [**setInputMap**](http://docs.google.com/javax/swing/JComponent.html#setInputMap(int,%20javax.swing.InputMap))(int condition, [InputMap](http://docs.google.com/javax/swing/InputMap.html) map)            Sets the InputMap to use under the condition condition to map. |
| void | [**setInputVerifier**](http://docs.google.com/javax/swing/JComponent.html#setInputVerifier(javax.swing.InputVerifier))([InputVerifier](http://docs.google.com/javax/swing/InputVerifier.html) inputVerifier)            Sets the input verifier for this component. |
| void | [**setMaximumSize**](http://docs.google.com/javax/swing/JComponent.html#setMaximumSize(java.awt.Dimension))([Dimension](http://docs.google.com/java/awt/Dimension.html) maximumSize)            Sets the maximum size of this component to a constant value. |
| void | [**setMinimumSize**](http://docs.google.com/javax/swing/JComponent.html#setMinimumSize(java.awt.Dimension))([Dimension](http://docs.google.com/java/awt/Dimension.html) minimumSize)            Sets the minimum size of this component to a constant value. |
| void | [**setNextFocusableComponent**](http://docs.google.com/javax/swing/JComponent.html#setNextFocusableComponent(java.awt.Component))([Component](http://docs.google.com/java/awt/Component.html) aComponent)  **Deprecated.** *As of 1.4, replaced by FocusTraversalPolicy* |
| void | [**setOpaque**](http://docs.google.com/javax/swing/JComponent.html#setOpaque(boolean))(boolean isOpaque)            If true the component paints every pixel within its bounds. |
| void | [**setPreferredSize**](http://docs.google.com/javax/swing/JComponent.html#setPreferredSize(java.awt.Dimension))([Dimension](http://docs.google.com/java/awt/Dimension.html) preferredSize)            Sets the preferred size of this component. |
| void | [**setRequestFocusEnabled**](http://docs.google.com/javax/swing/JComponent.html#setRequestFocusEnabled(boolean))(boolean requestFocusEnabled)            Provides a hint as to whether or not this JComponent should get focus. |
| void | [**setToolTipText**](http://docs.google.com/javax/swing/JComponent.html#setToolTipText(java.lang.String))([String](http://docs.google.com/java/lang/String.html) text)            Registers the text to display in a tool tip. |
| void | [**setTransferHandler**](http://docs.google.com/javax/swing/JComponent.html#setTransferHandler(javax.swing.TransferHandler))([TransferHandler](http://docs.google.com/javax/swing/TransferHandler.html) newHandler)            Sets the transferHandler property, which is null if the component does not support data transfer operations. |
| protected  void | [**setUI**](http://docs.google.com/javax/swing/JComponent.html#setUI(javax.swing.plaf.ComponentUI))([ComponentUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html) newUI)            Sets the look and feel delegate for this component. |
| void | [**setVerifyInputWhenFocusTarget**](http://docs.google.com/javax/swing/JComponent.html#setVerifyInputWhenFocusTarget(boolean))(boolean verifyInputWhenFocusTarget)            Sets the value to indicate whether input verifier for the current focus owner will be called before this component requests focus. |
| void | [**setVisible**](http://docs.google.com/javax/swing/JComponent.html#setVisible(boolean))(boolean aFlag)            Makes the component visible or invisible. |
| void | [**unregisterKeyboardAction**](http://docs.google.com/javax/swing/JComponent.html#unregisterKeyboardAction(javax.swing.KeyStroke))([KeyStroke](http://docs.google.com/javax/swing/KeyStroke.html) aKeyStroke)            This method is now obsolete. |
| void | [**update**](http://docs.google.com/javax/swing/JComponent.html#update(java.awt.Graphics))([Graphics](http://docs.google.com/java/awt/Graphics.html) g)            Calls paint. |
| void | [**updateUI**](http://docs.google.com/javax/swing/JComponent.html#updateUI())()            Resets the UI property to a value from the current look and feel. |

| **Methods inherited from class java.awt.**[**Container**](http://docs.google.com/java/awt/Container.html) |
| --- |
| [add](http://docs.google.com/java/awt/Container.html#add(java.awt.Component)), [add](http://docs.google.com/java/awt/Container.html#add(java.awt.Component,%20int)), [add](http://docs.google.com/java/awt/Container.html#add(java.awt.Component,%20java.lang.Object)), [add](http://docs.google.com/java/awt/Container.html#add(java.awt.Component,%20java.lang.Object,%20int)), [add](http://docs.google.com/java/awt/Container.html#add(java.lang.String,%20java.awt.Component)), [addContainerListener](http://docs.google.com/java/awt/Container.html#addContainerListener(java.awt.event.ContainerListener)), [addImpl](http://docs.google.com/java/awt/Container.html#addImpl(java.awt.Component,%20java.lang.Object,%20int)), [addPropertyChangeListener](http://docs.google.com/java/awt/Container.html#addPropertyChangeListener(java.beans.PropertyChangeListener)), [addPropertyChangeListener](http://docs.google.com/java/awt/Container.html#addPropertyChangeListener(java.lang.String,%20java.beans.PropertyChangeListener)), [applyComponentOrientation](http://docs.google.com/java/awt/Container.html#applyComponentOrientation(java.awt.ComponentOrientation)), [areFocusTraversalKeysSet](http://docs.google.com/java/awt/Container.html#areFocusTraversalKeysSet(int)), [countComponents](http://docs.google.com/java/awt/Container.html#countComponents()), [deliverEvent](http://docs.google.com/java/awt/Container.html#deliverEvent(java.awt.Event)), [doLayout](http://docs.google.com/java/awt/Container.html#doLayout()), [findComponentAt](http://docs.google.com/java/awt/Container.html#findComponentAt(int,%20int)), [findComponentAt](http://docs.google.com/java/awt/Container.html#findComponentAt(java.awt.Point)), [getComponent](http://docs.google.com/java/awt/Container.html#getComponent(int)), [getComponentAt](http://docs.google.com/java/awt/Container.html#getComponentAt(int,%20int)), [getComponentAt](http://docs.google.com/java/awt/Container.html#getComponentAt(java.awt.Point)), [getComponentCount](http://docs.google.com/java/awt/Container.html#getComponentCount()), [getComponents](http://docs.google.com/java/awt/Container.html#getComponents()), [getComponentZOrder](http://docs.google.com/java/awt/Container.html#getComponentZOrder(java.awt.Component)), [getContainerListeners](http://docs.google.com/java/awt/Container.html#getContainerListeners()), [getFocusTraversalKeys](http://docs.google.com/java/awt/Container.html#getFocusTraversalKeys(int)), [getFocusTraversalPolicy](http://docs.google.com/java/awt/Container.html#getFocusTraversalPolicy()), [getLayout](http://docs.google.com/java/awt/Container.html#getLayout()), [getMousePosition](http://docs.google.com/java/awt/Container.html#getMousePosition(boolean)), [insets](http://docs.google.com/java/awt/Container.html#insets()), [invalidate](http://docs.google.com/java/awt/Container.html#invalidate()), [isAncestorOf](http://docs.google.com/java/awt/Container.html#isAncestorOf(java.awt.Component)), [isFocusCycleRoot](http://docs.google.com/java/awt/Container.html#isFocusCycleRoot()), [isFocusCycleRoot](http://docs.google.com/java/awt/Container.html#isFocusCycleRoot(java.awt.Container)), [isFocusTraversalPolicyProvider](http://docs.google.com/java/awt/Container.html#isFocusTraversalPolicyProvider()), [isFocusTraversalPolicySet](http://docs.google.com/java/awt/Container.html#isFocusTraversalPolicySet()), [layout](http://docs.google.com/java/awt/Container.html#layout()), [list](http://docs.google.com/java/awt/Container.html#list(java.io.PrintStream,%20int)), [list](http://docs.google.com/java/awt/Container.html#list(java.io.PrintWriter,%20int)), [locate](http://docs.google.com/java/awt/Container.html#locate(int,%20int)), [minimumSize](http://docs.google.com/java/awt/Container.html#minimumSize()), [paintComponents](http://docs.google.com/java/awt/Container.html#paintComponents(java.awt.Graphics)), [preferredSize](http://docs.google.com/java/awt/Container.html#preferredSize()), [printComponents](http://docs.google.com/java/awt/Container.html#printComponents(java.awt.Graphics)), [processContainerEvent](http://docs.google.com/java/awt/Container.html#processContainerEvent(java.awt.event.ContainerEvent)), [processEvent](http://docs.google.com/java/awt/Container.html#processEvent(java.awt.AWTEvent)), [remove](http://docs.google.com/java/awt/Container.html#remove(java.awt.Component)), [remove](http://docs.google.com/java/awt/Container.html#remove(int)), [removeAll](http://docs.google.com/java/awt/Container.html#removeAll()), [removeContainerListener](http://docs.google.com/java/awt/Container.html#removeContainerListener(java.awt.event.ContainerListener)), [setComponentZOrder](http://docs.google.com/java/awt/Container.html#setComponentZOrder(java.awt.Component,%20int)), [setFocusCycleRoot](http://docs.google.com/java/awt/Container.html#setFocusCycleRoot(boolean)), [setFocusTraversalPolicy](http://docs.google.com/java/awt/Container.html#setFocusTraversalPolicy(java.awt.FocusTraversalPolicy)), [setFocusTraversalPolicyProvider](http://docs.google.com/java/awt/Container.html#setFocusTraversalPolicyProvider(boolean)), [setLayout](http://docs.google.com/java/awt/Container.html#setLayout(java.awt.LayoutManager)), [transferFocusBackward](http://docs.google.com/java/awt/Container.html#transferFocusBackward()), [transferFocusDownCycle](http://docs.google.com/java/awt/Container.html#transferFocusDownCycle()), [validate](http://docs.google.com/java/awt/Container.html#validate()), [validateTree](http://docs.google.com/java/awt/Container.html#validateTree()) |

| **Methods inherited from class java.awt.**[**Component**](http://docs.google.com/java/awt/Component.html) |
| --- |
| [action](http://docs.google.com/java/awt/Component.html#action(java.awt.Event,%20java.lang.Object)), [add](http://docs.google.com/java/awt/Component.html#add(java.awt.PopupMenu)), [addComponentListener](http://docs.google.com/java/awt/Component.html#addComponentListener(java.awt.event.ComponentListener)), [addFocusListener](http://docs.google.com/java/awt/Component.html#addFocusListener(java.awt.event.FocusListener)), [addHierarchyBoundsListener](http://docs.google.com/java/awt/Component.html#addHierarchyBoundsListener(java.awt.event.HierarchyBoundsListener)), [addHierarchyListener](http://docs.google.com/java/awt/Component.html#addHierarchyListener(java.awt.event.HierarchyListener)), [addInputMethodListener](http://docs.google.com/java/awt/Component.html#addInputMethodListener(java.awt.event.InputMethodListener)), [addKeyListener](http://docs.google.com/java/awt/Component.html#addKeyListener(java.awt.event.KeyListener)), [addMouseListener](http://docs.google.com/java/awt/Component.html#addMouseListener(java.awt.event.MouseListener)), [addMouseMotionListener](http://docs.google.com/java/awt/Component.html#addMouseMotionListener(java.awt.event.MouseMotionListener)), [addMouseWheelListener](http://docs.google.com/java/awt/Component.html#addMouseWheelListener(java.awt.event.MouseWheelListener)), [bounds](http://docs.google.com/java/awt/Component.html#bounds()), [checkImage](http://docs.google.com/java/awt/Component.html#checkImage(java.awt.Image,%20java.awt.image.ImageObserver)), [checkImage](http://docs.google.com/java/awt/Component.html#checkImage(java.awt.Image,%20int,%20int,%20java.awt.image.ImageObserver)), [coalesceEvents](http://docs.google.com/java/awt/Component.html#coalesceEvents(java.awt.AWTEvent,%20java.awt.AWTEvent)), [contains](http://docs.google.com/java/awt/Component.html#contains(java.awt.Point)), [createImage](http://docs.google.com/java/awt/Component.html#createImage(java.awt.image.ImageProducer)), [createImage](http://docs.google.com/java/awt/Component.html#createImage(int,%20int)), [createVolatileImage](http://docs.google.com/java/awt/Component.html#createVolatileImage(int,%20int)), [createVolatileImage](http://docs.google.com/java/awt/Component.html#createVolatileImage(int,%20int,%20java.awt.ImageCapabilities)), [disableEvents](http://docs.google.com/java/awt/Component.html#disableEvents(long)), [dispatchEvent](http://docs.google.com/java/awt/Component.html#dispatchEvent(java.awt.AWTEvent)), [enable](http://docs.google.com/java/awt/Component.html#enable(boolean)), [enableEvents](http://docs.google.com/java/awt/Component.html#enableEvents(long)), [enableInputMethods](http://docs.google.com/java/awt/Component.html#enableInputMethods(boolean)), [firePropertyChange](http://docs.google.com/java/awt/Component.html#firePropertyChange(java.lang.String,%20byte,%20byte)), [firePropertyChange](http://docs.google.com/java/awt/Component.html#firePropertyChange(java.lang.String,%20double,%20double)), [firePropertyChange](http://docs.google.com/java/awt/Component.html#firePropertyChange(java.lang.String,%20float,%20float)), [firePropertyChange](http://docs.google.com/java/awt/Component.html#firePropertyChange(java.lang.String,%20long,%20long)), [firePropertyChange](http://docs.google.com/java/awt/Component.html#firePropertyChange(java.lang.String,%20java.lang.Object,%20java.lang.Object)), [firePropertyChange](http://docs.google.com/java/awt/Component.html#firePropertyChange(java.lang.String,%20short,%20short)), [getBackground](http://docs.google.com/java/awt/Component.html#getBackground()), [getBounds](http://docs.google.com/java/awt/Component.html#getBounds()), [getColorModel](http://docs.google.com/java/awt/Component.html#getColorModel()), [getComponentListeners](http://docs.google.com/java/awt/Component.html#getComponentListeners()), [getComponentOrientation](http://docs.google.com/java/awt/Component.html#getComponentOrientation()), [getCursor](http://docs.google.com/java/awt/Component.html#getCursor()), [getDropTarget](http://docs.google.com/java/awt/Component.html#getDropTarget()), [getFocusCycleRootAncestor](http://docs.google.com/java/awt/Component.html#getFocusCycleRootAncestor()), [getFocusListeners](http://docs.google.com/java/awt/Component.html#getFocusListeners()), [getFocusTraversalKeysEnabled](http://docs.google.com/java/awt/Component.html#getFocusTraversalKeysEnabled()), [getFont](http://docs.google.com/java/awt/Component.html#getFont()), [getForeground](http://docs.google.com/java/awt/Component.html#getForeground()), [getGraphicsConfiguration](http://docs.google.com/java/awt/Component.html#getGraphicsConfiguration()), [getHierarchyBoundsListeners](http://docs.google.com/java/awt/Component.html#getHierarchyBoundsListeners()), [getHierarchyListeners](http://docs.google.com/java/awt/Component.html#getHierarchyListeners()), [getIgnoreRepaint](http://docs.google.com/java/awt/Component.html#getIgnoreRepaint()), [getInputContext](http://docs.google.com/java/awt/Component.html#getInputContext()), [getInputMethodListeners](http://docs.google.com/java/awt/Component.html#getInputMethodListeners()), [getInputMethodRequests](http://docs.google.com/java/awt/Component.html#getInputMethodRequests()), [getKeyListeners](http://docs.google.com/java/awt/Component.html#getKeyListeners()), [getLocale](http://docs.google.com/java/awt/Component.html#getLocale()), [getLocation](http://docs.google.com/java/awt/Component.html#getLocation()), [getLocationOnScreen](http://docs.google.com/java/awt/Component.html#getLocationOnScreen()), [getMouseListeners](http://docs.google.com/java/awt/Component.html#getMouseListeners()), [getMouseMotionListeners](http://docs.google.com/java/awt/Component.html#getMouseMotionListeners()), [getMousePosition](http://docs.google.com/java/awt/Component.html#getMousePosition()), [getMouseWheelListeners](http://docs.google.com/java/awt/Component.html#getMouseWheelListeners()), [getName](http://docs.google.com/java/awt/Component.html#getName()), [getParent](http://docs.google.com/java/awt/Component.html#getParent()), [getPeer](http://docs.google.com/java/awt/Component.html#getPeer()), [getPropertyChangeListeners](http://docs.google.com/java/awt/Component.html#getPropertyChangeListeners()), [getPropertyChangeListeners](http://docs.google.com/java/awt/Component.html#getPropertyChangeListeners(java.lang.String)), [getSize](http://docs.google.com/java/awt/Component.html#getSize()), [getToolkit](http://docs.google.com/java/awt/Component.html#getToolkit()), [getTreeLock](http://docs.google.com/java/awt/Component.html#getTreeLock()), [gotFocus](http://docs.google.com/java/awt/Component.html#gotFocus(java.awt.Event,%20java.lang.Object)), [handleEvent](http://docs.google.com/java/awt/Component.html#handleEvent(java.awt.Event)), [hasFocus](http://docs.google.com/java/awt/Component.html#hasFocus()), [hide](http://docs.google.com/java/awt/Component.html#hide()), [imageUpdate](http://docs.google.com/java/awt/Component.html#imageUpdate(java.awt.Image,%20int,%20int,%20int,%20int,%20int)), [inside](http://docs.google.com/java/awt/Component.html#inside(int,%20int)), [isBackgroundSet](http://docs.google.com/java/awt/Component.html#isBackgroundSet()), [isCursorSet](http://docs.google.com/java/awt/Component.html#isCursorSet()), [isDisplayable](http://docs.google.com/java/awt/Component.html#isDisplayable()), [isEnabled](http://docs.google.com/java/awt/Component.html#isEnabled()), [isFocusable](http://docs.google.com/java/awt/Component.html#isFocusable()), [isFocusOwner](http://docs.google.com/java/awt/Component.html#isFocusOwner()), [isFocusTraversable](http://docs.google.com/java/awt/Component.html#isFocusTraversable()), [isFontSet](http://docs.google.com/java/awt/Component.html#isFontSet()), [isForegroundSet](http://docs.google.com/java/awt/Component.html#isForegroundSet()), [isLightweight](http://docs.google.com/java/awt/Component.html#isLightweight()), [isMaximumSizeSet](http://docs.google.com/java/awt/Component.html#isMaximumSizeSet()), [isMinimumSizeSet](http://docs.google.com/java/awt/Component.html#isMinimumSizeSet()), [isPreferredSizeSet](http://docs.google.com/java/awt/Component.html#isPreferredSizeSet()), [isShowing](http://docs.google.com/java/awt/Component.html#isShowing()), [isValid](http://docs.google.com/java/awt/Component.html#isValid()), [isVisible](http://docs.google.com/java/awt/Component.html#isVisible()), [keyDown](http://docs.google.com/java/awt/Component.html#keyDown(java.awt.Event,%20int)), [keyUp](http://docs.google.com/java/awt/Component.html#keyUp(java.awt.Event,%20int)), [list](http://docs.google.com/java/awt/Component.html#list()), [list](http://docs.google.com/java/awt/Component.html#list(java.io.PrintStream)), [list](http://docs.google.com/java/awt/Component.html#list(java.io.PrintWriter)), [location](http://docs.google.com/java/awt/Component.html#location()), [lostFocus](http://docs.google.com/java/awt/Component.html#lostFocus(java.awt.Event,%20java.lang.Object)), [mouseDown](http://docs.google.com/java/awt/Component.html#mouseDown(java.awt.Event,%20int,%20int)), [mouseDrag](http://docs.google.com/java/awt/Component.html#mouseDrag(java.awt.Event,%20int,%20int)), [mouseEnter](http://docs.google.com/java/awt/Component.html#mouseEnter(java.awt.Event,%20int,%20int)), [mouseExit](http://docs.google.com/java/awt/Component.html#mouseExit(java.awt.Event,%20int,%20int)), [mouseMove](http://docs.google.com/java/awt/Component.html#mouseMove(java.awt.Event,%20int,%20int)), [mouseUp](http://docs.google.com/java/awt/Component.html#mouseUp(java.awt.Event,%20int,%20int)), [move](http://docs.google.com/java/awt/Component.html#move(int,%20int)), [nextFocus](http://docs.google.com/java/awt/Component.html#nextFocus()), [paintAll](http://docs.google.com/java/awt/Component.html#paintAll(java.awt.Graphics)), [postEvent](http://docs.google.com/java/awt/Component.html#postEvent(java.awt.Event)), [prepareImage](http://docs.google.com/java/awt/Component.html#prepareImage(java.awt.Image,%20java.awt.image.ImageObserver)), [prepareImage](http://docs.google.com/java/awt/Component.html#prepareImage(java.awt.Image,%20int,%20int,%20java.awt.image.ImageObserver)), [processComponentEvent](http://docs.google.com/java/awt/Component.html#processComponentEvent(java.awt.event.ComponentEvent)), [processFocusEvent](http://docs.google.com/java/awt/Component.html#processFocusEvent(java.awt.event.FocusEvent)), [processHierarchyBoundsEvent](http://docs.google.com/java/awt/Component.html#processHierarchyBoundsEvent(java.awt.event.HierarchyEvent)), [processHierarchyEvent](http://docs.google.com/java/awt/Component.html#processHierarchyEvent(java.awt.event.HierarchyEvent)), [processInputMethodEvent](http://docs.google.com/java/awt/Component.html#processInputMethodEvent(java.awt.event.InputMethodEvent)), [processMouseWheelEvent](http://docs.google.com/java/awt/Component.html#processMouseWheelEvent(java.awt.event.MouseWheelEvent)), [remove](http://docs.google.com/java/awt/Component.html#remove(java.awt.MenuComponent)), [removeComponentListener](http://docs.google.com/java/awt/Component.html#removeComponentListener(java.awt.event.ComponentListener)), [removeFocusListener](http://docs.google.com/java/awt/Component.html#removeFocusListener(java.awt.event.FocusListener)), [removeHierarchyBoundsListener](http://docs.google.com/java/awt/Component.html#removeHierarchyBoundsListener(java.awt.event.HierarchyBoundsListener)), [removeHierarchyListener](http://docs.google.com/java/awt/Component.html#removeHierarchyListener(java.awt.event.HierarchyListener)), [removeInputMethodListener](http://docs.google.com/java/awt/Component.html#removeInputMethodListener(java.awt.event.InputMethodListener)), [removeKeyListener](http://docs.google.com/java/awt/Component.html#removeKeyListener(java.awt.event.KeyListener)), [removeMouseListener](http://docs.google.com/java/awt/Component.html#removeMouseListener(java.awt.event.MouseListener)), [removeMouseMotionListener](http://docs.google.com/java/awt/Component.html#removeMouseMotionListener(java.awt.event.MouseMotionListener)), [removeMouseWheelListener](http://docs.google.com/java/awt/Component.html#removeMouseWheelListener(java.awt.event.MouseWheelListener)), [removePropertyChangeListener](http://docs.google.com/java/awt/Component.html#removePropertyChangeListener(java.beans.PropertyChangeListener)), [removePropertyChangeListener](http://docs.google.com/java/awt/Component.html#removePropertyChangeListener(java.lang.String,%20java.beans.PropertyChangeListener)), [repaint](http://docs.google.com/java/awt/Component.html#repaint()), [repaint](http://docs.google.com/java/awt/Component.html#repaint(int,%20int,%20int,%20int)), [repaint](http://docs.google.com/java/awt/Component.html#repaint(long)), [resize](http://docs.google.com/java/awt/Component.html#resize(java.awt.Dimension)), [resize](http://docs.google.com/java/awt/Component.html#resize(int,%20int)), [setBounds](http://docs.google.com/java/awt/Component.html#setBounds(int,%20int,%20int,%20int)), [setBounds](http://docs.google.com/java/awt/Component.html#setBounds(java.awt.Rectangle)), [setComponentOrientation](http://docs.google.com/java/awt/Component.html#setComponentOrientation(java.awt.ComponentOrientation)), [setCursor](http://docs.google.com/java/awt/Component.html#setCursor(java.awt.Cursor)), [setDropTarget](http://docs.google.com/java/awt/Component.html#setDropTarget(java.awt.dnd.DropTarget)), [setFocusable](http://docs.google.com/java/awt/Component.html#setFocusable(boolean)), [setFocusTraversalKeysEnabled](http://docs.google.com/java/awt/Component.html#setFocusTraversalKeysEnabled(boolean)), [setIgnoreRepaint](http://docs.google.com/java/awt/Component.html#setIgnoreRepaint(boolean)), [setLocale](http://docs.google.com/java/awt/Component.html#setLocale(java.util.Locale)), [setLocation](http://docs.google.com/java/awt/Component.html#setLocation(int,%20int)), [setLocation](http://docs.google.com/java/awt/Component.html#setLocation(java.awt.Point)), [setName](http://docs.google.com/java/awt/Component.html#setName(java.lang.String)), [setSize](http://docs.google.com/java/awt/Component.html#setSize(java.awt.Dimension)), [setSize](http://docs.google.com/java/awt/Component.html#setSize(int,%20int)), [show](http://docs.google.com/java/awt/Component.html#show()), [show](http://docs.google.com/java/awt/Component.html#show(boolean)), [size](http://docs.google.com/java/awt/Component.html#size()), [toString](http://docs.google.com/java/awt/Component.html#toString()), [transferFocus](http://docs.google.com/java/awt/Component.html#transferFocus()), [transferFocusUpCycle](http://docs.google.com/java/awt/Component.html#transferFocusUpCycle()) |

| **Methods inherited from class java.lang.**[**Object**](http://docs.google.com/java/lang/Object.html) |
| --- |
| [clone](http://docs.google.com/java/lang/Object.html#clone()), [equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [finalize](http://docs.google.com/java/lang/Object.html#finalize()), [getClass](http://docs.google.com/java/lang/Object.html#getClass()), [hashCode](http://docs.google.com/java/lang/Object.html#hashCode()), [notify](http://docs.google.com/java/lang/Object.html#notify()), [notifyAll](http://docs.google.com/java/lang/Object.html#notifyAll()), [wait](http://docs.google.com/java/lang/Object.html#wait()), [wait](http://docs.google.com/java/lang/Object.html#wait(long)), [wait](http://docs.google.com/java/lang/Object.html#wait(long,%20int)) |

| **Field Detail** |
| --- |

### ui

protected transient [ComponentUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html) **ui**

The look and feel delegate for this component.

### listenerList

protected [EventListenerList](http://docs.google.com/javax/swing/event/EventListenerList.html) **listenerList**

A list of event listeners for this component.

### WHEN\_FOCUSED

public static final int **WHEN\_FOCUSED**

Constant used for registerKeyboardAction that means that the command should be invoked when the component has the focus.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#javax.swing.JComponent.WHEN_FOCUSED)

### WHEN\_ANCESTOR\_OF\_FOCUSED\_COMPONENT

public static final int **WHEN\_ANCESTOR\_OF\_FOCUSED\_COMPONENT**

Constant used for registerKeyboardAction that means that the command should be invoked when the receiving component is an ancestor of the focused component or is itself the focused component.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#javax.swing.JComponent.WHEN_ANCESTOR_OF_FOCUSED_COMPONENT)

### WHEN\_IN\_FOCUSED\_WINDOW

public static final int **WHEN\_IN\_FOCUSED\_WINDOW**

Constant used for registerKeyboardAction that means that the command should be invoked when the receiving component is in the window that has the focus or is itself the focused component.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#javax.swing.JComponent.WHEN_IN_FOCUSED_WINDOW)

### UNDEFINED\_CONDITION

public static final int **UNDEFINED\_CONDITION**

Constant used by some of the APIs to mean that no condition is defined.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#javax.swing.JComponent.UNDEFINED_CONDITION)

### TOOL\_TIP\_TEXT\_KEY

public static final [String](http://docs.google.com/java/lang/String.html) **TOOL\_TIP\_TEXT\_KEY**

The comment to display when the cursor is over the component, also known as a "value tip", "flyover help", or "flyover label".

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#javax.swing.JComponent.TOOL_TIP_TEXT_KEY)

### accessibleContext

protected [AccessibleContext](http://docs.google.com/javax/accessibility/AccessibleContext.html) **accessibleContext**

The AccessibleContext associated with this JComponent.

| **Constructor Detail** |
| --- |

### JComponent

public **JComponent**()

Default JComponent constructor. This constructor does very little initialization beyond calling the Container constructor. For example, the initial layout manager is null. It does, however, set the component's locale property to the value returned by JComponent.getDefaultLocale.

**See Also:**[getDefaultLocale()](http://docs.google.com/javax/swing/JComponent.html#getDefaultLocale())

| **Method Detail** |
| --- |

### setInheritsPopupMenu

public void **setInheritsPopupMenu**(boolean value)

Sets whether or not getComponentPopupMenu should delegate to the parent if this component does not have a JPopupMenu assigned to it.

The default value for this is false, but some JComponent subclasses that are implemented as a number of JComponents may set this to true.

This is a bound property.

**Parameters:**value - whether or not the JPopupMenu is inherited**Since:** 1.5 **See Also:**[setComponentPopupMenu(javax.swing.JPopupMenu)](http://docs.google.com/javax/swing/JComponent.html#setComponentPopupMenu(javax.swing.JPopupMenu))

### getInheritsPopupMenu

public boolean **getInheritsPopupMenu**()

Returns true if the JPopupMenu should be inherited from the parent.

**Since:** 1.5 **See Also:**[setComponentPopupMenu(javax.swing.JPopupMenu)](http://docs.google.com/javax/swing/JComponent.html#setComponentPopupMenu(javax.swing.JPopupMenu))

### setComponentPopupMenu

public void **setComponentPopupMenu**([JPopupMenu](http://docs.google.com/javax/swing/JPopupMenu.html) popup)

Sets the JPopupMenu for this JComponent. The UI is responsible for registering bindings and adding the necessary listeners such that the JPopupMenu will be shown at the appropriate time. When the JPopupMenu is shown depends upon the look and feel: some may show it on a mouse event, some may enable a key binding.

If popup is null, and getInheritsPopupMenu returns true, then getComponentPopupMenu will be delegated to the parent. This provides for a way to make all child components inherit the popupmenu of the parent.

This is a bound property.

**Parameters:**popup - - the popup that will be assigned to this component may be null**Since:** 1.5 **See Also:**[getComponentPopupMenu()](http://docs.google.com/javax/swing/JComponent.html#getComponentPopupMenu())

### getComponentPopupMenu

public [JPopupMenu](http://docs.google.com/javax/swing/JPopupMenu.html) **getComponentPopupMenu**()

Returns JPopupMenu that assigned for this component. If this component does not have a JPopupMenu assigned to it and getInheritsPopupMenu is true, this will return getParent().getComponentPopupMenu() (assuming the parent is valid.)

**Returns:**JPopupMenu assigned for this component or null if no popup assigned**Since:** 1.5 **See Also:**[setComponentPopupMenu(javax.swing.JPopupMenu)](http://docs.google.com/javax/swing/JComponent.html#setComponentPopupMenu(javax.swing.JPopupMenu))

### updateUI

public void **updateUI**()

Resets the UI property to a value from the current look and feel. JComponent subclasses must override this method like this:

public void updateUI() {  
 setUI((SliderUI)UIManager.getUI(this);  
 }

**See Also:**[setUI(javax.swing.plaf.ComponentUI)](http://docs.google.com/javax/swing/JComponent.html#setUI(javax.swing.plaf.ComponentUI)), [UIManager.getLookAndFeel()](http://docs.google.com/javax/swing/UIManager.html#getLookAndFeel()), [UIManager.getUI(javax.swing.JComponent)](http://docs.google.com/javax/swing/UIManager.html#getUI(javax.swing.JComponent))

### setUI

protected void **setUI**([ComponentUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html) newUI)

Sets the look and feel delegate for this component. JComponent subclasses generally override this method to narrow the argument type. For example, in JSlider:

public void setUI(SliderUI newUI) {  
 super.setUI(newUI);  
 }

Additionally JComponent subclasses must provide a getUI method that returns the correct type. For example:

public SliderUI getUI() {  
 return (SliderUI)ui;  
 }

**Parameters:**newUI - the new UI delegate**See Also:**[updateUI()](http://docs.google.com/javax/swing/JComponent.html#updateUI()), [UIManager.getLookAndFeel()](http://docs.google.com/javax/swing/UIManager.html#getLookAndFeel()), [UIManager.getUI(javax.swing.JComponent)](http://docs.google.com/javax/swing/UIManager.html#getUI(javax.swing.JComponent))

### getUIClassID

public [String](http://docs.google.com/java/lang/String.html) **getUIClassID**()

Returns the UIDefaults key used to look up the name of the swing.plaf.ComponentUI class that defines the look and feel for this component. Most applications will never need to call this method. Subclasses of JComponent that support pluggable look and feel should override this method to return a UIDefaults key that maps to the ComponentUI subclass that defines their look and feel.

**Returns:**the UIDefaults key for a ComponentUI subclass**See Also:**[UIDefaults.getUI(javax.swing.JComponent)](http://docs.google.com/javax/swing/UIDefaults.html#getUI(javax.swing.JComponent))

### getComponentGraphics

protected [Graphics](http://docs.google.com/java/awt/Graphics.html) **getComponentGraphics**([Graphics](http://docs.google.com/java/awt/Graphics.html) g)

Returns the graphics object used to paint this component. If DebugGraphics is turned on we create a new DebugGraphics object if necessary. Otherwise we just configure the specified graphics object's foreground and font.

**Parameters:**g - the original Graphics object **Returns:**a Graphics object configured for this component

### paintComponent

protected void **paintComponent**([Graphics](http://docs.google.com/java/awt/Graphics.html) g)

Calls the UI delegate's paint method, if the UI delegate is non-null. We pass the delegate a copy of the Graphics object to protect the rest of the paint code from irrevocable changes (for example, Graphics.translate).

If you override this in a subclass you should not make permanent changes to the passed in Graphics. For example, you should not alter the clip Rectangle or modify the transform. If you need to do these operations you may find it easier to create a new Graphics from the passed in Graphics and manipulate it. Further, if you do not invoker super's implementation you must honor the opaque property, that is if this component is opaque, you must completely fill in the background in a non-opaque color. If you do not honor the opaque property you will likely see visual artifacts.

The passed in Graphics object might have a transform other than the identify transform installed on it. In this case, you might get unexpected results if you cumulatively apply another transform.

**Parameters:**g - the Graphics object to protect**See Also:**[paint(java.awt.Graphics)](http://docs.google.com/javax/swing/JComponent.html#paint(java.awt.Graphics)), [ComponentUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html)

### paintChildren

protected void **paintChildren**([Graphics](http://docs.google.com/java/awt/Graphics.html) g)

Paints this component's children. If shouldUseBuffer is true, no component ancestor has a buffer and the component children can use a buffer if they have one. Otherwise, one ancestor has a buffer currently in use and children should not use a buffer to paint.

**Parameters:**g - the Graphics context in which to paint**See Also:**[paint(java.awt.Graphics)](http://docs.google.com/javax/swing/JComponent.html#paint(java.awt.Graphics)), [Container.paint(java.awt.Graphics)](http://docs.google.com/java/awt/Container.html#paint(java.awt.Graphics))

### paintBorder

protected void **paintBorder**([Graphics](http://docs.google.com/java/awt/Graphics.html) g)

Paints the component's border.

If you override this in a subclass you should not make permanent changes to the passed in Graphics. For example, you should not alter the clip Rectangle or modify the transform. If you need to do these operations you may find it easier to create a new Graphics from the passed in Graphics and manipulate it.

**Parameters:**g - the Graphics context in which to paint**See Also:**[paint(java.awt.Graphics)](http://docs.google.com/javax/swing/JComponent.html#paint(java.awt.Graphics)), [setBorder(javax.swing.border.Border)](http://docs.google.com/javax/swing/JComponent.html#setBorder(javax.swing.border.Border))

### update

public void **update**([Graphics](http://docs.google.com/java/awt/Graphics.html) g)

Calls paint. Doesn't clear the background but see ComponentUI.update, which is called by paintComponent.

**Overrides:**[update](http://docs.google.com/java/awt/Container.html#update(java.awt.Graphics)) in class [Container](http://docs.google.com/java/awt/Container.html) **Parameters:**g - the Graphics context in which to paint**See Also:**[paint(java.awt.Graphics)](http://docs.google.com/javax/swing/JComponent.html#paint(java.awt.Graphics)), [paintComponent(java.awt.Graphics)](http://docs.google.com/javax/swing/JComponent.html#paintComponent(java.awt.Graphics)), [ComponentUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html)

### paint

public void **paint**([Graphics](http://docs.google.com/java/awt/Graphics.html) g)

Invoked by Swing to draw components. Applications should not invoke paint directly, but should instead use the repaint method to schedule the component for redrawing.

This method actually delegates the work of painting to three protected methods: paintComponent, paintBorder, and paintChildren. They're called in the order listed to ensure that children appear on top of component itself. Generally speaking, the component and its children should not paint in the insets area allocated to the border. Subclasses can just override this method, as always. A subclass that just wants to specialize the UI (look and feel) delegate's paint method should just override paintComponent.

**Overrides:**[paint](http://docs.google.com/java/awt/Container.html#paint(java.awt.Graphics)) in class [Container](http://docs.google.com/java/awt/Container.html) **Parameters:**g - the Graphics context in which to paint**See Also:**[paintComponent(java.awt.Graphics)](http://docs.google.com/javax/swing/JComponent.html#paintComponent(java.awt.Graphics)), [paintBorder(java.awt.Graphics)](http://docs.google.com/javax/swing/JComponent.html#paintBorder(java.awt.Graphics)), [paintChildren(java.awt.Graphics)](http://docs.google.com/javax/swing/JComponent.html#paintChildren(java.awt.Graphics)), [getComponentGraphics(java.awt.Graphics)](http://docs.google.com/javax/swing/JComponent.html#getComponentGraphics(java.awt.Graphics)), [repaint(long, int, int, int, int)](http://docs.google.com/javax/swing/JComponent.html#repaint(long,%20int,%20int,%20int,%20int))

### printAll

public void **printAll**([Graphics](http://docs.google.com/java/awt/Graphics.html) g)

Invoke this method to print the component. This method invokes print on the component.

**Overrides:**[printAll](http://docs.google.com/java/awt/Component.html#printAll(java.awt.Graphics)) in class [Component](http://docs.google.com/java/awt/Component.html) **Parameters:**g - the Graphics context in which to paint**See Also:**[print(java.awt.Graphics)](http://docs.google.com/javax/swing/JComponent.html#print(java.awt.Graphics)), [printComponent(java.awt.Graphics)](http://docs.google.com/javax/swing/JComponent.html#printComponent(java.awt.Graphics)), [printBorder(java.awt.Graphics)](http://docs.google.com/javax/swing/JComponent.html#printBorder(java.awt.Graphics)), [printChildren(java.awt.Graphics)](http://docs.google.com/javax/swing/JComponent.html#printChildren(java.awt.Graphics))

### print

public void **print**([Graphics](http://docs.google.com/java/awt/Graphics.html) g)

Invoke this method to print the component to the specified Graphics. This method will result in invocations of printComponent, printBorder and printChildren. It is recommended that you override one of the previously mentioned methods rather than this one if your intention is to customize the way printing looks. However, it can be useful to override this method should you want to prepare state before invoking the superclass behavior. As an example, if you wanted to change the component's background color before printing, you could do the following:

public void print(Graphics g) {  
 Color orig = getBackground();  
 setBackground(Color.WHITE);  
  
 // wrap in try/finally so that we always restore the state  
 try {  
 super.print(g);  
 } finally {  
 setBackground(orig);  
 }  
 }

Alternatively, or for components that delegate painting to other objects, you can query during painting whether or not the component is in the midst of a print operation. The isPaintingForPrint method provides this ability and its return value will be changed by this method: to true immediately before rendering and to false immediately after. With each change a property change event is fired on this component with the name "paintingForPrint".

This method sets the component's state such that the double buffer will not be used: painting will be done directly on the passed in Graphics.

**Overrides:**[print](http://docs.google.com/java/awt/Container.html#print(java.awt.Graphics)) in class [Container](http://docs.google.com/java/awt/Container.html) **Parameters:**g - the Graphics context in which to paint**See Also:**[printComponent(java.awt.Graphics)](http://docs.google.com/javax/swing/JComponent.html#printComponent(java.awt.Graphics)), [printBorder(java.awt.Graphics)](http://docs.google.com/javax/swing/JComponent.html#printBorder(java.awt.Graphics)), [printChildren(java.awt.Graphics)](http://docs.google.com/javax/swing/JComponent.html#printChildren(java.awt.Graphics)), [isPaintingForPrint()](http://docs.google.com/javax/swing/JComponent.html#isPaintingForPrint())

### printComponent

protected void **printComponent**([Graphics](http://docs.google.com/java/awt/Graphics.html) g)

This is invoked during a printing operation. This is implemented to invoke paintComponent on the component. Override this if you wish to add special painting behavior when printing.

**Parameters:**g - the Graphics context in which to paint**Since:** 1.3 **See Also:**[print(java.awt.Graphics)](http://docs.google.com/javax/swing/JComponent.html#print(java.awt.Graphics))

### printChildren

protected void **printChildren**([Graphics](http://docs.google.com/java/awt/Graphics.html) g)

Prints this component's children. This is implemented to invoke paintChildren on the component. Override this if you wish to print the children differently than painting.

**Parameters:**g - the Graphics context in which to paint**Since:** 1.3 **See Also:**[print(java.awt.Graphics)](http://docs.google.com/javax/swing/JComponent.html#print(java.awt.Graphics))

### printBorder

protected void **printBorder**([Graphics](http://docs.google.com/java/awt/Graphics.html) g)

Prints the component's border. This is implemented to invoke paintBorder on the component. Override this if you wish to print the border differently that it is painted.

**Parameters:**g - the Graphics context in which to paint**Since:** 1.3 **See Also:**[print(java.awt.Graphics)](http://docs.google.com/javax/swing/JComponent.html#print(java.awt.Graphics))

### isPaintingTile

public boolean **isPaintingTile**()

Returns true if the component is currently painting a tile. If this method returns true, paint will be called again for another tile. This method returns false if you are not painting a tile or if the last tile is painted. Use this method to keep some state you might need between tiles.

**Returns:**true if the component is currently painting a tile, false otherwise

### isPaintingForPrint

public final boolean **isPaintingForPrint**()

Returns true if the current painting operation on this component is part of a print operation. This method is useful when you want to customize what you print versus what you show on the screen.

You can detect changes in the value of this property by listening for property change events on this component with name "paintingForPrint".

Note: This method provides complimentary functionality to that provided by other high level Swing printing APIs. However, it deals strictly with painting and should not be confused as providing information on higher level print processes. For example, a [JTable.print()](http://docs.google.com/javax/swing/JTable.html#print()) operation doesn't necessarily result in a continuous rendering of the full component, and the return value of this method can change multiple times during that operation. It is even possible for the component to be painted to the screen while the printing process is ongoing. In such a case, the return value of this method is true when, and only when, the table is being painted as part of the printing process.

**Returns:**true if the current painting operation on this component is part of a print operation**Since:** 1.6 **See Also:**[print(java.awt.Graphics)](http://docs.google.com/javax/swing/JComponent.html#print(java.awt.Graphics))

### isManagingFocus

[@Deprecated](http://docs.google.com/java/lang/Deprecated.html)  
public boolean **isManagingFocus**()

**Deprecated.** *As of 1.4, replaced by Component.setFocusTraversalKeys(int, Set) and Container.setFocusCycleRoot(boolean).*

In release 1.4, the focus subsystem was rearchitected. For more information, see  [How to Use the Focus Subsystem](http://java.sun.com/docs/books/tutorial/uiswing/misc/focus.html), a section in *The Java Tutorial*.

Changes this JComponent's focus traversal keys to CTRL+TAB and CTRL+SHIFT+TAB. Also prevents SortingFocusTraversalPolicy from considering descendants of this JComponent when computing a focus traversal cycle.

**See Also:**[Component.setFocusTraversalKeys(int, java.util.Set)](http://docs.google.com/java/awt/Component.html#setFocusTraversalKeys(int,%20java.util.Set)), [SortingFocusTraversalPolicy](http://docs.google.com/javax/swing/SortingFocusTraversalPolicy.html)

### setNextFocusableComponent

[@Deprecated](http://docs.google.com/java/lang/Deprecated.html)  
public void **setNextFocusableComponent**([Component](http://docs.google.com/java/awt/Component.html) aComponent)

**Deprecated.** *As of 1.4, replaced by FocusTraversalPolicy*

In release 1.4, the focus subsystem was rearchitected. For more information, see  [How to Use the Focus Subsystem](http://java.sun.com/docs/books/tutorial/uiswing/misc/focus.html), a section in *The Java Tutorial*.

Overrides the default FocusTraversalPolicy for this JComponent's focus traversal cycle by unconditionally setting the specified Component as the next Component in the cycle, and this JComponent as the specified Component's previous Component in the cycle.

**Parameters:**aComponent - the Component that should follow this JComponent in the focus traversal cycle**See Also:**[getNextFocusableComponent()](http://docs.google.com/javax/swing/JComponent.html#getNextFocusableComponent()), [FocusTraversalPolicy](http://docs.google.com/java/awt/FocusTraversalPolicy.html)

### getNextFocusableComponent

[@Deprecated](http://docs.google.com/java/lang/Deprecated.html)  
public [Component](http://docs.google.com/java/awt/Component.html) **getNextFocusableComponent**()

**Deprecated.** *As of 1.4, replaced by FocusTraversalPolicy.*

In release 1.4, the focus subsystem was rearchitected. For more information, see  [How to Use the Focus Subsystem](http://java.sun.com/docs/books/tutorial/uiswing/misc/focus.html), a section in *The Java Tutorial*.

Returns the Component set by a prior call to setNextFocusableComponent(Component) on this JComponent.

**Returns:**the Component that will follow this JComponent in the focus traversal cycle, or null if none has been explicitly specified**See Also:**[setNextFocusableComponent(java.awt.Component)](http://docs.google.com/javax/swing/JComponent.html#setNextFocusableComponent(java.awt.Component))

### setRequestFocusEnabled

public void **setRequestFocusEnabled**(boolean requestFocusEnabled)

Provides a hint as to whether or not this JComponent should get focus. This is only a hint, and it is up to consumers that are requesting focus to honor this property. This is typically honored for mouse operations, but not keyboard operations. For example, look and feels could verify this property is true before requesting focus during a mouse operation. This would often times be used if you did not want a mouse press on a JComponent to steal focus, but did want the JComponent to be traversable via the keyboard. If you do not want this JComponent focusable at all, use the setFocusable method instead.

Please see  [How to Use the Focus Subsystem](http://java.sun.com/docs/books/tutorial/uiswing/misc/focus.html), a section in *The Java Tutorial*, for more information.

**Parameters:**requestFocusEnabled - indicates whether you want this JComponent to be focusable or not**See Also:**[Focus Specification](http://docs.google.com/java/awt/doc-files/FocusSpec.html), [Component.setFocusable(boolean)](http://docs.google.com/java/awt/Component.html#setFocusable(boolean))

### isRequestFocusEnabled

public boolean **isRequestFocusEnabled**()

Returns true if this JComponent should get focus; otherwise returns false.

Please see  [How to Use the Focus Subsystem](http://java.sun.com/docs/books/tutorial/uiswing/misc/focus.html), a section in *The Java Tutorial*, for more information.

**Returns:**true if this component should get focus, otherwise returns false**See Also:**[setRequestFocusEnabled(boolean)](http://docs.google.com/javax/swing/JComponent.html#setRequestFocusEnabled(boolean)), [Focus Specification](http://docs.google.com/java/awt/doc-files/FocusSpec.html), [Component.isFocusable()](http://docs.google.com/java/awt/Component.html#isFocusable())

### requestFocus

public void **requestFocus**()

Requests that this Component gets the input focus. Refer to [Component.requestFocus()](http://docs.google.com/java/awt/Component.html#requestFocus()) for a complete description of this method.

Note that the use of this method is discouraged because its behavior is platform dependent. Instead we recommend the use of [requestFocusInWindow()](http://docs.google.com/javax/swing/JComponent.html#requestFocusInWindow()). If you would like more information on focus, see  [How to Use the Focus Subsystem](http://java.sun.com/docs/books/tutorial/uiswing/misc/focus.html), a section in *The Java Tutorial*.

**Overrides:**[requestFocus](http://docs.google.com/java/awt/Component.html#requestFocus()) in class [Component](http://docs.google.com/java/awt/Component.html) **Since:** 1.4 **See Also:**[Component.requestFocusInWindow()](http://docs.google.com/java/awt/Component.html#requestFocusInWindow()), [Component.requestFocusInWindow(boolean)](http://docs.google.com/java/awt/Component.html#requestFocusInWindow(boolean))

### requestFocus

public boolean **requestFocus**(boolean temporary)

Requests that this Component gets the input focus. Refer to [Component.requestFocus(boolean)](http://docs.google.com/java/awt/Component.html#requestFocus(boolean)) for a complete description of this method.

Note that the use of this method is discouraged because its behavior is platform dependent. Instead we recommend the use of [requestFocusInWindow(boolean)](http://docs.google.com/javax/swing/JComponent.html#requestFocusInWindow(boolean)). If you would like more information on focus, see  [How to Use the Focus Subsystem](http://java.sun.com/docs/books/tutorial/uiswing/misc/focus.html), a section in *The Java Tutorial*.

**Overrides:**[requestFocus](http://docs.google.com/java/awt/Component.html#requestFocus(boolean)) in class [Component](http://docs.google.com/java/awt/Component.html) **Parameters:**temporary - boolean indicating if the focus change is temporary **Returns:**false if the focus change request is guaranteed to fail; true if it is likely to succeed**Since:** 1.4 **See Also:**[Component.requestFocusInWindow()](http://docs.google.com/java/awt/Component.html#requestFocusInWindow()), [Component.requestFocusInWindow(boolean)](http://docs.google.com/java/awt/Component.html#requestFocusInWindow(boolean))

### requestFocusInWindow

public boolean **requestFocusInWindow**()

Requests that this Component gets the input focus. Refer to [Component.requestFocusInWindow()](http://docs.google.com/java/awt/Component.html#requestFocusInWindow()) for a complete description of this method.

If you would like more information on focus, see  [How to Use the Focus Subsystem](http://java.sun.com/docs/books/tutorial/uiswing/misc/focus.html), a section in *The Java Tutorial*.

**Overrides:**[requestFocusInWindow](http://docs.google.com/java/awt/Component.html#requestFocusInWindow()) in class [Component](http://docs.google.com/java/awt/Component.html) **Returns:**false if the focus change request is guaranteed to fail; true if it is likely to succeed**Since:** 1.4 **See Also:**[Component.requestFocusInWindow()](http://docs.google.com/java/awt/Component.html#requestFocusInWindow()), [Component.requestFocusInWindow(boolean)](http://docs.google.com/java/awt/Component.html#requestFocusInWindow(boolean))

### requestFocusInWindow

protected boolean **requestFocusInWindow**(boolean temporary)

Requests that this Component gets the input focus. Refer to [Component.requestFocusInWindow(boolean)](http://docs.google.com/java/awt/Component.html#requestFocusInWindow(boolean)) for a complete description of this method.

If you would like more information on focus, see  [How to Use the Focus Subsystem](http://java.sun.com/docs/books/tutorial/uiswing/misc/focus.html), a section in *The Java Tutorial*.

**Overrides:**[requestFocusInWindow](http://docs.google.com/java/awt/Component.html#requestFocusInWindow(boolean)) in class [Component](http://docs.google.com/java/awt/Component.html) **Parameters:**temporary - boolean indicating if the focus change is temporary **Returns:**false if the focus change request is guaranteed to fail; true if it is likely to succeed**Since:** 1.4 **See Also:**[Component.requestFocusInWindow()](http://docs.google.com/java/awt/Component.html#requestFocusInWindow()), [Component.requestFocusInWindow(boolean)](http://docs.google.com/java/awt/Component.html#requestFocusInWindow(boolean))

### grabFocus

public void **grabFocus**()

Requests that this Component get the input focus, and that this Component's top-level ancestor become the focused Window. This component must be displayable, visible, and focusable for the request to be granted.

This method is intended for use by focus implementations. Client code should not use this method; instead, it should use requestFocusInWindow().

**See Also:**[requestFocusInWindow()](http://docs.google.com/javax/swing/JComponent.html#requestFocusInWindow())

### setVerifyInputWhenFocusTarget

public void **setVerifyInputWhenFocusTarget**(boolean verifyInputWhenFocusTarget)

Sets the value to indicate whether input verifier for the current focus owner will be called before this component requests focus. The default is true. Set to false on components such as a Cancel button or a scrollbar, which should activate even if the input in the current focus owner is not "passed" by the input verifier for that component.

**Parameters:**verifyInputWhenFocusTarget - value for the verifyInputWhenFocusTarget property**Since:** 1.3 **See Also:**[InputVerifier](http://docs.google.com/javax/swing/InputVerifier.html), [setInputVerifier(javax.swing.InputVerifier)](http://docs.google.com/javax/swing/JComponent.html#setInputVerifier(javax.swing.InputVerifier)), [getInputVerifier()](http://docs.google.com/javax/swing/JComponent.html#getInputVerifier()), [getVerifyInputWhenFocusTarget()](http://docs.google.com/javax/swing/JComponent.html#getVerifyInputWhenFocusTarget())

### getVerifyInputWhenFocusTarget

public boolean **getVerifyInputWhenFocusTarget**()

Returns the value that indicates whether the input verifier for the current focus owner will be called before this component requests focus.

**Returns:**value of the verifyInputWhenFocusTarget property**Since:** 1.3 **See Also:**[InputVerifier](http://docs.google.com/javax/swing/InputVerifier.html), [setInputVerifier(javax.swing.InputVerifier)](http://docs.google.com/javax/swing/JComponent.html#setInputVerifier(javax.swing.InputVerifier)), [getInputVerifier()](http://docs.google.com/javax/swing/JComponent.html#getInputVerifier()), [setVerifyInputWhenFocusTarget(boolean)](http://docs.google.com/javax/swing/JComponent.html#setVerifyInputWhenFocusTarget(boolean))

### getFontMetrics

public [FontMetrics](http://docs.google.com/java/awt/FontMetrics.html) **getFontMetrics**([Font](http://docs.google.com/java/awt/Font.html) font)

Gets the FontMetrics for the specified Font.

**Overrides:**[getFontMetrics](http://docs.google.com/java/awt/Component.html#getFontMetrics(java.awt.Font)) in class [Component](http://docs.google.com/java/awt/Component.html) **Parameters:**font - the font for which font metrics is to be obtained **Returns:**the font metrics for font **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if font is null**Since:** 1.5 **See Also:**[Component.getFont()](http://docs.google.com/java/awt/Component.html#getFont()), [Component.getPeer()](http://docs.google.com/java/awt/Component.html#getPeer()), ComponentPeer.getFontMetrics(Font), [Toolkit.getFontMetrics(Font)](http://docs.google.com/java/awt/Toolkit.html#getFontMetrics(java.awt.Font))

### setPreferredSize

public void **setPreferredSize**([Dimension](http://docs.google.com/java/awt/Dimension.html) preferredSize)

Sets the preferred size of this component. If preferredSize is null, the UI will be asked for the preferred size.

**Overrides:**[setPreferredSize](http://docs.google.com/java/awt/Component.html#setPreferredSize(java.awt.Dimension)) in class [Component](http://docs.google.com/java/awt/Component.html) **Parameters:**preferredSize - The new preferred size, or null**See Also:**[Component.getPreferredSize()](http://docs.google.com/java/awt/Component.html#getPreferredSize()), [Component.isPreferredSizeSet()](http://docs.google.com/java/awt/Component.html#isPreferredSizeSet())

### getPreferredSize

public [Dimension](http://docs.google.com/java/awt/Dimension.html) **getPreferredSize**()

If the preferredSize has been set to a non-null value just returns it. If the UI delegate's getPreferredSize method returns a non null value then return that; otherwise defer to the component's layout manager.

**Overrides:**[getPreferredSize](http://docs.google.com/java/awt/Container.html#getPreferredSize()) in class [Container](http://docs.google.com/java/awt/Container.html) **Returns:**the value of the preferredSize property**See Also:**[setPreferredSize(java.awt.Dimension)](http://docs.google.com/javax/swing/JComponent.html#setPreferredSize(java.awt.Dimension)), [ComponentUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html)

### setMaximumSize

public void **setMaximumSize**([Dimension](http://docs.google.com/java/awt/Dimension.html) maximumSize)

Sets the maximum size of this component to a constant value. Subsequent calls to getMaximumSize will always return this value; the component's UI will not be asked to compute it. Setting the maximum size to null restores the default behavior.

**Overrides:**[setMaximumSize](http://docs.google.com/java/awt/Component.html#setMaximumSize(java.awt.Dimension)) in class [Component](http://docs.google.com/java/awt/Component.html) **Parameters:**maximumSize - a Dimension containing the desired maximum allowable size**See Also:**[getMaximumSize()](http://docs.google.com/javax/swing/JComponent.html#getMaximumSize())

### getMaximumSize

public [Dimension](http://docs.google.com/java/awt/Dimension.html) **getMaximumSize**()

If the maximum size has been set to a non-null value just returns it. If the UI delegate's getMaximumSize method returns a non-null value then return that; otherwise defer to the component's layout manager.

**Overrides:**[getMaximumSize](http://docs.google.com/java/awt/Container.html#getMaximumSize()) in class [Container](http://docs.google.com/java/awt/Container.html) **Returns:**the value of the maximumSize property**See Also:**[setMaximumSize(java.awt.Dimension)](http://docs.google.com/javax/swing/JComponent.html#setMaximumSize(java.awt.Dimension)), [ComponentUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html)

### setMinimumSize

public void **setMinimumSize**([Dimension](http://docs.google.com/java/awt/Dimension.html) minimumSize)

Sets the minimum size of this component to a constant value. Subsequent calls to getMinimumSize will always return this value; the component's UI will not be asked to compute it. Setting the minimum size to null restores the default behavior.

**Overrides:**[setMinimumSize](http://docs.google.com/java/awt/Component.html#setMinimumSize(java.awt.Dimension)) in class [Component](http://docs.google.com/java/awt/Component.html) **Parameters:**minimumSize - the new minimum size of this component**See Also:**[getMinimumSize()](http://docs.google.com/javax/swing/JComponent.html#getMinimumSize())

### getMinimumSize

public [Dimension](http://docs.google.com/java/awt/Dimension.html) **getMinimumSize**()

If the minimum size has been set to a non-null value just returns it. If the UI delegate's getMinimumSize method returns a non-null value then return that; otherwise defer to the component's layout manager.

**Overrides:**[getMinimumSize](http://docs.google.com/java/awt/Container.html#getMinimumSize()) in class [Container](http://docs.google.com/java/awt/Container.html) **Returns:**the value of the minimumSize property**See Also:**[setMinimumSize(java.awt.Dimension)](http://docs.google.com/javax/swing/JComponent.html#setMinimumSize(java.awt.Dimension)), [ComponentUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html)

### contains

public boolean **contains**(int x,  
 int y)

Gives the UI delegate an opportunity to define the precise shape of this component for the sake of mouse processing.

**Overrides:**[contains](http://docs.google.com/java/awt/Component.html#contains(int,%20int)) in class [Component](http://docs.google.com/java/awt/Component.html) **Parameters:**x - the *x* coordinate of the pointy - the *y* coordinate of the point **Returns:**true if this component logically contains x,y**See Also:**[Component.contains(int, int)](http://docs.google.com/java/awt/Component.html#contains(int,%20int)), [ComponentUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html)

### setBorder

public void **setBorder**([Border](http://docs.google.com/javax/swing/border/Border.html) border)

Sets the border of this component. The Border object is responsible for defining the insets for the component (overriding any insets set directly on the component) and for optionally rendering any border decorations within the bounds of those insets. Borders should be used (rather than insets) for creating both decorative and non-decorative (such as margins and padding) regions for a swing component. Compound borders can be used to nest multiple borders within a single component.

Although technically you can set the border on any object that inherits from JComponent, the look and feel implementation of many standard Swing components doesn't work well with user-set borders. In general, when you want to set a border on a standard Swing component other than JPanel or JLabel, we recommend that you put the component in a JPanel and set the border on the JPanel.

This is a bound property.

**Parameters:**border - the border to be rendered for this component**See Also:**[Border](http://docs.google.com/javax/swing/border/Border.html), [CompoundBorder](http://docs.google.com/javax/swing/border/CompoundBorder.html)

### getBorder

public [Border](http://docs.google.com/javax/swing/border/Border.html) **getBorder**()

Returns the border of this component or null if no border is currently set.

**Returns:**the border object for this component**See Also:**[setBorder(javax.swing.border.Border)](http://docs.google.com/javax/swing/JComponent.html#setBorder(javax.swing.border.Border))

### getInsets

public [Insets](http://docs.google.com/java/awt/Insets.html) **getInsets**()

If a border has been set on this component, returns the border's insets; otherwise calls super.getInsets.

**Overrides:**[getInsets](http://docs.google.com/java/awt/Container.html#getInsets()) in class [Container](http://docs.google.com/java/awt/Container.html) **Returns:**the value of the insets property**See Also:**[setBorder(javax.swing.border.Border)](http://docs.google.com/javax/swing/JComponent.html#setBorder(javax.swing.border.Border))

### getInsets

public [Insets](http://docs.google.com/java/awt/Insets.html) **getInsets**([Insets](http://docs.google.com/java/awt/Insets.html) insets)

Returns an Insets object containing this component's inset values. The passed-in Insets object will be reused if possible. Calling methods cannot assume that the same object will be returned, however. All existing values within this object are overwritten. If insets is null, this will allocate a new one.

**Parameters:**insets - the Insets object, which can be reused **Returns:**the Insets object**See Also:**[getInsets()](http://docs.google.com/javax/swing/JComponent.html#getInsets())

### getAlignmentY

public float **getAlignmentY**()

Overrides Container.getAlignmentY to return the horizontal alignment.

**Overrides:**[getAlignmentY](http://docs.google.com/java/awt/Container.html#getAlignmentY()) in class [Container](http://docs.google.com/java/awt/Container.html) **Returns:**the value of the alignmentY property**See Also:**[setAlignmentY(float)](http://docs.google.com/javax/swing/JComponent.html#setAlignmentY(float)), [Component.getAlignmentY()](http://docs.google.com/java/awt/Component.html#getAlignmentY())

### setAlignmentY

public void **setAlignmentY**(float alignmentY)

Sets the the horizontal alignment.

**Parameters:**alignmentY - the new horizontal alignment**See Also:**[getAlignmentY()](http://docs.google.com/javax/swing/JComponent.html#getAlignmentY())

### getAlignmentX

public float **getAlignmentX**()

Overrides Container.getAlignmentX to return the vertical alignment.

**Overrides:**[getAlignmentX](http://docs.google.com/java/awt/Container.html#getAlignmentX()) in class [Container](http://docs.google.com/java/awt/Container.html) **Returns:**the value of the alignmentX property**See Also:**[setAlignmentX(float)](http://docs.google.com/javax/swing/JComponent.html#setAlignmentX(float)), [Component.getAlignmentX()](http://docs.google.com/java/awt/Component.html#getAlignmentX())

### setAlignmentX

public void **setAlignmentX**(float alignmentX)

Sets the the vertical alignment.

**Parameters:**alignmentX - the new vertical alignment**See Also:**[getAlignmentX()](http://docs.google.com/javax/swing/JComponent.html#getAlignmentX())

### setInputVerifier

public void **setInputVerifier**([InputVerifier](http://docs.google.com/javax/swing/InputVerifier.html) inputVerifier)

Sets the input verifier for this component.

**Parameters:**inputVerifier - the new input verifier**Since:** 1.3 **See Also:**[InputVerifier](http://docs.google.com/javax/swing/InputVerifier.html)

### getInputVerifier

public [InputVerifier](http://docs.google.com/javax/swing/InputVerifier.html) **getInputVerifier**()

Returns the input verifier for this component.

**Returns:**the inputVerifier property**Since:** 1.3 **See Also:**[InputVerifier](http://docs.google.com/javax/swing/InputVerifier.html)

### getGraphics

public [Graphics](http://docs.google.com/java/awt/Graphics.html) **getGraphics**()

Returns this component's graphics context, which lets you draw on a component. Use this method to get a Graphics object and then invoke operations on that object to draw on the component.

**Overrides:**[getGraphics](http://docs.google.com/java/awt/Component.html#getGraphics()) in class [Component](http://docs.google.com/java/awt/Component.html) **Returns:**this components graphics context**See Also:**[Component.paint(java.awt.Graphics)](http://docs.google.com/java/awt/Component.html#paint(java.awt.Graphics))

### setDebugGraphicsOptions

public void **setDebugGraphicsOptions**(int debugOptions)

Enables or disables diagnostic information about every graphics operation performed within the component or one of its children.

**Parameters:**debugOptions - determines how the component should display the information; one of the following options:

* DebugGraphics.LOG\_OPTION - causes a text message to be printed.
* DebugGraphics.FLASH\_OPTION - causes the drawing to flash several times.
* DebugGraphics.BUFFERED\_OPTION - creates an ExternalWindow that displays the operations performed on the View's offscreen buffer.
* DebugGraphics.NONE\_OPTION disables debugging.
* A value of 0 causes no changes to the debugging options.

debugOptions is bitwise OR'd into the current value

### getDebugGraphicsOptions

public int **getDebugGraphicsOptions**()

Returns the state of graphics debugging.

**Returns:**a bitwise OR'd flag of zero or more of the following options:

* DebugGraphics.LOG\_OPTION - causes a text message to be printed.
* DebugGraphics.FLASH\_OPTION - causes the drawing to flash several times.
* DebugGraphics.BUFFERED\_OPTION - creates an ExternalWindow that displays the operations performed on the View's offscreen buffer.
* DebugGraphics.NONE\_OPTION disables debugging.
* A value of 0 causes no changes to the debugging options.

**See Also:**[setDebugGraphicsOptions(int)](http://docs.google.com/javax/swing/JComponent.html#setDebugGraphicsOptions(int))

### registerKeyboardAction

public void **registerKeyboardAction**([ActionListener](http://docs.google.com/java/awt/event/ActionListener.html) anAction,  
 [String](http://docs.google.com/java/lang/String.html) aCommand,  
 [KeyStroke](http://docs.google.com/javax/swing/KeyStroke.html) aKeyStroke,  
 int aCondition)

This method is now obsolete, please use a combination of getActionMap() and getInputMap() for similiar behavior. For example, to bind the KeyStroke aKeyStroke to the Action anAction now use:

component.getInputMap().put(aKeyStroke, aCommand);  
 component.getActionMap().put(aCommmand, anAction);

The above assumes you want the binding to be applicable for WHEN\_FOCUSED. To register bindings for other focus states use the getInputMap method that takes an integer.

Register a new keyboard action. anAction will be invoked if a key event matching aKeyStroke occurs and aCondition is verified. The KeyStroke object defines a particular combination of a keyboard key and one or more modifiers (alt, shift, ctrl, meta).

The aCommand will be set in the delivered event if specified.

The aCondition can be one of:

WHEN\_FOCUSED The action will be invoked only when the keystroke occurs while the component has the focus. WHEN\_IN\_FOCUSED\_WINDOW The action will be invoked when the keystroke occurs while the component has the focus or if the component is in the window that has the focus. Note that the component need not be an immediate descendent of the window -- it can be anywhere in the window's containment hierarchy. In other words, whenever *any* component in the window has the focus, the action registered with this component is invoked. WHEN\_ANCESTOR\_OF\_FOCUSED\_COMPONENT The action will be invoked when the keystroke occurs while the component has the focus or if the component is an ancestor of the component that has the focus.

The combination of keystrokes and conditions lets you define high level (semantic) action events for a specified keystroke+modifier combination (using the KeyStroke class) and direct to a parent or child of a component that has the focus, or to the component itself. In other words, in any hierarchical structure of components, an arbitrary key-combination can be immediately directed to the appropriate component in the hierarchy, and cause a specific method to be invoked (usually by way of adapter objects).

If an action has already been registered for the receiving container, with the same charCode and the same modifiers, anAction will replace the action.

**Parameters:**anAction - the Action to be registeredaCommand - the command to be set in the delivered eventaKeyStroke - the KeyStroke to bind to the actionaCondition - the condition that needs to be met, see above**See Also:**[KeyStroke](http://docs.google.com/javax/swing/KeyStroke.html)

### registerKeyboardAction

public void **registerKeyboardAction**([ActionListener](http://docs.google.com/java/awt/event/ActionListener.html) anAction,  
 [KeyStroke](http://docs.google.com/javax/swing/KeyStroke.html) aKeyStroke,  
 int aCondition)

This method is now obsolete, please use a combination of getActionMap() and getInputMap() for similiar behavior.

### unregisterKeyboardAction

public void **unregisterKeyboardAction**([KeyStroke](http://docs.google.com/javax/swing/KeyStroke.html) aKeyStroke)

This method is now obsolete. To unregister an existing binding you can either remove the binding from the ActionMap/InputMap, or place a dummy binding the InputMap. Removing the binding from the InputMap allows bindings in parent InputMaps to be active, whereas putting a dummy binding in the InputMap effectively disables the binding from ever happening.

Unregisters a keyboard action. This will remove the binding from the ActionMap (if it exists) as well as the InputMaps.

### getRegisteredKeyStrokes

public [KeyStroke](http://docs.google.com/javax/swing/KeyStroke.html)[] **getRegisteredKeyStrokes**()

Returns the KeyStrokes that will initiate registered actions.

**Returns:**an array of KeyStroke objects**See Also:**[registerKeyboardAction(java.awt.event.ActionListener, java.lang.String, javax.swing.KeyStroke, int)](http://docs.google.com/javax/swing/JComponent.html#registerKeyboardAction(java.awt.event.ActionListener,%20java.lang.String,%20javax.swing.KeyStroke,%20int))

### getConditionForKeyStroke

public int **getConditionForKeyStroke**([KeyStroke](http://docs.google.com/javax/swing/KeyStroke.html) aKeyStroke)

Returns the condition that determines whether a registered action occurs in response to the specified keystroke.

For Java 2 platform v1.3, a KeyStroke can be associated with more than one condition. For example, 'a' could be bound for the two conditions WHEN\_FOCUSED and WHEN\_IN\_FOCUSED\_WINDOW condition.

**Returns:**the action-keystroke condition

### getActionForKeyStroke

public [ActionListener](http://docs.google.com/java/awt/event/ActionListener.html) **getActionForKeyStroke**([KeyStroke](http://docs.google.com/javax/swing/KeyStroke.html) aKeyStroke)

Returns the object that will perform the action registered for a given keystroke.

**Returns:**the ActionListener object invoked when the keystroke occurs

### resetKeyboardActions

public void **resetKeyboardActions**()

Unregisters all the bindings in the first tier InputMaps and ActionMap. This has the effect of removing any local bindings, and allowing the bindings defined in parent InputMap/ActionMaps (the UI is usually defined in the second tier) to persist.

### setInputMap

public final void **setInputMap**(int condition,  
 [InputMap](http://docs.google.com/javax/swing/InputMap.html) map)

Sets the InputMap to use under the condition condition to map. A null value implies you do not want any bindings to be used, even from the UI. This will not reinstall the UI InputMap (if there was one). condition has one of the following values:

* WHEN\_IN\_FOCUSED\_WINDOW
* WHEN\_FOCUSED
* WHEN\_ANCESTOR\_OF\_FOCUSED\_COMPONENT

If condition is WHEN\_IN\_FOCUSED\_WINDOW and map is not a ComponentInputMap, an IllegalArgumentException will be thrown. Similarly, if condition is not one of the values listed, an IllegalArgumentException will be thrown.

**Parameters:**condition - one of the values listed abovemap - the InputMap to use for the given condition **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if condition is WHEN\_IN\_FOCUSED\_WINDOW and map is not an instance of ComponentInputMap; or if condition is not one of the legal values specified above**Since:** 1.3

### getInputMap

public final [InputMap](http://docs.google.com/javax/swing/InputMap.html) **getInputMap**(int condition)

Returns the InputMap that is used during condition.

**Parameters:**condition - one of WHEN\_IN\_FOCUSED\_WINDOW, WHEN\_FOCUSED, WHEN\_ANCESTOR\_OF\_FOCUSED\_COMPONENT **Returns:**the InputMap for the specified condition**Since:** 1.3

### getInputMap

public final [InputMap](http://docs.google.com/javax/swing/InputMap.html) **getInputMap**()

Returns the InputMap that is used when the component has focus. This is convenience method for getInputMap(WHEN\_FOCUSED).

**Returns:**the InputMap used when the component has focus**Since:** 1.3

### setActionMap

public final void **setActionMap**([ActionMap](http://docs.google.com/javax/swing/ActionMap.html) am)

Sets the ActionMap to am. This does not set the parent of the am to be the ActionMap from the UI (if there was one), it is up to the caller to have done this.

**Parameters:**am - the new ActionMap**Since:** 1.3

### getActionMap

public final [ActionMap](http://docs.google.com/javax/swing/ActionMap.html) **getActionMap**()

Returns the ActionMap used to determine what Action to fire for particular KeyStroke binding. The returned ActionMap, unless otherwise set, will have the ActionMap from the UI set as the parent.

**Returns:**the ActionMap containing the key/action bindings**Since:** 1.3

### getBaseline

public int **getBaseline**(int width,  
 int height)

Returns the baseline. The baseline is measured from the top of the component. This method is primarily meant for LayoutManagers to align components along their baseline. A return value less than 0 indicates this component does not have a reasonable baseline and that LayoutManagers should not align this component on its baseline.

This method calls into the ComponentUI method of the same name. If this component does not have a ComponentUI -1 will be returned. If a value >= 0 is returned, then the component has a valid baseline for any size >= the minimum size and getBaselineResizeBehavior can be used to determine how the baseline changes with size.

**Overrides:**[getBaseline](http://docs.google.com/java/awt/Component.html#getBaseline(int,%20int)) in class [Component](http://docs.google.com/java/awt/Component.html) **Parameters:**width - the width to get the baseline forheight - the height to get the baseline for **Returns:**the baseline or < 0 indicating there is no reasonable baseline **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if width or height is < 0**Since:** 1.6 **See Also:**[getBaselineResizeBehavior()](http://docs.google.com/javax/swing/JComponent.html#getBaselineResizeBehavior()), [FontMetrics](http://docs.google.com/java/awt/FontMetrics.html)

### getBaselineResizeBehavior

public [Component.BaselineResizeBehavior](http://docs.google.com/java/awt/Component.BaselineResizeBehavior.html) **getBaselineResizeBehavior**()

Returns an enum indicating how the baseline of the component changes as the size changes. This method is primarily meant for layout managers and GUI builders.

This method calls into the ComponentUI method of the same name. If this component does not have a ComponentUI BaselineResizeBehavior.OTHER will be returned. Subclasses should never return null; if the baseline can not be calculated return BaselineResizeBehavior.OTHER. Callers should first ask for the baseline using getBaseline and if a value >= 0 is returned use this method. It is acceptable for this method to return a value other than BaselineResizeBehavior.OTHER even if getBaseline returns a value less than 0.

**Overrides:**[getBaselineResizeBehavior](http://docs.google.com/java/awt/Component.html#getBaselineResizeBehavior()) in class [Component](http://docs.google.com/java/awt/Component.html) **Returns:**an enum indicating how the baseline changes as the component size changes**Since:** 1.6 **See Also:**[getBaseline(int, int)](http://docs.google.com/javax/swing/JComponent.html#getBaseline(int,%20int))

### requestDefaultFocus

[@Deprecated](http://docs.google.com/java/lang/Deprecated.html)  
public boolean **requestDefaultFocus**()

**Deprecated.** *As of 1.4, replaced by FocusTraversalPolicy.getDefaultComponent(Container).requestFocus()*

In release 1.4, the focus subsystem was rearchitected. For more information, see  [How to Use the Focus Subsystem](http://java.sun.com/docs/books/tutorial/uiswing/misc/focus.html), a section in *The Java Tutorial*.

Requests focus on this JComponent's FocusTraversalPolicy's default Component. If this JComponent is a focus cycle root, then its FocusTraversalPolicy is used. Otherwise, the FocusTraversalPolicy of this JComponent's focus-cycle-root ancestor is used.

**See Also:**[FocusTraversalPolicy.getDefaultComponent(java.awt.Container)](http://docs.google.com/java/awt/FocusTraversalPolicy.html#getDefaultComponent(java.awt.Container))

### setVisible

public void **setVisible**(boolean aFlag)

Makes the component visible or invisible. Overrides Component.setVisible.

**Overrides:**[setVisible](http://docs.google.com/java/awt/Component.html#setVisible(boolean)) in class [Component](http://docs.google.com/java/awt/Component.html) **Parameters:**aFlag - true to make the component visible; false to make it invisible**See Also:**[Component.isVisible()](http://docs.google.com/java/awt/Component.html#isVisible())

### setEnabled

public void **setEnabled**(boolean enabled)

Sets whether or not this component is enabled. A component that is enabled may respond to user input, while a component that is not enabled cannot respond to user input. Some components may alter their visual representation when they are disabled in order to provide feedback to the user that they cannot take input.

Note: Disabling a component does not disable its children.

Note: Disabling a lightweight component does not prevent it from receiving MouseEvents.

**Overrides:**[setEnabled](http://docs.google.com/java/awt/Component.html#setEnabled(boolean)) in class [Component](http://docs.google.com/java/awt/Component.html) **Parameters:**enabled - true if this component should be enabled, false otherwise**See Also:**[Component.isEnabled()](http://docs.google.com/java/awt/Component.html#isEnabled()), [Component.isLightweight()](http://docs.google.com/java/awt/Component.html#isLightweight())

### setForeground

public void **setForeground**([Color](http://docs.google.com/java/awt/Color.html) fg)

Sets the foreground color of this component. It is up to the look and feel to honor this property, some may choose to ignore it.

**Overrides:**[setForeground](http://docs.google.com/java/awt/Component.html#setForeground(java.awt.Color)) in class [Component](http://docs.google.com/java/awt/Component.html) **Parameters:**fg - the desired foreground Color**See Also:**[Component.getForeground()](http://docs.google.com/java/awt/Component.html#getForeground())

### setBackground

public void **setBackground**([Color](http://docs.google.com/java/awt/Color.html) bg)

Sets the background color of this component. The background color is used only if the component is opaque, and only by subclasses of JComponent or ComponentUI implementations. Direct subclasses of JComponent must override paintComponent to honor this property.

It is up to the look and feel to honor this property, some may choose to ignore it.

**Overrides:**[setBackground](http://docs.google.com/java/awt/Component.html#setBackground(java.awt.Color)) in class [Component](http://docs.google.com/java/awt/Component.html) **Parameters:**bg - the desired background Color**See Also:**[Component.getBackground()](http://docs.google.com/java/awt/Component.html#getBackground()), [setOpaque(boolean)](http://docs.google.com/javax/swing/JComponent.html#setOpaque(boolean))

### setFont

public void **setFont**([Font](http://docs.google.com/java/awt/Font.html) font)

Sets the font for this component.

**Overrides:**[setFont](http://docs.google.com/java/awt/Container.html#setFont(java.awt.Font)) in class [Container](http://docs.google.com/java/awt/Container.html) **Parameters:**font - the desired Font for this component**See Also:**[Component.getFont()](http://docs.google.com/java/awt/Component.html#getFont())

### getDefaultLocale

public static [Locale](http://docs.google.com/java/util/Locale.html) **getDefaultLocale**()

Returns the default locale used to initialize each JComponent's locale property upon creation. The default locale has "AppContext" scope so that applets (and potentially multiple lightweight applications running in a single VM) can have their own setting. An applet can safely alter its default locale because it will have no affect on other applets (or the browser).

**Returns:**the default Locale.**Since:** 1.4 **See Also:**[setDefaultLocale(java.util.Locale)](http://docs.google.com/javax/swing/JComponent.html#setDefaultLocale(java.util.Locale)), [Component.getLocale()](http://docs.google.com/java/awt/Component.html#getLocale()), [Component.setLocale(java.util.Locale)](http://docs.google.com/java/awt/Component.html#setLocale(java.util.Locale))

### setDefaultLocale

public static void **setDefaultLocale**([Locale](http://docs.google.com/java/util/Locale.html) l)

Sets the default locale used to initialize each JComponent's locale property upon creation. The initial value is the VM's default locale. The default locale has "AppContext" scope so that applets (and potentially multiple lightweight applications running in a single VM) can have their own setting. An applet can safely alter its default locale because it will have no affect on other applets (or the browser).

**Parameters:**l - the desired default Locale for new components.**Since:** 1.4 **See Also:**[getDefaultLocale()](http://docs.google.com/javax/swing/JComponent.html#getDefaultLocale()), [Component.getLocale()](http://docs.google.com/java/awt/Component.html#getLocale()), [Component.setLocale(java.util.Locale)](http://docs.google.com/java/awt/Component.html#setLocale(java.util.Locale))

### processComponentKeyEvent

protected void **processComponentKeyEvent**([KeyEvent](http://docs.google.com/java/awt/event/KeyEvent.html) e)

Processes any key events that the component itself recognizes. This is called after the focus manager and any interested listeners have been given a chance to steal away the event. This method is called only if the event has not yet been consumed. This method is called prior to the keyboard UI logic.

This method is implemented to do nothing. Subclasses would normally override this method if they process some key events themselves. If the event is processed, it should be consumed.

### processKeyEvent

protected void **processKeyEvent**([KeyEvent](http://docs.google.com/java/awt/event/KeyEvent.html) e)

Overrides processKeyEvent to process events.

**Overrides:**[processKeyEvent](http://docs.google.com/java/awt/Component.html#processKeyEvent(java.awt.event.KeyEvent)) in class [Component](http://docs.google.com/java/awt/Component.html) **Parameters:**e - the key event**See Also:**[KeyEvent](http://docs.google.com/java/awt/event/KeyEvent.html), [KeyListener](http://docs.google.com/java/awt/event/KeyListener.html), [KeyboardFocusManager](http://docs.google.com/java/awt/KeyboardFocusManager.html), [DefaultKeyboardFocusManager](http://docs.google.com/java/awt/DefaultKeyboardFocusManager.html), [Component.processEvent(java.awt.AWTEvent)](http://docs.google.com/java/awt/Component.html#processEvent(java.awt.AWTEvent)), [Component.dispatchEvent(java.awt.AWTEvent)](http://docs.google.com/java/awt/Component.html#dispatchEvent(java.awt.AWTEvent)), [Component.addKeyListener(java.awt.event.KeyListener)](http://docs.google.com/java/awt/Component.html#addKeyListener(java.awt.event.KeyListener)), [Component.enableEvents(long)](http://docs.google.com/java/awt/Component.html#enableEvents(long)), [Component.isShowing()](http://docs.google.com/java/awt/Component.html#isShowing())

### processKeyBinding

protected boolean **processKeyBinding**([KeyStroke](http://docs.google.com/javax/swing/KeyStroke.html) ks,  
 [KeyEvent](http://docs.google.com/java/awt/event/KeyEvent.html) e,  
 int condition,  
 boolean pressed)

Invoked to process the key bindings for ks as the result of the KeyEvent e. This obtains the appropriate InputMap, gets the binding, gets the action from the ActionMap, and then (if the action is found and the component is enabled) invokes notifyAction to notify the action.

**Parameters:**ks - the KeyStroke queriede - the KeyEventcondition - one of the following values:

* JComponent.WHEN\_FOCUSED
* JComponent.WHEN\_ANCESTOR\_OF\_FOCUSED\_COMPONENT
* JComponent.WHEN\_IN\_FOCUSED\_WINDOW

pressed - true if the key is pressed **Returns:**true if there was a binding to an action, and the action was enabled**Since:** 1.3

### setToolTipText

public void **setToolTipText**([String](http://docs.google.com/java/lang/String.html) text)

Registers the text to display in a tool tip. The text displays when the cursor lingers over the component.

See [How to Use Tool Tips](http://java.sun.com/docs/books/tutorial/uiswing/components/tooltip.html) in *The Java Tutorial* for further documentation.

**Parameters:**text - the string to display; if the text is null, the tool tip is turned off for this component**See Also:**[TOOL\_TIP\_TEXT\_KEY](http://docs.google.com/javax/swing/JComponent.html#TOOL_TIP_TEXT_KEY)

### getToolTipText

public [String](http://docs.google.com/java/lang/String.html) **getToolTipText**()

Returns the tooltip string that has been set with setToolTipText.

**Returns:**the text of the tool tip**See Also:**[TOOL\_TIP\_TEXT\_KEY](http://docs.google.com/javax/swing/JComponent.html#TOOL_TIP_TEXT_KEY)

### getToolTipText

public [String](http://docs.google.com/java/lang/String.html) **getToolTipText**([MouseEvent](http://docs.google.com/java/awt/event/MouseEvent.html) event)

Returns the string to be used as the tooltip for *event*. By default this returns any string set using setToolTipText. If a component provides more extensive API to support differing tooltips at different locations, this method should be overridden.

### getToolTipLocation

public [Point](http://docs.google.com/java/awt/Point.html) **getToolTipLocation**([MouseEvent](http://docs.google.com/java/awt/event/MouseEvent.html) event)

Returns the tooltip location in this component's coordinate system. If null is returned, Swing will choose a location. The default implementation returns null.

**Parameters:**event - the MouseEvent that caused the ToolTipManager to show the tooltip **Returns:**always returns null

### getPopupLocation

public [Point](http://docs.google.com/java/awt/Point.html) **getPopupLocation**([MouseEvent](http://docs.google.com/java/awt/event/MouseEvent.html) event)

Returns the preferred location to display the popup menu in this component's coordinate system. It is up to the look and feel to honor this property, some may choose to ignore it. If null, the look and feel will choose a suitable location.

**Parameters:**event - the MouseEvent that triggered the popup to be shown, or null if the popup is not being shown as the result of a mouse event **Returns:**location to display the JPopupMenu, or null**Since:** 1.5

### createToolTip

public [JToolTip](http://docs.google.com/javax/swing/JToolTip.html) **createToolTip**()

Returns the instance of JToolTip that should be used to display the tooltip. Components typically would not override this method, but it can be used to cause different tooltips to be displayed differently.

**Returns:**the JToolTip used to display this toolTip

### scrollRectToVisible

public void **scrollRectToVisible**([Rectangle](http://docs.google.com/java/awt/Rectangle.html) aRect)

Forwards the scrollRectToVisible() message to the JComponent's parent. Components that can service the request, such as JViewport, override this method and perform the scrolling.

**Parameters:**aRect - the visible Rectangle**See Also:**[JViewport](http://docs.google.com/javax/swing/JViewport.html)

### setAutoscrolls

public void **setAutoscrolls**(boolean autoscrolls)

Sets the autoscrolls property. If true mouse dragged events will be synthetically generated when the mouse is dragged outside of the component's bounds and mouse motion has paused (while the button continues to be held down). The synthetic events make it appear that the drag gesture has resumed in the direction established when the component's boundary was crossed. Components that support autoscrolling must handle mouseDragged events by calling scrollRectToVisible with a rectangle that contains the mouse event's location. All of the Swing components that support item selection and are typically displayed in a JScrollPane (JTable, JList, JTree, JTextArea, and JEditorPane) already handle mouse dragged events in this way. To enable autoscrolling in any other component, add a mouse motion listener that calls scrollRectToVisible. For example, given a JPanel, myPanel:

MouseMotionListener doScrollRectToVisible = new MouseMotionAdapter() {  
 public void mouseDragged(MouseEvent e) {  
 Rectangle r = new Rectangle(e.getX(), e.getY(), 1, 1);  
 ((JPanel)e.getSource()).scrollRectToVisible(r);  
 }  
 };  
 myPanel.addMouseMotionListener(doScrollRectToVisible);

The default value of the autoScrolls property is false.

**Parameters:**autoscrolls - if true, synthetic mouse dragged events are generated when the mouse is dragged outside of a component's bounds and the mouse button continues to be held down; otherwise false**See Also:**[getAutoscrolls()](http://docs.google.com/javax/swing/JComponent.html#getAutoscrolls()), [JViewport](http://docs.google.com/javax/swing/JViewport.html), [JScrollPane](http://docs.google.com/javax/swing/JScrollPane.html)

### getAutoscrolls

public boolean **getAutoscrolls**()

Gets the autoscrolls property.

**Returns:**the value of the autoscrolls property**See Also:**[JViewport](http://docs.google.com/javax/swing/JViewport.html), [setAutoscrolls(boolean)](http://docs.google.com/javax/swing/JComponent.html#setAutoscrolls(boolean))

### setTransferHandler

public void **setTransferHandler**([TransferHandler](http://docs.google.com/javax/swing/TransferHandler.html) newHandler)

Sets the transferHandler property, which is null if the component does not support data transfer operations.

If newHandler is not null, and the system property suppressSwingDropSupport is not true, this will install a DropTarget on the JComponent. The default for the system property is false, so that a DropTarget will be added.

Please see  [How to Use Drag and Drop and Data Transfer](http://java.sun.com/docs/books/tutorial/uiswing/misc/dnd.html), a section in *The Java Tutorial*, for more information.

**Parameters:**newHandler - mechanism for transfer of data to and from the component**Since:** 1.4 **See Also:**[TransferHandler](http://docs.google.com/javax/swing/TransferHandler.html), [getTransferHandler()](http://docs.google.com/javax/swing/JComponent.html#getTransferHandler())

### getTransferHandler

public [TransferHandler](http://docs.google.com/javax/swing/TransferHandler.html) **getTransferHandler**()

Gets the transferHandler property.

**Returns:**the value of the transferHandler property**Since:** 1.4 **See Also:**[TransferHandler](http://docs.google.com/javax/swing/TransferHandler.html), [setTransferHandler(javax.swing.TransferHandler)](http://docs.google.com/javax/swing/JComponent.html#setTransferHandler(javax.swing.TransferHandler))

### processMouseEvent

protected void **processMouseEvent**([MouseEvent](http://docs.google.com/java/awt/event/MouseEvent.html) e)

Processes mouse events occurring on this component by dispatching them to any registered MouseListener objects, refer to [Component.processMouseEvent(MouseEvent)](http://docs.google.com/java/awt/Component.html#processMouseEvent(java.awt.event.MouseEvent)) for a complete description of this method.

**Overrides:**[processMouseEvent](http://docs.google.com/java/awt/Component.html#processMouseEvent(java.awt.event.MouseEvent)) in class [Component](http://docs.google.com/java/awt/Component.html) **Parameters:**e - the mouse event**Since:** 1.5 **See Also:**[Component.processMouseEvent(java.awt.event.MouseEvent)](http://docs.google.com/java/awt/Component.html#processMouseEvent(java.awt.event.MouseEvent))

### processMouseMotionEvent

protected void **processMouseMotionEvent**([MouseEvent](http://docs.google.com/java/awt/event/MouseEvent.html) e)

Processes mouse motion events, such as MouseEvent.MOUSE\_DRAGGED.

**Overrides:**[processMouseMotionEvent](http://docs.google.com/java/awt/Component.html#processMouseMotionEvent(java.awt.event.MouseEvent)) in class [Component](http://docs.google.com/java/awt/Component.html) **Parameters:**e - the MouseEvent**See Also:**[MouseEvent](http://docs.google.com/java/awt/event/MouseEvent.html)

### enable

[@Deprecated](http://docs.google.com/java/lang/Deprecated.html)  
public void **enable**()

**Deprecated.** *As of JDK version 1.1, replaced by java.awt.Component.setEnabled(boolean).*

**Overrides:**[enable](http://docs.google.com/java/awt/Component.html#enable()) in class [Component](http://docs.google.com/java/awt/Component.html)

### disable

[@Deprecated](http://docs.google.com/java/lang/Deprecated.html)  
public void **disable**()

**Deprecated.** *As of JDK version 1.1, replaced by java.awt.Component.setEnabled(boolean).*

**Overrides:**[disable](http://docs.google.com/java/awt/Component.html#disable()) in class [Component](http://docs.google.com/java/awt/Component.html)

### getAccessibleContext

public [AccessibleContext](http://docs.google.com/javax/accessibility/AccessibleContext.html) **getAccessibleContext**()

Returns the AccessibleContext associated with this JComponent. The method implemented by this base class returns null. Classes that extend JComponent should implement this method to return the AccessibleContext associated with the subclass.

**Overrides:**[getAccessibleContext](http://docs.google.com/java/awt/Component.html#getAccessibleContext()) in class [Component](http://docs.google.com/java/awt/Component.html) **Returns:**the AccessibleContext of this JComponent

### getClientProperty

public final [Object](http://docs.google.com/java/lang/Object.html) **getClientProperty**([Object](http://docs.google.com/java/lang/Object.html) key)

Returns the value of the property with the specified key. Only properties added with putClientProperty will return a non-null value.

**Parameters:**key - the being queried **Returns:**the value of this property or null**See Also:**[putClientProperty(java.lang.Object, java.lang.Object)](http://docs.google.com/javax/swing/JComponent.html#putClientProperty(java.lang.Object,%20java.lang.Object))

### putClientProperty

public final void **putClientProperty**([Object](http://docs.google.com/java/lang/Object.html) key,  
 [Object](http://docs.google.com/java/lang/Object.html) value)

Adds an arbitrary key/value "client property" to this component.

The get/putClientProperty methods provide access to a small per-instance hashtable. Callers can use get/putClientProperty to annotate components that were created by another module. For example, a layout manager might store per child constraints this way. For example:

componentA.putClientProperty("to the left of", componentB);

If value is null this method will remove the property. Changes to client properties are reported with PropertyChange events. The name of the property (for the sake of PropertyChange events) is key.toString().

The clientProperty dictionary is not intended to support large scale extensions to JComponent nor should be it considered an alternative to subclassing when designing a new component.

**Parameters:**key - the new client property keyvalue - the new client property value; if null this method will remove the property**See Also:**[getClientProperty(java.lang.Object)](http://docs.google.com/javax/swing/JComponent.html#getClientProperty(java.lang.Object)), [Container.addPropertyChangeListener(java.beans.PropertyChangeListener)](http://docs.google.com/java/awt/Container.html#addPropertyChangeListener(java.beans.PropertyChangeListener))

### setFocusTraversalKeys

public void **setFocusTraversalKeys**(int id,  
 [Set](http://docs.google.com/java/util/Set.html)<? extends [AWTKeyStroke](http://docs.google.com/java/awt/AWTKeyStroke.html)> keystrokes)

Sets the focus traversal keys for a given traversal operation for this Component. Refer to [Component.setFocusTraversalKeys(int, java.util.Set)](http://docs.google.com/java/awt/Component.html#setFocusTraversalKeys(int,%20java.util.Set)) for a complete description of this method.

**Overrides:**[setFocusTraversalKeys](http://docs.google.com/java/awt/Container.html#setFocusTraversalKeys(int,%20java.util.Set)) in class [Container](http://docs.google.com/java/awt/Container.html) **Parameters:**id - one of KeyboardFocusManager.FORWARD\_TRAVERSAL\_KEYS, KeyboardFocusManager.BACKWARD\_TRAVERSAL\_KEYS, or KeyboardFocusManager.UP\_CYCLE\_TRAVERSAL\_KEYSkeystrokes - the Set of AWTKeyStroke for the specified operation **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if id is not one of KeyboardFocusManager.FORWARD\_TRAVERSAL\_KEYS, KeyboardFocusManager.BACKWARD\_TRAVERSAL\_KEYS, or KeyboardFocusManager.UP\_CYCLE\_TRAVERSAL\_KEYS, or if keystrokes contains null, or if any Object in keystrokes is not an AWTKeyStroke, or if any keystroke represents a KEY\_TYPED event, or if any keystroke already maps to another focus traversal operation for this Component**Since:** 1.5 **See Also:**[KeyboardFocusManager.FORWARD\_TRAVERSAL\_KEYS](http://docs.google.com/java/awt/KeyboardFocusManager.html#FORWARD_TRAVERSAL_KEYS), [KeyboardFocusManager.BACKWARD\_TRAVERSAL\_KEYS](http://docs.google.com/java/awt/KeyboardFocusManager.html#BACKWARD_TRAVERSAL_KEYS), [KeyboardFocusManager.UP\_CYCLE\_TRAVERSAL\_KEYS](http://docs.google.com/java/awt/KeyboardFocusManager.html#UP_CYCLE_TRAVERSAL_KEYS)

### isLightweightComponent

public static boolean **isLightweightComponent**([Component](http://docs.google.com/java/awt/Component.html) c)

Returns true if this component is lightweight, that is, if it doesn't have a native window system peer.

**Returns:**true if this component is lightweight

### reshape

[@Deprecated](http://docs.google.com/java/lang/Deprecated.html)  
public void **reshape**(int x,  
 int y,  
 int w,  
 int h)

**Deprecated.** *As of JDK 5, replaced by Component.setBounds(int, int, int, int).*

*Moves and resizes this component.*

**Overrides:**[reshape](http://docs.google.com/java/awt/Component.html#reshape(int,%20int,%20int,%20int)) in class [Component](http://docs.google.com/java/awt/Component.html) **Parameters:**x - the new horizontal locationy - the new vertical locationw - the new widthh - the new height**See Also:**[Component.setBounds(int, int, int, int)](http://docs.google.com/java/awt/Component.html#setBounds(int,%20int,%20int,%20int))

### getBounds

public [Rectangle](http://docs.google.com/java/awt/Rectangle.html) **getBounds**([Rectangle](http://docs.google.com/java/awt/Rectangle.html) rv)

Stores the bounds of this component into "return value" rv and returns rv. If rv is null a new Rectangle is allocated. This version of getBounds is useful if the caller wants to avoid allocating a new Rectangle object on the heap.

**Overrides:**[getBounds](http://docs.google.com/java/awt/Component.html#getBounds(java.awt.Rectangle)) in class [Component](http://docs.google.com/java/awt/Component.html) **Parameters:**rv - the return value, modified to the component's bounds **Returns:**rv; if rv is null return a newly created Rectangle with this component's bounds

### getSize

public [Dimension](http://docs.google.com/java/awt/Dimension.html) **getSize**([Dimension](http://docs.google.com/java/awt/Dimension.html) rv)

Stores the width/height of this component into "return value" rv and returns rv. If rv is null a new Dimension object is allocated. This version of getSize is useful if the caller wants to avoid allocating a new Dimension object on the heap.

**Overrides:**[getSize](http://docs.google.com/java/awt/Component.html#getSize(java.awt.Dimension)) in class [Component](http://docs.google.com/java/awt/Component.html) **Parameters:**rv - the return value, modified to the component's size **Returns:**rv

### getLocation

public [Point](http://docs.google.com/java/awt/Point.html) **getLocation**([Point](http://docs.google.com/java/awt/Point.html) rv)

Stores the x,y origin of this component into "return value" rv and returns rv. If rv is null a new Point is allocated. This version of getLocation is useful if the caller wants to avoid allocating a new Point object on the heap.

**Overrides:**[getLocation](http://docs.google.com/java/awt/Component.html#getLocation(java.awt.Point)) in class [Component](http://docs.google.com/java/awt/Component.html) **Parameters:**rv - the return value, modified to the component's location **Returns:**rv

### getX

public int **getX**()

Returns the current x coordinate of the component's origin. This method is preferable to writing component.getBounds().x, or component.getLocation().x because it doesn't cause any heap allocations.

**Overrides:**[getX](http://docs.google.com/java/awt/Component.html#getX()) in class [Component](http://docs.google.com/java/awt/Component.html) **Returns:**the current x coordinate of the component's origin

### getY

public int **getY**()

Returns the current y coordinate of the component's origin. This method is preferable to writing component.getBounds().y, or component.getLocation().y because it doesn't cause any heap allocations.

**Overrides:**[getY](http://docs.google.com/java/awt/Component.html#getY()) in class [Component](http://docs.google.com/java/awt/Component.html) **Returns:**the current y coordinate of the component's origin

### getWidth

public int **getWidth**()

Returns the current width of this component. This method is preferable to writing component.getBounds().width, or component.getSize().width because it doesn't cause any heap allocations.

**Overrides:**[getWidth](http://docs.google.com/java/awt/Component.html#getWidth()) in class [Component](http://docs.google.com/java/awt/Component.html) **Returns:**the current width of this component

### getHeight

public int **getHeight**()

Returns the current height of this component. This method is preferable to writing component.getBounds().height, or component.getSize().height because it doesn't cause any heap allocations.

**Overrides:**[getHeight](http://docs.google.com/java/awt/Component.html#getHeight()) in class [Component](http://docs.google.com/java/awt/Component.html) **Returns:**the current height of this component

### isOpaque

public boolean **isOpaque**()

Returns true if this component is completely opaque.

An opaque component paints every pixel within its rectangular bounds. A non-opaque component paints only a subset of its pixels or none at all, allowing the pixels underneath it to "show through". Therefore, a component that does not fully paint its pixels provides a degree of transparency.

Subclasses that guarantee to always completely paint their contents should override this method and return true.

**Overrides:**[isOpaque](http://docs.google.com/java/awt/Component.html#isOpaque()) in class [Component](http://docs.google.com/java/awt/Component.html) **Returns:**true if this component is completely opaque**See Also:**[setOpaque(boolean)](http://docs.google.com/javax/swing/JComponent.html#setOpaque(boolean))

### setOpaque

public void **setOpaque**(boolean isOpaque)

If true the component paints every pixel within its bounds. Otherwise, the component may not paint some or all of its pixels, allowing the underlying pixels to show through.

The default value of this property is false for JComponent. However, the default value for this property on most standard JComponent subclasses (such as JButton and JTree) is look-and-feel dependent.

**Parameters:**isOpaque - true if this component should be opaque**See Also:**[isOpaque()](http://docs.google.com/javax/swing/JComponent.html#isOpaque())

### computeVisibleRect

public void **computeVisibleRect**([Rectangle](http://docs.google.com/java/awt/Rectangle.html) visibleRect)

Returns the Component's "visible rect rectangle" - the intersection of the visible rectangles for this component and all of its ancestors. The return value is stored in visibleRect.

**Parameters:**visibleRect - a Rectangle computed as the intersection of all visible rectangles for this component and all of its ancestors -- this is the return value for this method**See Also:**[getVisibleRect()](http://docs.google.com/javax/swing/JComponent.html#getVisibleRect())

### getVisibleRect

public [Rectangle](http://docs.google.com/java/awt/Rectangle.html) **getVisibleRect**()

Returns the Component's "visible rectangle" - the intersection of this component's visible rectangle, new Rectangle(0, 0, getWidth(), getHeight()), and all of its ancestors' visible rectangles.

**Returns:**the visible rectangle

### firePropertyChange

public void **firePropertyChange**([String](http://docs.google.com/java/lang/String.html) propertyName,  
 boolean oldValue,  
 boolean newValue)

Support for reporting bound property changes for boolean properties. This method can be called when a bound property has changed and it will send the appropriate PropertyChangeEvent to any registered PropertyChangeListeners.

**Overrides:**[firePropertyChange](http://docs.google.com/java/awt/Component.html#firePropertyChange(java.lang.String,%20boolean,%20boolean)) in class [Component](http://docs.google.com/java/awt/Component.html) **Parameters:**propertyName - the property whose value has changedoldValue - the property's previous valuenewValue - the property's new value

### firePropertyChange

public void **firePropertyChange**([String](http://docs.google.com/java/lang/String.html) propertyName,  
 int oldValue,  
 int newValue)

Support for reporting bound property changes for integer properties. This method can be called when a bound property has changed and it will send the appropriate PropertyChangeEvent to any registered PropertyChangeListeners.

**Overrides:**[firePropertyChange](http://docs.google.com/java/awt/Component.html#firePropertyChange(java.lang.String,%20int,%20int)) in class [Component](http://docs.google.com/java/awt/Component.html) **Parameters:**propertyName - the property whose value has changedoldValue - the property's previous valuenewValue - the property's new value

### firePropertyChange

public void **firePropertyChange**([String](http://docs.google.com/java/lang/String.html) propertyName,  
 char oldValue,  
 char newValue)

**Description copied from class:** [**Component**](http://docs.google.com/java/awt/Component.html#firePropertyChange(java.lang.String,%20char,%20char)) Reports a bound property change.

**Overrides:**[firePropertyChange](http://docs.google.com/java/awt/Component.html#firePropertyChange(java.lang.String,%20char,%20char)) in class [Component](http://docs.google.com/java/awt/Component.html) **Parameters:**propertyName - the programmatic name of the property that was changedoldValue - the old value of the property (as a char)newValue - the new value of the property (as a char)**See Also:**[Component.firePropertyChange(java.lang.String, java.lang.Object, java.lang.Object)](http://docs.google.com/java/awt/Component.html#firePropertyChange(java.lang.String,%20java.lang.Object,%20java.lang.Object))

### fireVetoableChange

protected void **fireVetoableChange**([String](http://docs.google.com/java/lang/String.html) propertyName,  
 [Object](http://docs.google.com/java/lang/Object.html) oldValue,  
 [Object](http://docs.google.com/java/lang/Object.html) newValue)  
 throws [PropertyVetoException](http://docs.google.com/java/beans/PropertyVetoException.html)

Supports reporting constrained property changes. This method can be called when a constrained property has changed and it will send the appropriate PropertyChangeEvent to any registered VetoableChangeListeners.

**Parameters:**propertyName - the name of the property that was listened onoldValue - the old value of the propertynewValue - the new value of the property **Throws:** [PropertyVetoException](http://docs.google.com/java/beans/PropertyVetoException.html) - when the attempt to set the property is vetoed by the component

### addVetoableChangeListener

public void **addVetoableChangeListener**([VetoableChangeListener](http://docs.google.com/java/beans/VetoableChangeListener.html) listener)

Adds a VetoableChangeListener to the listener list. The listener is registered for all properties.

**Parameters:**listener - the VetoableChangeListener to be added

### removeVetoableChangeListener

public void **removeVetoableChangeListener**([VetoableChangeListener](http://docs.google.com/java/beans/VetoableChangeListener.html) listener)

Removes a VetoableChangeListener from the listener list. This removes a VetoableChangeListener that was registered for all properties.

**Parameters:**listener - the VetoableChangeListener to be removed

### getVetoableChangeListeners

public [VetoableChangeListener](http://docs.google.com/java/beans/VetoableChangeListener.html)[] **getVetoableChangeListeners**()

Returns an array of all the vetoable change listeners registered on this component.

**Returns:**all of the component's VetoableChangeListeners or an empty array if no vetoable change listeners are currently registered**Since:** 1.4 **See Also:**[addVetoableChangeListener(java.beans.VetoableChangeListener)](http://docs.google.com/javax/swing/JComponent.html#addVetoableChangeListener(java.beans.VetoableChangeListener)), [removeVetoableChangeListener(java.beans.VetoableChangeListener)](http://docs.google.com/javax/swing/JComponent.html#removeVetoableChangeListener(java.beans.VetoableChangeListener))

### getTopLevelAncestor

public [Container](http://docs.google.com/java/awt/Container.html) **getTopLevelAncestor**()

Returns the top-level ancestor of this component (either the containing Window or Applet), or null if this component has not been added to any container.

**Returns:**the top-level Container that this component is in, or null if not in any container

### addAncestorListener

public void **addAncestorListener**([AncestorListener](http://docs.google.com/javax/swing/event/AncestorListener.html) listener)

Registers listener so that it will receive AncestorEvents when it or any of its ancestors move or are made visible or invisible. Events are also sent when the component or its ancestors are added or removed from the containment hierarchy.

**Parameters:**listener - the AncestorListener to register**See Also:**[AncestorEvent](http://docs.google.com/javax/swing/event/AncestorEvent.html)

### removeAncestorListener

public void **removeAncestorListener**([AncestorListener](http://docs.google.com/javax/swing/event/AncestorListener.html) listener)

Unregisters listener so that it will no longer receive AncestorEvents.

**Parameters:**listener - the AncestorListener to be removed**See Also:**[addAncestorListener(javax.swing.event.AncestorListener)](http://docs.google.com/javax/swing/JComponent.html#addAncestorListener(javax.swing.event.AncestorListener))

### getAncestorListeners

public [AncestorListener](http://docs.google.com/javax/swing/event/AncestorListener.html)[] **getAncestorListeners**()

Returns an array of all the ancestor listeners registered on this component.

**Returns:**all of the component's AncestorListeners or an empty array if no ancestor listeners are currently registered**Since:** 1.4 **See Also:**[addAncestorListener(javax.swing.event.AncestorListener)](http://docs.google.com/javax/swing/JComponent.html#addAncestorListener(javax.swing.event.AncestorListener)), [removeAncestorListener(javax.swing.event.AncestorListener)](http://docs.google.com/javax/swing/JComponent.html#removeAncestorListener(javax.swing.event.AncestorListener))

### getListeners

public <T extends [EventListener](http://docs.google.com/java/util/EventListener.html)> T[] **getListeners**([Class](http://docs.google.com/java/lang/Class.html)<T> listenerType)

Returns an array of all the objects currently registered as *Foo*Listeners upon this JComponent. *Foo*Listeners are registered using the add*Foo*Listener method.

You can specify the listenerType argument with a class literal, such as *Foo*Listener.class. For example, you can query a JComponent c for its mouse listeners with the following code:

MouseListener[] mls = (MouseListener[])(c.getListeners(MouseListener.class));

If no such listeners exist, this method returns an empty array.

**Overrides:**[getListeners](http://docs.google.com/java/awt/Container.html#getListeners(java.lang.Class)) in class [Container](http://docs.google.com/java/awt/Container.html) **Parameters:**listenerType - the type of listeners requested; this parameter should specify an interface that descends from java.util.EventListener **Returns:**an array of all objects registered as *Foo*Listeners on this component, or an empty array if no such listeners have been added **Throws:** [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if listenerType doesn't specify a class or interface that implements java.util.EventListener**Since:** 1.3 **See Also:**[getVetoableChangeListeners()](http://docs.google.com/javax/swing/JComponent.html#getVetoableChangeListeners()), [getAncestorListeners()](http://docs.google.com/javax/swing/JComponent.html#getAncestorListeners())

### addNotify

public void **addNotify**()

Notifies this component that it now has a parent component. When this method is invoked, the chain of parent components is set up with KeyboardAction event listeners.

**Overrides:**[addNotify](http://docs.google.com/java/awt/Container.html#addNotify()) in class [Container](http://docs.google.com/java/awt/Container.html) **See Also:**[registerKeyboardAction(java.awt.event.ActionListener, java.lang.String, javax.swing.KeyStroke, int)](http://docs.google.com/javax/swing/JComponent.html#registerKeyboardAction(java.awt.event.ActionListener,%20java.lang.String,%20javax.swing.KeyStroke,%20int))

### removeNotify

public void **removeNotify**()

Notifies this component that it no longer has a parent component. When this method is invoked, any KeyboardActions set up in the the chain of parent components are removed.

**Overrides:**[removeNotify](http://docs.google.com/java/awt/Container.html#removeNotify()) in class [Container](http://docs.google.com/java/awt/Container.html) **See Also:**[registerKeyboardAction(java.awt.event.ActionListener, java.lang.String, javax.swing.KeyStroke, int)](http://docs.google.com/javax/swing/JComponent.html#registerKeyboardAction(java.awt.event.ActionListener,%20java.lang.String,%20javax.swing.KeyStroke,%20int))

### repaint

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

Adds the specified region to the dirty region list if the component is showing. The component will be repainted after all of the currently pending events have been dispatched.

**Overrides:**[repaint](http://docs.google.com/java/awt/Component.html#repaint(long,%20int,%20int,%20int,%20int)) in class [Component](http://docs.google.com/java/awt/Component.html) **Parameters:**tm - this parameter is not usedx - the x value of the dirty regiony - the y value of the dirty regionwidth - the width of the dirty regionheight - the height of the dirty region**See Also:**[Component.isShowing()](http://docs.google.com/java/awt/Component.html#isShowing()), [RepaintManager.addDirtyRegion(javax.swing.JComponent, int, int, int, int)](http://docs.google.com/javax/swing/RepaintManager.html#addDirtyRegion(javax.swing.JComponent,%20int,%20int,%20int,%20int))

### repaint

public void **repaint**([Rectangle](http://docs.google.com/java/awt/Rectangle.html) r)

Adds the specified region to the dirty region list if the component is showing. The component will be repainted after all of the currently pending events have been dispatched.

**Parameters:**r - a Rectangle containing the dirty region**See Also:**[Component.isShowing()](http://docs.google.com/java/awt/Component.html#isShowing()), [RepaintManager.addDirtyRegion(javax.swing.JComponent, int, int, int, int)](http://docs.google.com/javax/swing/RepaintManager.html#addDirtyRegion(javax.swing.JComponent,%20int,%20int,%20int,%20int))

### revalidate

public void **revalidate**()

Supports deferred automatic layout.

Calls invalidate and then adds this component's validateRoot to a list of components that need to be validated. Validation will occur after all currently pending events have been dispatched. In other words after this method is called, the first validateRoot (if any) found when walking up the containment hierarchy of this component will be validated. By default, JRootPane, JScrollPane, and JTextField return true from isValidateRoot.

This method will automatically be called on this component when a property value changes such that size, location, or internal layout of this component has been affected. This automatic updating differs from the AWT because programs generally no longer need to invoke validate to get the contents of the GUI to update.

**See Also:**[Component.invalidate()](http://docs.google.com/java/awt/Component.html#invalidate()), [Container.validate()](http://docs.google.com/java/awt/Container.html#validate()), [isValidateRoot()](http://docs.google.com/javax/swing/JComponent.html#isValidateRoot()), [RepaintManager.addInvalidComponent(javax.swing.JComponent)](http://docs.google.com/javax/swing/RepaintManager.html#addInvalidComponent(javax.swing.JComponent))

### isValidateRoot

public boolean **isValidateRoot**()

If this method returns true, revalidate calls by descendants of this component will cause the entire tree beginning with this root to be validated. Returns false by default. JScrollPane overrides this method and returns true.

**Returns:**always returns false**See Also:**[revalidate()](http://docs.google.com/javax/swing/JComponent.html#revalidate()), [Component.invalidate()](http://docs.google.com/java/awt/Component.html#invalidate()), [Container.validate()](http://docs.google.com/java/awt/Container.html#validate())

### isOptimizedDrawingEnabled

public boolean **isOptimizedDrawingEnabled**()

Returns true if this component tiles its children -- that is, if it can guarantee that the children will not overlap. The repainting system is substantially more efficient in this common case. JComponent subclasses that can't make this guarantee, such as JLayeredPane, should override this method to return false.

**Returns:**always returns true

### paintImmediately

public void **paintImmediately**(int x,  
 int y,  
 int w,  
 int h)

Paints the specified region in this component and all of its descendants that overlap the region, immediately.

It's rarely necessary to call this method. In most cases it's more efficient to call repaint, which defers the actual painting and can collapse redundant requests into a single paint call. This method is useful if one needs to update the display while the current event is being dispatched.

**Parameters:**x - the x value of the region to be paintedy - the y value of the region to be paintedw - the width of the region to be paintedh - the height of the region to be painted**See Also:**[repaint(long, int, int, int, int)](http://docs.google.com/javax/swing/JComponent.html#repaint(long,%20int,%20int,%20int,%20int))

### paintImmediately

public void **paintImmediately**([Rectangle](http://docs.google.com/java/awt/Rectangle.html) r)

Paints the specified region now.

**Parameters:**r - a Rectangle containing the region to be painted

### setDoubleBuffered

public void **setDoubleBuffered**(boolean aFlag)

Sets whether this component should use a buffer to paint. If set to true, all the drawing from this component will be done in an offscreen painting buffer. The offscreen painting buffer will the be copied onto the screen. If a Component is buffered and one of its ancestor is also buffered, the ancestor buffer will be used.

**Parameters:**aFlag - if true, set this component to be double buffered

### isDoubleBuffered

public boolean **isDoubleBuffered**()

Returns whether this component should use a buffer to paint.

**Overrides:**[isDoubleBuffered](http://docs.google.com/java/awt/Component.html#isDoubleBuffered()) in class [Component](http://docs.google.com/java/awt/Component.html) **Returns:**true if this component is double buffered, otherwise false

### getRootPane

public [JRootPane](http://docs.google.com/javax/swing/JRootPane.html) **getRootPane**()

Returns the JRootPane ancestor for this component.

**Returns:**the JRootPane that contains this component, or null if no JRootPane is found

### paramString

protected [String](http://docs.google.com/java/lang/String.html) **paramString**()

Returns a string representation of this JComponent. This method is intended to be used only for debugging purposes, and the content and format of the returned string may vary between implementations. The returned string may be empty but may not be null.

**Overrides:**[paramString](http://docs.google.com/java/awt/Container.html#paramString()) in class [Container](http://docs.google.com/java/awt/Container.html) **Returns:**a string representation of this JComponent

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/JComponent.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/javax/swing/JComboBox.KeySelectionManager.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/JComponent.AccessibleJComponent.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/JComponent.html)    [**NO FRAMES**](http://docs.google.com/JComponent.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | [FIELD](#3dy6vkm) | [CONSTR](#2s8eyo1) | [METHOD](#17dp8vu) | DETAIL: [FIELD](#35nkun2) | [CONSTR](#1ci93xb) | [METHOD](#2bn6wsx) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).