| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/BasicComboBoxUI.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/plaf/basic/BasicComboBoxRenderer.UIResource.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.ComboBoxLayoutManager.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/plaf/basic/BasicComboBoxUI.html)    [**NO FRAMES**](http://docs.google.com/BasicComboBoxUI.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | [FIELD](#2et92p0) | [CONSTR](#tyjcwt) | [METHOD](#3dy6vkm) | DETAIL: [FIELD](#2s8eyo1) | [CONSTR](#3as4poj) | [METHOD](#49x2ik5) |

## **javax.swing.plaf.basic**

Class BasicComboBoxUI

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 [javax.swing.plaf.ComponentUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html)  
 [javax.swing.plaf.ComboBoxUI](http://docs.google.com/javax/swing/plaf/ComboBoxUI.html)  
 **javax.swing.plaf.basic.BasicComboBoxUI**

**Direct Known Subclasses:** [MetalComboBoxUI](http://docs.google.com/javax/swing/plaf/metal/MetalComboBoxUI.html)

public class **BasicComboBoxUI**extends [ComboBoxUI](http://docs.google.com/javax/swing/plaf/ComboBoxUI.html)

Basic UI implementation for JComboBox.

The combo box is a compound component which means that it is an agregate of many simpler components. This class creates and manages the listeners on the combo box and the combo box model. These listeners update the user interface in response to changes in the properties and state of the combo box.

All event handling is handled by listener classes created with the createxxxListener() methods and internal classes. You can change the behavior of this class by overriding the createxxxListener() methods and supplying your own event listeners or subclassing from the ones supplied in this class.

For adding specific actions, overide installKeyboardActions to add actions in response to KeyStroke bindings. See the article [Keyboard Bindings in Swing](http://java.sun.com/products/jfc/tsc/special_report/kestrel/keybindings.html) at [*The Swing Connection*](http://java.sun.com/products/jfc/tsc).

| **Nested Class Summary** | |
| --- | --- |
| class | [**BasicComboBoxUI.ComboBoxLayoutManager**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.ComboBoxLayoutManager.html)            This layout manager handles the 'standard' layout of combo boxes. |
| class | [**BasicComboBoxUI.FocusHandler**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.FocusHandler.html)            This listener hides the popup when the focus is lost. |
| class | [**BasicComboBoxUI.ItemHandler**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.ItemHandler.html)            This listener watches for changes to the selection in the combo box. |
| class | [**BasicComboBoxUI.KeyHandler**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.KeyHandler.html)            This listener checks to see if the key event isn't a navigation key. |
| class | [**BasicComboBoxUI.ListDataHandler**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.ListDataHandler.html)            This listener watches for changes in the ComboBoxModel. |
| class | [**BasicComboBoxUI.PropertyChangeHandler**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.PropertyChangeHandler.html)            This listener watches for bound properties that have changed in the combo box. |

| **Field Summary** | |
| --- | --- |
| protected  [JButton](http://docs.google.com/javax/swing/JButton.html) | [**arrowButton**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#arrowButton) |
| protected  [Dimension](http://docs.google.com/java/awt/Dimension.html) | [**cachedMinimumSize**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#cachedMinimumSize) |
| protected  [JComboBox](http://docs.google.com/javax/swing/JComboBox.html) | [**comboBox**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#comboBox) |
| protected  [CellRendererPane](http://docs.google.com/javax/swing/CellRendererPane.html) | [**currentValuePane**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#currentValuePane) |
| protected  [Component](http://docs.google.com/java/awt/Component.html) | [**editor**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#editor) |
| protected  [FocusListener](http://docs.google.com/java/awt/event/FocusListener.html) | [**focusListener**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#focusListener)            This protected field is implementation specific. |
| protected  boolean | [**hasFocus**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#hasFocus)            This protected field is implementation specific. |
| protected  boolean | [**isMinimumSizeDirty**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#isMinimumSizeDirty) |
| protected  [ItemListener](http://docs.google.com/java/awt/event/ItemListener.html) | [**itemListener**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#itemListener)            This protected field is implementation specific. |
| protected  [KeyListener](http://docs.google.com/java/awt/event/KeyListener.html) | [**keyListener**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#keyListener)            This protected field is implementation specific. |
| protected  [JList](http://docs.google.com/javax/swing/JList.html) | [**listBox**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#listBox) |
| protected  [ListDataListener](http://docs.google.com/javax/swing/event/ListDataListener.html) | [**listDataListener**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#listDataListener)            This protected field is implementation specific. |
| protected  [ComboPopup](http://docs.google.com/javax/swing/plaf/basic/ComboPopup.html) | [**popup**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#popup) |
| protected  [KeyListener](http://docs.google.com/java/awt/event/KeyListener.html) | [**popupKeyListener**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#popupKeyListener) |
| protected  [MouseListener](http://docs.google.com/java/awt/event/MouseListener.html) | [**popupMouseListener**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#popupMouseListener) |
| protected  [MouseMotionListener](http://docs.google.com/java/awt/event/MouseMotionListener.html) | [**popupMouseMotionListener**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#popupMouseMotionListener) |
| protected  [PropertyChangeListener](http://docs.google.com/java/beans/PropertyChangeListener.html) | [**propertyChangeListener**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#propertyChangeListener)            This protected field is implementation specific. |

| **Constructor Summary** | |
| --- | --- |
| [**BasicComboBoxUI**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#BasicComboBoxUI())() |

| **Method Summary** | |
| --- | --- |
| void | [**addEditor**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#addEditor())()            This public method is implementation specific and should be private. |
| void | [**configureArrowButton**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#configureArrowButton())()            This public method is implementation specific and should be private. |
| protected  void | [**configureEditor**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#configureEditor())()            This protected method is implementation specific and should be private. |
| protected  [JButton](http://docs.google.com/javax/swing/JButton.html) | [**createArrowButton**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#createArrowButton())()            Creates an button which will be used as the control to show or hide the popup portion of the combo box. |
| protected  [ComboBoxEditor](http://docs.google.com/javax/swing/ComboBoxEditor.html) | [**createEditor**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#createEditor())()            Creates the default editor that will be used in editable combo boxes. |
| protected  [FocusListener](http://docs.google.com/java/awt/event/FocusListener.html) | [**createFocusListener**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#createFocusListener())()            Creates a FocusListener which will be added to the combo box. |
| protected  [ItemListener](http://docs.google.com/java/awt/event/ItemListener.html) | [**createItemListener**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#createItemListener())()            Creates an ItemListener which will be added to the combo box. |
| protected  [KeyListener](http://docs.google.com/java/awt/event/KeyListener.html) | [**createKeyListener**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#createKeyListener())()            Creates a KeyListener which will be added to the combo box. |
| protected  [LayoutManager](http://docs.google.com/java/awt/LayoutManager.html) | [**createLayoutManager**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#createLayoutManager())()            Creates a layout manager for managing the components which make up the combo box. |
| protected  [ListDataListener](http://docs.google.com/javax/swing/event/ListDataListener.html) | [**createListDataListener**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#createListDataListener())()            Creates a list data listener which will be added to the ComboBoxModel. |
| protected  [ComboPopup](http://docs.google.com/javax/swing/plaf/basic/ComboPopup.html) | [**createPopup**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#createPopup())()            Creates the popup portion of the combo box. |
| protected  [PropertyChangeListener](http://docs.google.com/java/beans/PropertyChangeListener.html) | [**createPropertyChangeListener**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#createPropertyChangeListener())()            Creates a PropertyChangeListener which will be added to the combo box. |
| protected  [ListCellRenderer](http://docs.google.com/javax/swing/ListCellRenderer.html) | [**createRenderer**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#createRenderer())()            Creates the default renderer that will be used in a non-editiable combo box. |
| static [ComponentUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html) | [**createUI**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#createUI(javax.swing.JComponent))([JComponent](http://docs.google.com/javax/swing/JComponent.html) c) |
| [Accessible](http://docs.google.com/javax/accessibility/Accessible.html) | [**getAccessibleChild**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#getAccessibleChild(javax.swing.JComponent,%20int))([JComponent](http://docs.google.com/javax/swing/JComponent.html) c, int i)            Returns the ith Accessible child of the object. |
| int | [**getAccessibleChildrenCount**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#getAccessibleChildrenCount(javax.swing.JComponent))([JComponent](http://docs.google.com/javax/swing/JComponent.html) c)            Returns the number of accessible children in the object. |
| int | [**getBaseline**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#getBaseline(javax.swing.JComponent,%20int,%20int))([JComponent](http://docs.google.com/javax/swing/JComponent.html) c, 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/plaf/basic/BasicComboBoxUI.html#getBaselineResizeBehavior(javax.swing.JComponent))([JComponent](http://docs.google.com/javax/swing/JComponent.html) c)            Returns an enum indicating how the baseline of the component changes as the size changes. |
| protected  [Dimension](http://docs.google.com/java/awt/Dimension.html) | [**getDefaultSize**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#getDefaultSize())()            Return the default size of an empty display area of the combo box using the current renderer and font. |
| protected  [Dimension](http://docs.google.com/java/awt/Dimension.html) | [**getDisplaySize**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#getDisplaySize())()            Returns the calculated size of the display area. |
| protected  [Insets](http://docs.google.com/java/awt/Insets.html) | [**getInsets**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#getInsets())()            Gets the insets from the JComboBox. |
| [Dimension](http://docs.google.com/java/awt/Dimension.html) | [**getMaximumSize**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#getMaximumSize(javax.swing.JComponent))([JComponent](http://docs.google.com/javax/swing/JComponent.html) c)            Returns the specified component's maximum size appropriate for the look and feel. |
| [Dimension](http://docs.google.com/java/awt/Dimension.html) | [**getMinimumSize**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#getMinimumSize(javax.swing.JComponent))([JComponent](http://docs.google.com/javax/swing/JComponent.html) c)            The minumum size is the size of the display area plus insets plus the button. |
| [Dimension](http://docs.google.com/java/awt/Dimension.html) | [**getPreferredSize**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#getPreferredSize(javax.swing.JComponent))([JComponent](http://docs.google.com/javax/swing/JComponent.html) c)            Returns the specified component's preferred size appropriate for the look and feel. |
| protected  void | [**installComponents**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#installComponents())()            Creates and initializes the components which make up the aggregate combo box. |
| protected  void | [**installDefaults**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#installDefaults())()            Installs the default colors, default font, default renderer, and default editor into the JComboBox. |
| protected  void | [**installKeyboardActions**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#installKeyboardActions())()            Adds keyboard actions to the JComboBox. |
| protected  void | [**installListeners**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#installListeners())()            Create and install the listeners for the combo box and its model. |
| void | [**installUI**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#installUI(javax.swing.JComponent))([JComponent](http://docs.google.com/javax/swing/JComponent.html) c)            Configures the specified component appropriate for the look and feel. |
| boolean | [**isFocusTraversable**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#isFocusTraversable(javax.swing.JComboBox))([JComboBox](http://docs.google.com/javax/swing/JComboBox.html) c)            Determines if the JComboBox is focus traversable. |
| protected  boolean | [**isNavigationKey**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#isNavigationKey(int))(int keyCode)            Returns whether or not the supplied keyCode maps to a key that is used for navigation. |
| boolean | [**isPopupVisible**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#isPopupVisible(javax.swing.JComboBox))([JComboBox](http://docs.google.com/javax/swing/JComboBox.html) c)            Tells if the popup is visible or not. |
| void | [**paint**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#paint(java.awt.Graphics,%20javax.swing.JComponent))([Graphics](http://docs.google.com/java/awt/Graphics.html) g, [JComponent](http://docs.google.com/javax/swing/JComponent.html) c)            Paints the specified component appropriate for the look and feel. |
| void | [**paintCurrentValue**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#paintCurrentValue(java.awt.Graphics,%20java.awt.Rectangle,%20boolean))([Graphics](http://docs.google.com/java/awt/Graphics.html) g, [Rectangle](http://docs.google.com/java/awt/Rectangle.html) bounds, boolean hasFocus)            Paints the currently selected item. |
| void | [**paintCurrentValueBackground**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#paintCurrentValueBackground(java.awt.Graphics,%20java.awt.Rectangle,%20boolean))([Graphics](http://docs.google.com/java/awt/Graphics.html) g, [Rectangle](http://docs.google.com/java/awt/Rectangle.html) bounds, boolean hasFocus)            Paints the background of the currently selected item. |
| protected  [Rectangle](http://docs.google.com/java/awt/Rectangle.html) | [**rectangleForCurrentValue**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#rectangleForCurrentValue())()            Returns the area that is reserved for drawing the currently selected item. |
| void | [**removeEditor**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#removeEditor())()            This public method is implementation specific and should be private. |
| protected  void | [**selectNextPossibleValue**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#selectNextPossibleValue())()            Selects the next item in the list. |
| protected  void | [**selectPreviousPossibleValue**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#selectPreviousPossibleValue())()            Selects the previous item in the list. |
| void | [**setPopupVisible**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#setPopupVisible(javax.swing.JComboBox,%20boolean))([JComboBox](http://docs.google.com/javax/swing/JComboBox.html) c, boolean v)            Hides the popup. |
| protected  void | [**toggleOpenClose**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#toggleOpenClose())()            Hides the popup if it is showing and shows the popup if it is hidden. |
| void | [**unconfigureArrowButton**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#unconfigureArrowButton())()            This public method is implementation specific and should be private. |
| protected  void | [**unconfigureEditor**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#unconfigureEditor())()            This protected method is implementation specific and should be private. |
| protected  void | [**uninstallComponents**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#uninstallComponents())()            The aggregate components which compise the combo box are unregistered and uninitialized. |
| protected  void | [**uninstallDefaults**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#uninstallDefaults())()            Uninstalls the default colors, default font, default renderer, and default editor into the JComboBox. |
| protected  void | [**uninstallKeyboardActions**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#uninstallKeyboardActions())()            Removes the focus InputMap and ActionMap. |
| protected  void | [**uninstallListeners**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#uninstallListeners())()            Remove the installed listeners from the combo box and its model. |
| void | [**uninstallUI**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#uninstallUI(javax.swing.JComponent))([JComponent](http://docs.google.com/javax/swing/JComponent.html) c)            Reverses configuration which was done on the specified component during installUI. |

| **Methods inherited from class javax.swing.plaf.**[**ComponentUI**](http://docs.google.com/javax/swing/plaf/ComponentUI.html) |
| --- |
| [contains](http://docs.google.com/javax/swing/plaf/ComponentUI.html#contains(javax.swing.JComponent,%20int,%20int)), [update](http://docs.google.com/javax/swing/plaf/ComponentUI.html#update(java.awt.Graphics,%20javax.swing.JComponent)) |

| **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()), [toString](http://docs.google.com/java/lang/Object.html#toString()), [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** |
| --- |

### comboBox

protected [JComboBox](http://docs.google.com/javax/swing/JComboBox.html) **comboBox**

### hasFocus

protected boolean **hasFocus**

This protected field is implementation specific. Do not access directly or override.

### listBox

protected [JList](http://docs.google.com/javax/swing/JList.html) **listBox**

### currentValuePane

protected [CellRendererPane](http://docs.google.com/javax/swing/CellRendererPane.html) **currentValuePane**

### popup

protected [ComboPopup](http://docs.google.com/javax/swing/plaf/basic/ComboPopup.html) **popup**

### editor

protected [Component](http://docs.google.com/java/awt/Component.html) **editor**

### arrowButton

protected [JButton](http://docs.google.com/javax/swing/JButton.html) **arrowButton**

### keyListener

protected [KeyListener](http://docs.google.com/java/awt/event/KeyListener.html) **keyListener**

This protected field is implementation specific. Do not access directly or override. Override the listener construction method instead.

**See Also:**[createKeyListener()](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#createKeyListener())

### focusListener

protected [FocusListener](http://docs.google.com/java/awt/event/FocusListener.html) **focusListener**

This protected field is implementation specific. Do not access directly or override. Override the listener construction method instead.

**See Also:**[createFocusListener()](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#createFocusListener())

### propertyChangeListener

protected [PropertyChangeListener](http://docs.google.com/java/beans/PropertyChangeListener.html) **propertyChangeListener**

This protected field is implementation specific. Do not access directly or override. Override the listener construction method instead.

**See Also:**[createPropertyChangeListener()](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#createPropertyChangeListener())

### itemListener

protected [ItemListener](http://docs.google.com/java/awt/event/ItemListener.html) **itemListener**

This protected field is implementation specific. Do not access directly or override. Override the listener construction method instead.

**See Also:**[createItemListener()](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#createItemListener())

### popupMouseListener

protected [MouseListener](http://docs.google.com/java/awt/event/MouseListener.html) **popupMouseListener**

### popupMouseMotionListener

protected [MouseMotionListener](http://docs.google.com/java/awt/event/MouseMotionListener.html) **popupMouseMotionListener**

### popupKeyListener

protected [KeyListener](http://docs.google.com/java/awt/event/KeyListener.html) **popupKeyListener**

### listDataListener

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

This protected field is implementation specific. Do not access directly or override. Override the listener construction method instead.

**See Also:**[createListDataListener()](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#createListDataListener())

### isMinimumSizeDirty

protected boolean **isMinimumSizeDirty**

### cachedMinimumSize

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

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

### BasicComboBoxUI

public **BasicComboBoxUI**()

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

### createUI

public static [ComponentUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html) **createUI**([JComponent](http://docs.google.com/javax/swing/JComponent.html) c)

### installUI

public void **installUI**([JComponent](http://docs.google.com/javax/swing/JComponent.html) c)

**Description copied from class:** [**ComponentUI**](http://docs.google.com/javax/swing/plaf/ComponentUI.html#installUI(javax.swing.JComponent)) Configures the specified component appropriate for the look and feel. This method is invoked when the ComponentUI instance is being installed as the UI delegate on the specified component. This method should completely configure the component for the look and feel, including the following:

1. Install any default property values for color, fonts, borders, icons, opacity, etc. on the component. Whenever possible, property values initialized by the client program should *not* be overridden.
2. Install a LayoutManager on the component if necessary.
3. Create/add any required sub-components to the component.
4. Create/install event listeners on the component.
5. Create/install a PropertyChangeListener on the component in order to detect and respond to component property changes appropriately.
6. Install keyboard UI (mnemonics, traversal, etc.) on the component.
7. Initialize any appropriate instance data.

**Overrides:**[installUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html#installUI(javax.swing.JComponent)) in class [ComponentUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html) **Parameters:**c - the component where this UI delegate is being installed**See Also:**[ComponentUI.uninstallUI(javax.swing.JComponent)](http://docs.google.com/javax/swing/plaf/ComponentUI.html#uninstallUI(javax.swing.JComponent)), [JComponent.setUI(javax.swing.plaf.ComponentUI)](http://docs.google.com/javax/swing/JComponent.html#setUI(javax.swing.plaf.ComponentUI)), [JComponent.updateUI()](http://docs.google.com/javax/swing/JComponent.html#updateUI())

### uninstallUI

public void **uninstallUI**([JComponent](http://docs.google.com/javax/swing/JComponent.html) c)

**Description copied from class:** [**ComponentUI**](http://docs.google.com/javax/swing/plaf/ComponentUI.html#uninstallUI(javax.swing.JComponent)) Reverses configuration which was done on the specified component during installUI. This method is invoked when this UIComponent instance is being removed as the UI delegate for the specified component. This method should undo the configuration performed in installUI, being careful to leave the JComponent instance in a clean state (no extraneous listeners, look-and-feel-specific property objects, etc.). This should include the following:

1. Remove any UI-set borders from the component.
2. Remove any UI-set layout managers on the component.
3. Remove any UI-added sub-components from the component.
4. Remove any UI-added event/property listeners from the component.
5. Remove any UI-installed keyboard UI from the component.
6. Nullify any allocated instance data objects to allow for GC.

**Overrides:**[uninstallUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html#uninstallUI(javax.swing.JComponent)) in class [ComponentUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html) **Parameters:**c - the component from which this UI delegate is being removed; this argument is often ignored, but might be used if the UI object is stateless and shared by multiple components**See Also:**[ComponentUI.installUI(javax.swing.JComponent)](http://docs.google.com/javax/swing/plaf/ComponentUI.html#installUI(javax.swing.JComponent)), [JComponent.updateUI()](http://docs.google.com/javax/swing/JComponent.html#updateUI())

### installDefaults

protected void **installDefaults**()

Installs the default colors, default font, default renderer, and default editor into the JComboBox.

### installListeners

protected void **installListeners**()

Create and install the listeners for the combo box and its model. This method is called when the UI is installed.

### uninstallDefaults

protected void **uninstallDefaults**()

Uninstalls the default colors, default font, default renderer, and default editor into the JComboBox.

### uninstallListeners

protected void **uninstallListeners**()

Remove the installed listeners from the combo box and its model. The number and types of listeners removed and in this method should be the same that was added in installListeners

### createPopup

protected [ComboPopup](http://docs.google.com/javax/swing/plaf/basic/ComboPopup.html) **createPopup**()

Creates the popup portion of the combo box.

**Returns:**an instance of ComboPopup**See Also:**[ComboPopup](http://docs.google.com/javax/swing/plaf/basic/ComboPopup.html)

### createKeyListener

protected [KeyListener](http://docs.google.com/java/awt/event/KeyListener.html) **createKeyListener**()

Creates a KeyListener which will be added to the combo box. If this method returns null then it will not be added to the combo box.

**Returns:**an instance KeyListener or null

### createFocusListener

protected [FocusListener](http://docs.google.com/java/awt/event/FocusListener.html) **createFocusListener**()

Creates a FocusListener which will be added to the combo box. If this method returns null then it will not be added to the combo box.

**Returns:**an instance of a FocusListener or null

### createListDataListener

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

Creates a list data listener which will be added to the ComboBoxModel. If this method returns null then it will not be added to the combo box model.

**Returns:**an instance of a ListDataListener or null

### createItemListener

protected [ItemListener](http://docs.google.com/java/awt/event/ItemListener.html) **createItemListener**()

Creates an ItemListener which will be added to the combo box. If this method returns null then it will not be added to the combo box.

Subclasses may override this method to return instances of their own ItemEvent handlers.

**Returns:**an instance of an ItemListener or null

### createPropertyChangeListener

protected [PropertyChangeListener](http://docs.google.com/java/beans/PropertyChangeListener.html) **createPropertyChangeListener**()

Creates a PropertyChangeListener which will be added to the combo box. If this method returns null then it will not be added to the combo box.

**Returns:**an instance of a PropertyChangeListener or null

### createLayoutManager

protected [LayoutManager](http://docs.google.com/java/awt/LayoutManager.html) **createLayoutManager**()

Creates a layout manager for managing the components which make up the combo box.

**Returns:**an instance of a layout manager

### createRenderer

protected [ListCellRenderer](http://docs.google.com/javax/swing/ListCellRenderer.html) **createRenderer**()

Creates the default renderer that will be used in a non-editiable combo box. A default renderer will used only if a renderer has not been explicitly set with setRenderer.

**Returns:**a ListCellRender used for the combo box**See Also:**[JComboBox.setRenderer(javax.swing.ListCellRenderer)](http://docs.google.com/javax/swing/JComboBox.html#setRenderer(javax.swing.ListCellRenderer))

### createEditor

protected [ComboBoxEditor](http://docs.google.com/javax/swing/ComboBoxEditor.html) **createEditor**()

Creates the default editor that will be used in editable combo boxes. A default editor will be used only if an editor has not been explicitly set with setEditor.

**Returns:**a ComboBoxEditor used for the combo box**See Also:**[JComboBox.setEditor(javax.swing.ComboBoxEditor)](http://docs.google.com/javax/swing/JComboBox.html#setEditor(javax.swing.ComboBoxEditor))

### installComponents

protected void **installComponents**()

Creates and initializes the components which make up the aggregate combo box. This method is called as part of the UI installation process.

### uninstallComponents

protected void **uninstallComponents**()

The aggregate components which compise the combo box are unregistered and uninitialized. This method is called as part of the UI uninstallation process.

### addEditor

public void **addEditor**()

This public method is implementation specific and should be private. do not call or override. To implement a specific editor create a custom ComboBoxEditor

**See Also:**[createEditor()](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#createEditor()), [JComboBox.setEditor(javax.swing.ComboBoxEditor)](http://docs.google.com/javax/swing/JComboBox.html#setEditor(javax.swing.ComboBoxEditor)), [ComboBoxEditor](http://docs.google.com/javax/swing/ComboBoxEditor.html)

### removeEditor

public void **removeEditor**()

This public method is implementation specific and should be private. do not call or override.

**See Also:**[addEditor()](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#addEditor())

### configureEditor

protected void **configureEditor**()

This protected method is implementation specific and should be private. do not call or override.

**See Also:**[addEditor()](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#addEditor())

### unconfigureEditor

protected void **unconfigureEditor**()

This protected method is implementation specific and should be private. Do not call or override.

**See Also:**[addEditor()](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#addEditor())

### configureArrowButton

public void **configureArrowButton**()

This public method is implementation specific and should be private. Do not call or override.

**See Also:**[createArrowButton()](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#createArrowButton())

### unconfigureArrowButton

public void **unconfigureArrowButton**()

This public method is implementation specific and should be private. Do not call or override.

**See Also:**[createArrowButton()](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#createArrowButton())

### createArrowButton

protected [JButton](http://docs.google.com/javax/swing/JButton.html) **createArrowButton**()

Creates an button which will be used as the control to show or hide the popup portion of the combo box.

**Returns:**a button which represents the popup control

### isPopupVisible

public boolean **isPopupVisible**([JComboBox](http://docs.google.com/javax/swing/JComboBox.html) c)

Tells if the popup is visible or not.

**Specified by:**[isPopupVisible](http://docs.google.com/javax/swing/plaf/ComboBoxUI.html#isPopupVisible(javax.swing.JComboBox)) in class [ComboBoxUI](http://docs.google.com/javax/swing/plaf/ComboBoxUI.html)

### setPopupVisible

public void **setPopupVisible**([JComboBox](http://docs.google.com/javax/swing/JComboBox.html) c,  
 boolean v)

Hides the popup.

**Specified by:**[setPopupVisible](http://docs.google.com/javax/swing/plaf/ComboBoxUI.html#setPopupVisible(javax.swing.JComboBox,%20boolean)) in class [ComboBoxUI](http://docs.google.com/javax/swing/plaf/ComboBoxUI.html)

### isFocusTraversable

public boolean **isFocusTraversable**([JComboBox](http://docs.google.com/javax/swing/JComboBox.html) c)

Determines if the JComboBox is focus traversable. If the JComboBox is editable this returns false, otherwise it returns true.

**Specified by:**[isFocusTraversable](http://docs.google.com/javax/swing/plaf/ComboBoxUI.html#isFocusTraversable(javax.swing.JComboBox)) in class [ComboBoxUI](http://docs.google.com/javax/swing/plaf/ComboBoxUI.html)

### paint

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

**Description copied from class:** [**ComponentUI**](http://docs.google.com/javax/swing/plaf/ComponentUI.html#paint(java.awt.Graphics,%20javax.swing.JComponent)) Paints the specified component appropriate for the look and feel. This method is invoked from the ComponentUI.update method when the specified component is being painted. Subclasses should override this method and use the specified Graphics object to render the content of the component.

**Overrides:**[paint](http://docs.google.com/javax/swing/plaf/ComponentUI.html#paint(java.awt.Graphics,%20javax.swing.JComponent)) in class [ComponentUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html) **Parameters:**g - the Graphics context in which to paintc - the component being painted; this argument is often ignored, but might be used if the UI object is stateless and shared by multiple components**See Also:**[ComponentUI.update(java.awt.Graphics, javax.swing.JComponent)](http://docs.google.com/javax/swing/plaf/ComponentUI.html#update(java.awt.Graphics,%20javax.swing.JComponent))

### getPreferredSize

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

**Description copied from class:** [**ComponentUI**](http://docs.google.com/javax/swing/plaf/ComponentUI.html#getPreferredSize(javax.swing.JComponent)) Returns the specified component's preferred size appropriate for the look and feel. If null is returned, the preferred size will be calculated by the component's layout manager instead (this is the preferred approach for any component with a specific layout manager installed). The default implementation of this method returns null.

**Overrides:**[getPreferredSize](http://docs.google.com/javax/swing/plaf/ComponentUI.html#getPreferredSize(javax.swing.JComponent)) in class [ComponentUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html) **Parameters:**c - the component whose preferred size is being queried; this argument is often ignored, but might be used if the UI object is stateless and shared by multiple components**See Also:**[JComponent.getPreferredSize()](http://docs.google.com/javax/swing/JComponent.html#getPreferredSize()), [LayoutManager.preferredLayoutSize(java.awt.Container)](http://docs.google.com/java/awt/LayoutManager.html#preferredLayoutSize(java.awt.Container))

### getMinimumSize

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

The minumum size is the size of the display area plus insets plus the button.

**Overrides:**[getMinimumSize](http://docs.google.com/javax/swing/plaf/ComponentUI.html#getMinimumSize(javax.swing.JComponent)) in class [ComponentUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html) **Parameters:**c - the component whose minimum size is being queried; this argument is often ignored, but might be used if the UI object is stateless and shared by multiple components **Returns:**a Dimension object or null**See Also:**[JComponent.getMinimumSize()](http://docs.google.com/javax/swing/JComponent.html#getMinimumSize()), [LayoutManager.minimumLayoutSize(java.awt.Container)](http://docs.google.com/java/awt/LayoutManager.html#minimumLayoutSize(java.awt.Container)), [ComponentUI.getPreferredSize(javax.swing.JComponent)](http://docs.google.com/javax/swing/plaf/ComponentUI.html#getPreferredSize(javax.swing.JComponent))

### getMaximumSize

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

**Description copied from class:** [**ComponentUI**](http://docs.google.com/javax/swing/plaf/ComponentUI.html#getMaximumSize(javax.swing.JComponent)) Returns the specified component's maximum size appropriate for the look and feel. If null is returned, the maximum size will be calculated by the component's layout manager instead (this is the preferred approach for any component with a specific layout manager installed). The default implementation of this method invokes getPreferredSize and returns that value.

**Overrides:**[getMaximumSize](http://docs.google.com/javax/swing/plaf/ComponentUI.html#getMaximumSize(javax.swing.JComponent)) in class [ComponentUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html) **Parameters:**c - the component whose maximum size is being queried; this argument is often ignored, but might be used if the UI object is stateless and shared by multiple components **Returns:**a Dimension object or null**See Also:**[JComponent.getMaximumSize()](http://docs.google.com/javax/swing/JComponent.html#getMaximumSize()), [LayoutManager2.maximumLayoutSize(java.awt.Container)](http://docs.google.com/java/awt/LayoutManager2.html#maximumLayoutSize(java.awt.Container))

### getBaseline

public int **getBaseline**([JComponent](http://docs.google.com/javax/swing/JComponent.html) c,  
 int width,  
 int height)

Returns the baseline.

**Overrides:**[getBaseline](http://docs.google.com/javax/swing/plaf/ComponentUI.html#getBaseline(javax.swing.JComponent,%20int,%20int)) in class [ComponentUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html) **Parameters:**c - JComponent baseline is being requested forwidth - the width to get the baseline forheight - the height to get the baseline for **Returns:**baseline or a value < 0 indicating there is no reasonable baseline **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if c is null [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if width or height is < 0**Since:** 1.6 **See Also:**[JComponent.getBaseline(int, int)](http://docs.google.com/javax/swing/JComponent.html#getBaseline(int,%20int))

### getBaselineResizeBehavior

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

Returns an enum indicating how the baseline of the component changes as the size changes.

**Overrides:**[getBaselineResizeBehavior](http://docs.google.com/javax/swing/plaf/ComponentUI.html#getBaselineResizeBehavior(javax.swing.JComponent)) in class [ComponentUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html) **Parameters:**c - JComponent to return baseline resize behavior for **Returns:**an enum indicating how the baseline changes as the component size changes **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if c is null**Since:** 1.6 **See Also:**[JComponent.getBaseline(int, int)](http://docs.google.com/javax/swing/JComponent.html#getBaseline(int,%20int))

### getAccessibleChildrenCount

public int **getAccessibleChildrenCount**([JComponent](http://docs.google.com/javax/swing/JComponent.html) c)

**Description copied from class:** [**ComponentUI**](http://docs.google.com/javax/swing/plaf/ComponentUI.html#getAccessibleChildrenCount(javax.swing.JComponent)) Returns the number of accessible children in the object. If all of the children of this object implement Accessible, this method should return the number of children of this object. UIs might wish to override this if they present areas on the screen that can be viewed as components, but actual components are not used for presenting those areas. Note: As of v1.3, it is recommended that developers call Component.AccessibleAWTComponent.getAccessibleChildrenCount() instead of this method.

**Overrides:**[getAccessibleChildrenCount](http://docs.google.com/javax/swing/plaf/ComponentUI.html#getAccessibleChildrenCount(javax.swing.JComponent)) in class [ComponentUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html) **Returns:**the number of accessible children in the object**See Also:**[ComponentUI.getAccessibleChild(javax.swing.JComponent, int)](http://docs.google.com/javax/swing/plaf/ComponentUI.html#getAccessibleChild(javax.swing.JComponent,%20int))

### getAccessibleChild

public [Accessible](http://docs.google.com/javax/accessibility/Accessible.html) **getAccessibleChild**([JComponent](http://docs.google.com/javax/swing/JComponent.html) c,  
 int i)

**Description copied from class:** [**ComponentUI**](http://docs.google.com/javax/swing/plaf/ComponentUI.html#getAccessibleChild(javax.swing.JComponent,%20int)) Returns the ith Accessible child of the object. UIs might need to override this if they present areas on the screen that can be viewed as components, but actual components are not used for presenting those areas.

Note: As of v1.3, it is recommended that developers call Component.AccessibleAWTComponent.getAccessibleChild() instead of this method.

**Overrides:**[getAccessibleChild](http://docs.google.com/javax/swing/plaf/ComponentUI.html#getAccessibleChild(javax.swing.JComponent,%20int)) in class [ComponentUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html) i - zero-based index of child **Returns:**the ith Accessible child of the object**See Also:**[ComponentUI.getAccessibleChildrenCount(javax.swing.JComponent)](http://docs.google.com/javax/swing/plaf/ComponentUI.html#getAccessibleChildrenCount(javax.swing.JComponent))

### isNavigationKey

protected boolean **isNavigationKey**(int keyCode)

Returns whether or not the supplied keyCode maps to a key that is used for navigation. This is used for optimizing key input by only passing non- navigation keys to the type-ahead mechanism. Subclasses should override this if they change the navigation keys.

### selectNextPossibleValue

protected void **selectNextPossibleValue**()

Selects the next item in the list. It won't change the selection if the currently selected item is already the last item.

### selectPreviousPossibleValue

protected void **selectPreviousPossibleValue**()

Selects the previous item in the list. It won't change the selection if the currently selected item is already the first item.

### toggleOpenClose

protected void **toggleOpenClose**()

Hides the popup if it is showing and shows the popup if it is hidden.

### rectangleForCurrentValue

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

Returns the area that is reserved for drawing the currently selected item.

### getInsets

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

Gets the insets from the JComboBox.

### paintCurrentValue

public void **paintCurrentValue**([Graphics](http://docs.google.com/java/awt/Graphics.html) g,  
 [Rectangle](http://docs.google.com/java/awt/Rectangle.html) bounds,  
 boolean hasFocus)

Paints the currently selected item.

### paintCurrentValueBackground

public void **paintCurrentValueBackground**([Graphics](http://docs.google.com/java/awt/Graphics.html) g,  
 [Rectangle](http://docs.google.com/java/awt/Rectangle.html) bounds,  
 boolean hasFocus)

Paints the background of the currently selected item.

### getDefaultSize

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

Return the default size of an empty display area of the combo box using the current renderer and font.

**Returns:**the size of an empty display area**See Also:**[getDisplaySize()](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.html#getDisplaySize())

### getDisplaySize

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

Returns the calculated size of the display area. The display area is the portion of the combo box in which the selected item is displayed. This method will use the prototype display value if it has been set.

For combo boxes with a non trivial number of items, it is recommended to use a prototype display value to significantly speed up the display size calculation.

**Returns:**the size of the display area calculated from the combo box items**See Also:**[JComboBox.setPrototypeDisplayValue(java.lang.Object)](http://docs.google.com/javax/swing/JComboBox.html#setPrototypeDisplayValue(java.lang.Object))

### installKeyboardActions

protected void **installKeyboardActions**()

Adds keyboard actions to the JComboBox. Actions on enter and esc are already supplied. Add more actions as you need them.

### uninstallKeyboardActions

protected void **uninstallKeyboardActions**()

Removes the focus InputMap and ActionMap.

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/BasicComboBoxUI.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/plaf/basic/BasicComboBoxRenderer.UIResource.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/plaf/basic/BasicComboBoxUI.ComboBoxLayoutManager.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/plaf/basic/BasicComboBoxUI.html)    [**NO FRAMES**](http://docs.google.com/BasicComboBoxUI.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | [FIELD](#2et92p0) | [CONSTR](#tyjcwt) | [METHOD](#3dy6vkm) | DETAIL: [FIELD](#2s8eyo1) | [CONSTR](#3as4poj) | [METHOD](#49x2ik5) |

[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).