| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/JTextComponent.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/text/InternationalFormatter.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/text/JTextComponent.AccessibleJTextComponent.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/text/JTextComponent.html)    [**NO FRAMES**](http://docs.google.com/JTextComponent.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | [FIELD](#1t3h5sf) | [CONSTR](#3rdcrjn) | [METHOD](#26in1rg) | DETAIL: [FIELD](#2jxsxqh) | [CONSTR](#1y810tw) | [METHOD](#2xcytpi) |

## **javax.swing.text**

Class JTextComponent

[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](http://docs.google.com/javax/swing/JComponent.html)  
 **javax.swing.text.JTextComponent**

**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), [Accessible](http://docs.google.com/javax/accessibility/Accessible.html), [Scrollable](http://docs.google.com/javax/swing/Scrollable.html) **Direct Known Subclasses:** [JEditorPane](http://docs.google.com/javax/swing/JEditorPane.html), [JTextArea](http://docs.google.com/javax/swing/JTextArea.html), [JTextField](http://docs.google.com/javax/swing/JTextField.html)

public abstract class **JTextComponent**extends [JComponent](http://docs.google.com/javax/swing/JComponent.html)implements [Scrollable](http://docs.google.com/javax/swing/Scrollable.html), [Accessible](http://docs.google.com/javax/accessibility/Accessible.html)

JTextComponent is the base class for swing text components. It tries to be compatible with the java.awt.TextComponent class where it can reasonably do so. Also provided are other services for additional flexibility (beyond the pluggable UI and bean support). You can find information on how to use the functionality this class provides in [General Rules for Using Text Components](http://java.sun.com/docs/books/tutorial/uiswing/components/generaltext.html), a section in *The Java Tutorial.*

**Caret Changes** The caret is a pluggable object in swing text components. Notification of changes to the caret position and the selection are sent to implementations of the CaretListener interface that have been registered with the text component. The UI will install a default caret unless a customized caret has been set.

By default the caret tracks all the document changes performed on the Event Dispatching Thread and updates it's position accordingly if an insertion occurs before or at the caret position or a removal occurs before the caret position. DefaultCaret tries to make itself visible which may lead to scrolling of a text component within JScrollPane. The default caret behavior can be changed by the [DefaultCaret.setUpdatePolicy(int)](http://docs.google.com/javax/swing/text/DefaultCaret.html#setUpdatePolicy(int)) method.

**Note**: Non-editable text components also have a caret though it may not be painted.

**Commands** Text components provide a number of commands that can be used to manipulate the component. This is essentially the way that the component expresses its capabilities. These are expressed in terms of the swing Action interface, using the TextAction implementation. The set of commands supported by the text component can be found with the [getActions()](http://docs.google.com/javax/swing/text/JTextComponent.html#getActions()) method. These actions can be bound to key events, fired from buttons, etc.

**Text Input** The text components support flexible and internationalized text input, using keymaps and the input method framework, while maintaining compatibility with the AWT listener model.

A [Keymap](http://docs.google.com/javax/swing/text/Keymap.html) lets an application bind key strokes to actions. In order to allow keymaps to be shared across multiple text components, they can use actions that extend TextAction. TextAction can determine which JTextComponent most recently has or had focus and therefore is the subject of the action (In the case that the ActionEvent sent to the action doesn't contain the target text component as its source).

The [input method framework](http://docs.google.com/technotes/guides/imf/spec.html) lets text components interact with input methods, separate software components that preprocess events to let users enter thousands of different characters using keyboards with far fewer keys. JTextComponent is an *active client* of the framework, so it implements the preferred user interface for interacting with input methods. As a consequence, some key events do not reach the text component because they are handled by an input method, and some text input reaches the text component as committed text within an [InputMethodEvent](http://docs.google.com/java/awt/event/InputMethodEvent.html) instead of as a key event. The complete text input is the combination of the characters in keyTyped key events and committed text in input method events.

The AWT listener model lets applications attach event listeners to components in order to bind events to actions. Swing encourages the use of keymaps instead of listeners, but maintains compatibility with listeners by giving the listeners a chance to steal an event by consuming it.

Keyboard event and input method events are handled in the following stages, with each stage capable of consuming the event:

| Stage | KeyEvent | InputMethodEvent |
| --- | --- | --- |
| 1. | input methods | (generated here) |
| 2. | focus manager |  |
| 3. | registered key listeners | registered input method listeners |
| 4. |  | input method handling in JTextComponent |
| 5. | keymap handling using the current keymap | |
| 6. | keyboard handling in JComponent (e.g. accelerators, component navigation, etc.) |  |

To maintain compatibility with applications that listen to key events but are not aware of input method events, the input method handling in stage 4 provides a compatibility mode for components that do not process input method events. For these components, the committed text is converted to keyTyped key events and processed in the key event pipeline starting at stage 3 instead of in the input method event pipeline.

By default the component will create a keymap (named **DEFAULT\_KEYMAP**) that is shared by all JTextComponent instances as the default keymap. Typically a look-and-feel implementation will install a different keymap that resolves to the default keymap for those bindings not found in the different keymap. The minimal bindings include:

* inserting content into the editor for the printable keys.
* removing content with the backspace and del keys.
* caret movement forward and backward

**Model/View Split** The text components have a model-view split. A text component pulls together the objects used to represent the model, view, and controller. The text document model may be shared by other views which act as observers of the model (e.g. a document may be shared by multiple components).



The model is defined by the [Document](http://docs.google.com/javax/swing/text/Document.html) interface. This is intended to provide a flexible text storage mechanism that tracks change during edits and can be extended to more sophisticated models. The model interfaces are meant to capture the capabilities of expression given by SGML, a system used to express a wide variety of content. Each modification to the document causes notification of the details of the change to be sent to all observers in the form of a [DocumentEvent](http://docs.google.com/javax/swing/event/DocumentEvent.html) which allows the views to stay up to date with the model. This event is sent to observers that have implemented the [DocumentListener](http://docs.google.com/javax/swing/event/DocumentListener.html) interface and registered interest with the model being observed.

**Location Information** The capability of determining the location of text in the view is provided. There are two methods, [modelToView(int)](http://docs.google.com/javax/swing/text/JTextComponent.html#modelToView(int)) and [viewToModel(java.awt.Point)](http://docs.google.com/javax/swing/text/JTextComponent.html#viewToModel(java.awt.Point)) for determining this information.

**Undo/Redo support** Support for an edit history mechanism is provided to allow undo/redo operations. The text component does not itself provide the history buffer by default, but does provide the UndoableEdit records that can be used in conjunction with a history buffer to provide the undo/redo support. The support is provided by the Document model, which allows one to attach UndoableEditListener implementations.

**Thread Safety** The swing text components provide some support of thread safe operations. Because of the high level of configurability of the text components, it is possible to circumvent the protection provided. The protection primarily comes from the model, so the documentation of AbstractDocument describes the assumptions of the protection provided. The methods that are safe to call asynchronously are marked with comments.

**Newlines** For a discussion on how newlines are handled, see [DefaultEditorKit](http://docs.google.com/DefaultEditorKit.html).

**Printing support** Several [print](http://docs.google.com/javax/swing/text/JTextComponent.html#print()) methods are provided for basic document printing. If more advanced printing is needed, use the [getPrintable(java.text.MessageFormat, java.text.MessageFormat)](http://docs.google.com/javax/swing/text/JTextComponent.html#getPrintable(java.text.MessageFormat,%20java.text.MessageFormat)) method.

**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:**[Document](http://docs.google.com/javax/swing/text/Document.html), [DocumentEvent](http://docs.google.com/javax/swing/event/DocumentEvent.html), [DocumentListener](http://docs.google.com/javax/swing/event/DocumentListener.html), [Caret](http://docs.google.com/javax/swing/text/Caret.html), [CaretEvent](http://docs.google.com/javax/swing/event/CaretEvent.html), [CaretListener](http://docs.google.com/javax/swing/event/CaretListener.html), [TextUI](http://docs.google.com/javax/swing/plaf/TextUI.html), [View](http://docs.google.com/javax/swing/text/View.html), [ViewFactory](http://docs.google.com/javax/swing/text/ViewFactory.html)

| **Nested Class Summary** | |
| --- | --- |
| class | [**JTextComponent.AccessibleJTextComponent**](http://docs.google.com/javax/swing/text/JTextComponent.AccessibleJTextComponent.html)            This class implements accessibility support for the JTextComponent class. |
| static class | [**JTextComponent.DropLocation**](http://docs.google.com/javax/swing/text/JTextComponent.DropLocation.html)            Represents a drop location for JTextComponents. |
| static class | [**JTextComponent.KeyBinding**](http://docs.google.com/javax/swing/text/JTextComponent.KeyBinding.html)            Binding record for creating key bindings. |

| **Nested classes/interfaces inherited from class javax.swing.**[**JComponent**](http://docs.google.com/javax/swing/JComponent.html) |
| --- |
| [JComponent.AccessibleJComponent](http://docs.google.com/javax/swing/JComponent.AccessibleJComponent.html) |

| **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** | |
| --- | --- |
| static [String](http://docs.google.com/java/lang/String.html) | [**DEFAULT\_KEYMAP**](http://docs.google.com/javax/swing/text/JTextComponent.html#DEFAULT_KEYMAP)            The default keymap that will be shared by all JTextComponent instances unless they have had a different keymap set. |
| static [String](http://docs.google.com/java/lang/String.html) | [**FOCUS\_ACCELERATOR\_KEY**](http://docs.google.com/javax/swing/text/JTextComponent.html#FOCUS_ACCELERATOR_KEY)            The bound property name for the focus accelerator. |

| **Fields inherited from class javax.swing.**[**JComponent**](http://docs.google.com/javax/swing/JComponent.html) |
| --- |
| [accessibleContext](http://docs.google.com/javax/swing/JComponent.html#accessibleContext), [listenerList](http://docs.google.com/javax/swing/JComponent.html#listenerList), [TOOL\_TIP\_TEXT\_KEY](http://docs.google.com/javax/swing/JComponent.html#TOOL_TIP_TEXT_KEY), [ui](http://docs.google.com/javax/swing/JComponent.html#ui), [UNDEFINED\_CONDITION](http://docs.google.com/javax/swing/JComponent.html#UNDEFINED_CONDITION), [WHEN\_ANCESTOR\_OF\_FOCUSED\_COMPONENT](http://docs.google.com/javax/swing/JComponent.html#WHEN_ANCESTOR_OF_FOCUSED_COMPONENT), [WHEN\_FOCUSED](http://docs.google.com/javax/swing/JComponent.html#WHEN_FOCUSED), [WHEN\_IN\_FOCUSED\_WINDOW](http://docs.google.com/javax/swing/JComponent.html#WHEN_IN_FOCUSED_WINDOW) |

| **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** | |
| --- | --- |
| [**JTextComponent**](http://docs.google.com/javax/swing/text/JTextComponent.html#JTextComponent())()            Creates a new JTextComponent. |

| **Method Summary** | |
| --- | --- |
| void | [**addCaretListener**](http://docs.google.com/javax/swing/text/JTextComponent.html#addCaretListener(javax.swing.event.CaretListener))([CaretListener](http://docs.google.com/javax/swing/event/CaretListener.html) listener)            Adds a caret listener for notification of any changes to the caret. |
| void | [**addInputMethodListener**](http://docs.google.com/javax/swing/text/JTextComponent.html#addInputMethodListener(java.awt.event.InputMethodListener))([InputMethodListener](http://docs.google.com/java/awt/event/InputMethodListener.html) l)            Adds the specified input method listener to receive input method events from this component. |
| static [Keymap](http://docs.google.com/javax/swing/text/Keymap.html) | [**addKeymap**](http://docs.google.com/javax/swing/text/JTextComponent.html#addKeymap(java.lang.String,%20javax.swing.text.Keymap))([String](http://docs.google.com/java/lang/String.html) nm, [Keymap](http://docs.google.com/javax/swing/text/Keymap.html) parent)            Adds a new keymap into the keymap hierarchy. |
| void | [**copy**](http://docs.google.com/javax/swing/text/JTextComponent.html#copy())()            Transfers the currently selected range in the associated text model to the system clipboard, leaving the contents in the text model. |
| void | [**cut**](http://docs.google.com/javax/swing/text/JTextComponent.html#cut())()            Transfers the currently selected range in the associated text model to the system clipboard, removing the contents from the model. |
| protected  void | [**fireCaretUpdate**](http://docs.google.com/javax/swing/text/JTextComponent.html#fireCaretUpdate(javax.swing.event.CaretEvent))([CaretEvent](http://docs.google.com/javax/swing/event/CaretEvent.html) e)            Notifies all listeners that have registered interest for notification on this event type. |
| [AccessibleContext](http://docs.google.com/javax/accessibility/AccessibleContext.html) | [**getAccessibleContext**](http://docs.google.com/javax/swing/text/JTextComponent.html#getAccessibleContext())()            Gets the AccessibleContext associated with this JTextComponent. |
| [Action](http://docs.google.com/javax/swing/Action.html)[] | [**getActions**](http://docs.google.com/javax/swing/text/JTextComponent.html#getActions())()            Fetches the command list for the editor. |
| [Caret](http://docs.google.com/javax/swing/text/Caret.html) | [**getCaret**](http://docs.google.com/javax/swing/text/JTextComponent.html#getCaret())()            Fetches the caret that allows text-oriented navigation over the view. |
| [Color](http://docs.google.com/java/awt/Color.html) | [**getCaretColor**](http://docs.google.com/javax/swing/text/JTextComponent.html#getCaretColor())()            Fetches the current color used to render the caret. |
| [CaretListener](http://docs.google.com/javax/swing/event/CaretListener.html)[] | [**getCaretListeners**](http://docs.google.com/javax/swing/text/JTextComponent.html#getCaretListeners())()            Returns an array of all the caret listeners registered on this text component. |
| int | [**getCaretPosition**](http://docs.google.com/javax/swing/text/JTextComponent.html#getCaretPosition())()            Returns the position of the text insertion caret for the text component. |
| [Color](http://docs.google.com/java/awt/Color.html) | [**getDisabledTextColor**](http://docs.google.com/javax/swing/text/JTextComponent.html#getDisabledTextColor())()            Fetches the current color used to render the disabled text. |
| [Document](http://docs.google.com/javax/swing/text/Document.html) | [**getDocument**](http://docs.google.com/javax/swing/text/JTextComponent.html#getDocument())()            Fetches the model associated with the editor. |
| boolean | [**getDragEnabled**](http://docs.google.com/javax/swing/text/JTextComponent.html#getDragEnabled())()            Returns whether or not automatic drag handling is enabled. |
| [JTextComponent.DropLocation](http://docs.google.com/javax/swing/text/JTextComponent.DropLocation.html) | [**getDropLocation**](http://docs.google.com/javax/swing/text/JTextComponent.html#getDropLocation())()            Returns the location that this component should visually indicate as the drop location during a DnD operation over the component, or null if no location is to currently be shown. |
| [DropMode](http://docs.google.com/javax/swing/DropMode.html) | [**getDropMode**](http://docs.google.com/javax/swing/text/JTextComponent.html#getDropMode())()            Returns the drop mode for this component. |
| char | [**getFocusAccelerator**](http://docs.google.com/javax/swing/text/JTextComponent.html#getFocusAccelerator())()            Returns the key accelerator that will cause the receiving text component to get the focus. |
| [Highlighter](http://docs.google.com/javax/swing/text/Highlighter.html) | [**getHighlighter**](http://docs.google.com/javax/swing/text/JTextComponent.html#getHighlighter())()            Fetches the object responsible for making highlights. |
| [InputMethodRequests](http://docs.google.com/java/awt/im/InputMethodRequests.html) | [**getInputMethodRequests**](http://docs.google.com/javax/swing/text/JTextComponent.html#getInputMethodRequests())()            Gets the input method request handler which supports requests from input methods for this component. |
| [Keymap](http://docs.google.com/javax/swing/text/Keymap.html) | [**getKeymap**](http://docs.google.com/javax/swing/text/JTextComponent.html#getKeymap())()            Fetches the keymap currently active in this text component. |
| static [Keymap](http://docs.google.com/javax/swing/text/Keymap.html) | [**getKeymap**](http://docs.google.com/javax/swing/text/JTextComponent.html#getKeymap(java.lang.String))([String](http://docs.google.com/java/lang/String.html) nm)            Fetches a named keymap previously added to the document. |
| [Insets](http://docs.google.com/java/awt/Insets.html) | [**getMargin**](http://docs.google.com/javax/swing/text/JTextComponent.html#getMargin())()            Returns the margin between the text component's border and its text. |
| [NavigationFilter](http://docs.google.com/javax/swing/text/NavigationFilter.html) | [**getNavigationFilter**](http://docs.google.com/javax/swing/text/JTextComponent.html#getNavigationFilter())()            Returns the NavigationFilter. |
| [Dimension](http://docs.google.com/java/awt/Dimension.html) | [**getPreferredScrollableViewportSize**](http://docs.google.com/javax/swing/text/JTextComponent.html#getPreferredScrollableViewportSize())()            Returns the preferred size of the viewport for a view component. |
| [Printable](http://docs.google.com/java/awt/print/Printable.html) | [**getPrintable**](http://docs.google.com/javax/swing/text/JTextComponent.html#getPrintable(java.text.MessageFormat,%20java.text.MessageFormat))([MessageFormat](http://docs.google.com/java/text/MessageFormat.html) headerFormat, [MessageFormat](http://docs.google.com/java/text/MessageFormat.html) footerFormat)            Returns a Printable to use for printing the content of this JTextComponent. |
| int | [**getScrollableBlockIncrement**](http://docs.google.com/javax/swing/text/JTextComponent.html#getScrollableBlockIncrement(java.awt.Rectangle,%20int,%20int))([Rectangle](http://docs.google.com/java/awt/Rectangle.html) visibleRect, int orientation, int direction)            Components that display logical rows or columns should compute the scroll increment that will completely expose one block of rows or columns, depending on the value of orientation. |
| boolean | [**getScrollableTracksViewportHeight**](http://docs.google.com/javax/swing/text/JTextComponent.html#getScrollableTracksViewportHeight())()            Returns true if a viewport should always force the height of this Scrollable to match the height of the viewport. |
| boolean | [**getScrollableTracksViewportWidth**](http://docs.google.com/javax/swing/text/JTextComponent.html#getScrollableTracksViewportWidth())()            Returns true if a viewport should always force the width of this Scrollable to match the width of the viewport. |
| int | [**getScrollableUnitIncrement**](http://docs.google.com/javax/swing/text/JTextComponent.html#getScrollableUnitIncrement(java.awt.Rectangle,%20int,%20int))([Rectangle](http://docs.google.com/java/awt/Rectangle.html) visibleRect, int orientation, int direction)            Components that display logical rows or columns should compute the scroll increment that will completely expose one new row or column, depending on the value of orientation. |
| [String](http://docs.google.com/java/lang/String.html) | [**getSelectedText**](http://docs.google.com/javax/swing/text/JTextComponent.html#getSelectedText())()            Returns the selected text contained in this TextComponent. |
| [Color](http://docs.google.com/java/awt/Color.html) | [**getSelectedTextColor**](http://docs.google.com/javax/swing/text/JTextComponent.html#getSelectedTextColor())()            Fetches the current color used to render the selected text. |
| [Color](http://docs.google.com/java/awt/Color.html) | [**getSelectionColor**](http://docs.google.com/javax/swing/text/JTextComponent.html#getSelectionColor())()            Fetches the current color used to render the selection. |
| int | [**getSelectionEnd**](http://docs.google.com/javax/swing/text/JTextComponent.html#getSelectionEnd())()            Returns the selected text's end position. |
| int | [**getSelectionStart**](http://docs.google.com/javax/swing/text/JTextComponent.html#getSelectionStart())()            Returns the selected text's start position. |
| [String](http://docs.google.com/java/lang/String.html) | [**getText**](http://docs.google.com/javax/swing/text/JTextComponent.html#getText())()            Returns the text contained in this TextComponent. |
| [String](http://docs.google.com/java/lang/String.html) | [**getText**](http://docs.google.com/javax/swing/text/JTextComponent.html#getText(int,%20int))(int offs, int len)            Fetches a portion of the text represented by the component. |
| [String](http://docs.google.com/java/lang/String.html) | [**getToolTipText**](http://docs.google.com/javax/swing/text/JTextComponent.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. |
| [TextUI](http://docs.google.com/javax/swing/plaf/TextUI.html) | [**getUI**](http://docs.google.com/javax/swing/text/JTextComponent.html#getUI())()            Fetches the user-interface factory for this text-oriented editor. |
| boolean | [**isEditable**](http://docs.google.com/javax/swing/text/JTextComponent.html#isEditable())()            Returns the boolean indicating whether this TextComponent is editable or not. |
| static void | [**loadKeymap**](http://docs.google.com/javax/swing/text/JTextComponent.html#loadKeymap(javax.swing.text.Keymap,%20javax.swing.text.JTextComponent.KeyBinding%5B%5D,%20javax.swing.Action%5B%5D))([Keymap](http://docs.google.com/javax/swing/text/Keymap.html) map, [JTextComponent.KeyBinding](http://docs.google.com/javax/swing/text/JTextComponent.KeyBinding.html)[] bindings, [Action](http://docs.google.com/javax/swing/Action.html)[] actions)             Loads a keymap with a bunch of bindings. |
| [Rectangle](http://docs.google.com/java/awt/Rectangle.html) | [**modelToView**](http://docs.google.com/javax/swing/text/JTextComponent.html#modelToView(int))(int pos)            Converts the given location in the model to a place in the view coordinate system. |
| void | [**moveCaretPosition**](http://docs.google.com/javax/swing/text/JTextComponent.html#moveCaretPosition(int))(int pos)            Moves the caret to a new position, leaving behind a mark defined by the last time setCaretPosition was called. |
| protected  [String](http://docs.google.com/java/lang/String.html) | [**paramString**](http://docs.google.com/javax/swing/text/JTextComponent.html#paramString())()            Returns a string representation of this JTextComponent. |
| void | [**paste**](http://docs.google.com/javax/swing/text/JTextComponent.html#paste())()            Transfers the contents of the system clipboard into the associated text model. |
| boolean | [**print**](http://docs.google.com/javax/swing/text/JTextComponent.html#print())()            A convenience print method that displays a print dialog, and then prints this JTextComponent in *interactive* mode with no header or footer text. |
| boolean | [**print**](http://docs.google.com/javax/swing/text/JTextComponent.html#print(java.text.MessageFormat,%20java.text.MessageFormat))([MessageFormat](http://docs.google.com/java/text/MessageFormat.html) headerFormat, [MessageFormat](http://docs.google.com/java/text/MessageFormat.html) footerFormat)            A convenience print method that displays a print dialog, and then prints this JTextComponent in *interactive* mode with the specified header and footer text. |
| boolean | [**print**](http://docs.google.com/javax/swing/text/JTextComponent.html#print(java.text.MessageFormat,%20java.text.MessageFormat,%20boolean,%20javax.print.PrintService,%20javax.print.attribute.PrintRequestAttributeSet,%20boolean))([MessageFormat](http://docs.google.com/java/text/MessageFormat.html) headerFormat, [MessageFormat](http://docs.google.com/java/text/MessageFormat.html) footerFormat, boolean showPrintDialog, [PrintService](http://docs.google.com/javax/print/PrintService.html) service, [PrintRequestAttributeSet](http://docs.google.com/javax/print/attribute/PrintRequestAttributeSet.html) attributes, boolean interactive)            Prints the content of this JTextComponent. |
| protected  void | [**processInputMethodEvent**](http://docs.google.com/javax/swing/text/JTextComponent.html#processInputMethodEvent(java.awt.event.InputMethodEvent))([InputMethodEvent](http://docs.google.com/java/awt/event/InputMethodEvent.html) e)            Processes input method events occurring on this component by dispatching them to any registered InputMethodListener objects. |
| void | [**read**](http://docs.google.com/javax/swing/text/JTextComponent.html#read(java.io.Reader,%20java.lang.Object))([Reader](http://docs.google.com/java/io/Reader.html) in, [Object](http://docs.google.com/java/lang/Object.html) desc)            Initializes from a stream. |
| void | [**removeCaretListener**](http://docs.google.com/javax/swing/text/JTextComponent.html#removeCaretListener(javax.swing.event.CaretListener))([CaretListener](http://docs.google.com/javax/swing/event/CaretListener.html) listener)            Removes a caret listener. |
| static [Keymap](http://docs.google.com/javax/swing/text/Keymap.html) | [**removeKeymap**](http://docs.google.com/javax/swing/text/JTextComponent.html#removeKeymap(java.lang.String))([String](http://docs.google.com/java/lang/String.html) nm)            Removes a named keymap previously added to the document. |
| void | [**removeNotify**](http://docs.google.com/javax/swing/text/JTextComponent.html#removeNotify())()            Notifies this component that it no longer has a parent component. |
| void | [**replaceSelection**](http://docs.google.com/javax/swing/text/JTextComponent.html#replaceSelection(java.lang.String))([String](http://docs.google.com/java/lang/String.html) content)            Replaces the currently selected content with new content represented by the given string. |
| void | [**select**](http://docs.google.com/javax/swing/text/JTextComponent.html#select(int,%20int))(int selectionStart, int selectionEnd)            Selects the text between the specified start and end positions. |
| void | [**selectAll**](http://docs.google.com/javax/swing/text/JTextComponent.html#selectAll())()            Selects all the text in the TextComponent. |
| void | [**setCaret**](http://docs.google.com/javax/swing/text/JTextComponent.html#setCaret(javax.swing.text.Caret))([Caret](http://docs.google.com/javax/swing/text/Caret.html) c)            Sets the caret to be used. |
| void | [**setCaretColor**](http://docs.google.com/javax/swing/text/JTextComponent.html#setCaretColor(java.awt.Color))([Color](http://docs.google.com/java/awt/Color.html) c)            Sets the current color used to render the caret. |
| void | [**setCaretPosition**](http://docs.google.com/javax/swing/text/JTextComponent.html#setCaretPosition(int))(int position)            Sets the position of the text insertion caret for the TextComponent. |
| void | [**setComponentOrientation**](http://docs.google.com/javax/swing/text/JTextComponent.html#setComponentOrientation(java.awt.ComponentOrientation))([ComponentOrientation](http://docs.google.com/java/awt/ComponentOrientation.html) o)            Sets the language-sensitive orientation that is to be used to order the elements or text within this component. |
| void | [**setDisabledTextColor**](http://docs.google.com/javax/swing/text/JTextComponent.html#setDisabledTextColor(java.awt.Color))([Color](http://docs.google.com/java/awt/Color.html) c)            Sets the current color used to render the disabled text. |
| void | [**setDocument**](http://docs.google.com/javax/swing/text/JTextComponent.html#setDocument(javax.swing.text.Document))([Document](http://docs.google.com/javax/swing/text/Document.html) doc)            Associates the editor with a text document. |
| void | [**setDragEnabled**](http://docs.google.com/javax/swing/text/JTextComponent.html#setDragEnabled(boolean))(boolean b)            Turns on or off automatic drag handling. |
| void | [**setDropMode**](http://docs.google.com/javax/swing/text/JTextComponent.html#setDropMode(javax.swing.DropMode))([DropMode](http://docs.google.com/javax/swing/DropMode.html) dropMode)            Sets the drop mode for this component. |
| void | [**setEditable**](http://docs.google.com/javax/swing/text/JTextComponent.html#setEditable(boolean))(boolean b)            Sets the specified boolean to indicate whether or not this TextComponent should be editable. |
| void | [**setFocusAccelerator**](http://docs.google.com/javax/swing/text/JTextComponent.html#setFocusAccelerator(char))(char aKey)            Sets the key accelerator that will cause the receiving text component to get the focus. |
| void | [**setHighlighter**](http://docs.google.com/javax/swing/text/JTextComponent.html#setHighlighter(javax.swing.text.Highlighter))([Highlighter](http://docs.google.com/javax/swing/text/Highlighter.html) h)            Sets the highlighter to be used. |
| void | [**setKeymap**](http://docs.google.com/javax/swing/text/JTextComponent.html#setKeymap(javax.swing.text.Keymap))([Keymap](http://docs.google.com/javax/swing/text/Keymap.html) map)            Sets the keymap to use for binding events to actions. |
| void | [**setMargin**](http://docs.google.com/javax/swing/text/JTextComponent.html#setMargin(java.awt.Insets))([Insets](http://docs.google.com/java/awt/Insets.html) m)            Sets margin space between the text component's border and its text. |
| void | [**setNavigationFilter**](http://docs.google.com/javax/swing/text/JTextComponent.html#setNavigationFilter(javax.swing.text.NavigationFilter))([NavigationFilter](http://docs.google.com/javax/swing/text/NavigationFilter.html) filter)            Sets the NavigationFilter. |
| void | [**setSelectedTextColor**](http://docs.google.com/javax/swing/text/JTextComponent.html#setSelectedTextColor(java.awt.Color))([Color](http://docs.google.com/java/awt/Color.html) c)            Sets the current color used to render the selected text. |
| void | [**setSelectionColor**](http://docs.google.com/javax/swing/text/JTextComponent.html#setSelectionColor(java.awt.Color))([Color](http://docs.google.com/java/awt/Color.html) c)            Sets the current color used to render the selection. |
| void | [**setSelectionEnd**](http://docs.google.com/javax/swing/text/JTextComponent.html#setSelectionEnd(int))(int selectionEnd)            Sets the selection end to the specified position. |
| void | [**setSelectionStart**](http://docs.google.com/javax/swing/text/JTextComponent.html#setSelectionStart(int))(int selectionStart)            Sets the selection start to the specified position. |
| void | [**setText**](http://docs.google.com/javax/swing/text/JTextComponent.html#setText(java.lang.String))([String](http://docs.google.com/java/lang/String.html) t)            Sets the text of this TextComponent to the specified text. |
| void | [**setUI**](http://docs.google.com/javax/swing/text/JTextComponent.html#setUI(javax.swing.plaf.TextUI))([TextUI](http://docs.google.com/javax/swing/plaf/TextUI.html) ui)            Sets the user-interface factory for this text-oriented editor. |
| void | [**updateUI**](http://docs.google.com/javax/swing/text/JTextComponent.html#updateUI())()            Reloads the pluggable UI. |
| int | [**viewToModel**](http://docs.google.com/javax/swing/text/JTextComponent.html#viewToModel(java.awt.Point))([Point](http://docs.google.com/java/awt/Point.html) pt)            Converts the given place in the view coordinate system to the nearest representative location in the model. |
| void | [**write**](http://docs.google.com/javax/swing/text/JTextComponent.html#write(java.io.Writer))([Writer](http://docs.google.com/java/io/Writer.html) out)            Stores the contents of the model into the given stream. |

| **Methods inherited from class javax.swing.**[**JComponent**](http://docs.google.com/javax/swing/JComponent.html) |
| --- |
| [addAncestorListener](http://docs.google.com/javax/swing/JComponent.html#addAncestorListener(javax.swing.event.AncestorListener)), [addNotify](http://docs.google.com/javax/swing/JComponent.html#addNotify()), [addVetoableChangeListener](http://docs.google.com/javax/swing/JComponent.html#addVetoableChangeListener(java.beans.VetoableChangeListener)), [computeVisibleRect](http://docs.google.com/javax/swing/JComponent.html#computeVisibleRect(java.awt.Rectangle)), [contains](http://docs.google.com/javax/swing/JComponent.html#contains(int,%20int)), [createToolTip](http://docs.google.com/javax/swing/JComponent.html#createToolTip()), [disable](http://docs.google.com/javax/swing/JComponent.html#disable()), [enable](http://docs.google.com/javax/swing/JComponent.html#enable()), [firePropertyChange](http://docs.google.com/javax/swing/JComponent.html#firePropertyChange(java.lang.String,%20boolean,%20boolean)), [firePropertyChange](http://docs.google.com/javax/swing/JComponent.html#firePropertyChange(java.lang.String,%20char,%20char)), [firePropertyChange](http://docs.google.com/javax/swing/JComponent.html#firePropertyChange(java.lang.String,%20int,%20int)), [fireVetoableChange](http://docs.google.com/javax/swing/JComponent.html#fireVetoableChange(java.lang.String,%20java.lang.Object,%20java.lang.Object)), [getActionForKeyStroke](http://docs.google.com/javax/swing/JComponent.html#getActionForKeyStroke(javax.swing.KeyStroke)), [getActionMap](http://docs.google.com/javax/swing/JComponent.html#getActionMap()), [getAlignmentX](http://docs.google.com/javax/swing/JComponent.html#getAlignmentX()), [getAlignmentY](http://docs.google.com/javax/swing/JComponent.html#getAlignmentY()), [getAncestorListeners](http://docs.google.com/javax/swing/JComponent.html#getAncestorListeners()), [getAutoscrolls](http://docs.google.com/javax/swing/JComponent.html#getAutoscrolls()), [getBaseline](http://docs.google.com/javax/swing/JComponent.html#getBaseline(int,%20int)), [getBaselineResizeBehavior](http://docs.google.com/javax/swing/JComponent.html#getBaselineResizeBehavior()), [getBorder](http://docs.google.com/javax/swing/JComponent.html#getBorder()), [getBounds](http://docs.google.com/javax/swing/JComponent.html#getBounds(java.awt.Rectangle)), [getClientProperty](http://docs.google.com/javax/swing/JComponent.html#getClientProperty(java.lang.Object)), [getComponentGraphics](http://docs.google.com/javax/swing/JComponent.html#getComponentGraphics(java.awt.Graphics)), [getComponentPopupMenu](http://docs.google.com/javax/swing/JComponent.html#getComponentPopupMenu()), [getConditionForKeyStroke](http://docs.google.com/javax/swing/JComponent.html#getConditionForKeyStroke(javax.swing.KeyStroke)), [getDebugGraphicsOptions](http://docs.google.com/javax/swing/JComponent.html#getDebugGraphicsOptions()), [getDefaultLocale](http://docs.google.com/javax/swing/JComponent.html#getDefaultLocale()), [getFontMetrics](http://docs.google.com/javax/swing/JComponent.html#getFontMetrics(java.awt.Font)), [getGraphics](http://docs.google.com/javax/swing/JComponent.html#getGraphics()), [getHeight](http://docs.google.com/javax/swing/JComponent.html#getHeight()), [getInheritsPopupMenu](http://docs.google.com/javax/swing/JComponent.html#getInheritsPopupMenu()), [getInputMap](http://docs.google.com/javax/swing/JComponent.html#getInputMap()), [getInputMap](http://docs.google.com/javax/swing/JComponent.html#getInputMap(int)), [getInputVerifier](http://docs.google.com/javax/swing/JComponent.html#getInputVerifier()), [getInsets](http://docs.google.com/javax/swing/JComponent.html#getInsets()), [getInsets](http://docs.google.com/javax/swing/JComponent.html#getInsets(java.awt.Insets)), [getListeners](http://docs.google.com/javax/swing/JComponent.html#getListeners(java.lang.Class)), [getLocation](http://docs.google.com/javax/swing/JComponent.html#getLocation(java.awt.Point)), [getMaximumSize](http://docs.google.com/javax/swing/JComponent.html#getMaximumSize()), [getMinimumSize](http://docs.google.com/javax/swing/JComponent.html#getMinimumSize()), [getNextFocusableComponent](http://docs.google.com/javax/swing/JComponent.html#getNextFocusableComponent()), [getPopupLocation](http://docs.google.com/javax/swing/JComponent.html#getPopupLocation(java.awt.event.MouseEvent)), [getPreferredSize](http://docs.google.com/javax/swing/JComponent.html#getPreferredSize()), [getRegisteredKeyStrokes](http://docs.google.com/javax/swing/JComponent.html#getRegisteredKeyStrokes()), [getRootPane](http://docs.google.com/javax/swing/JComponent.html#getRootPane()), [getSize](http://docs.google.com/javax/swing/JComponent.html#getSize(java.awt.Dimension)), [getToolTipLocation](http://docs.google.com/javax/swing/JComponent.html#getToolTipLocation(java.awt.event.MouseEvent)), [getToolTipText](http://docs.google.com/javax/swing/JComponent.html#getToolTipText()), [getTopLevelAncestor](http://docs.google.com/javax/swing/JComponent.html#getTopLevelAncestor()), [getTransferHandler](http://docs.google.com/javax/swing/JComponent.html#getTransferHandler()), [getUIClassID](http://docs.google.com/javax/swing/JComponent.html#getUIClassID()), [getVerifyInputWhenFocusTarget](http://docs.google.com/javax/swing/JComponent.html#getVerifyInputWhenFocusTarget()), [getVetoableChangeListeners](http://docs.google.com/javax/swing/JComponent.html#getVetoableChangeListeners()), [getVisibleRect](http://docs.google.com/javax/swing/JComponent.html#getVisibleRect()), [getWidth](http://docs.google.com/javax/swing/JComponent.html#getWidth()), [getX](http://docs.google.com/javax/swing/JComponent.html#getX()), [getY](http://docs.google.com/javax/swing/JComponent.html#getY()), [grabFocus](http://docs.google.com/javax/swing/JComponent.html#grabFocus()), [isDoubleBuffered](http://docs.google.com/javax/swing/JComponent.html#isDoubleBuffered()), [isLightweightComponent](http://docs.google.com/javax/swing/JComponent.html#isLightweightComponent(java.awt.Component)), [isManagingFocus](http://docs.google.com/javax/swing/JComponent.html#isManagingFocus()), [isOpaque](http://docs.google.com/javax/swing/JComponent.html#isOpaque()), [isOptimizedDrawingEnabled](http://docs.google.com/javax/swing/JComponent.html#isOptimizedDrawingEnabled()), [isPaintingForPrint](http://docs.google.com/javax/swing/JComponent.html#isPaintingForPrint()), [isPaintingTile](http://docs.google.com/javax/swing/JComponent.html#isPaintingTile()), [isRequestFocusEnabled](http://docs.google.com/javax/swing/JComponent.html#isRequestFocusEnabled()), [isValidateRoot](http://docs.google.com/javax/swing/JComponent.html#isValidateRoot()), [paint](http://docs.google.com/javax/swing/JComponent.html#paint(java.awt.Graphics)), [paintBorder](http://docs.google.com/javax/swing/JComponent.html#paintBorder(java.awt.Graphics)), [paintChildren](http://docs.google.com/javax/swing/JComponent.html#paintChildren(java.awt.Graphics)), [paintComponent](http://docs.google.com/javax/swing/JComponent.html#paintComponent(java.awt.Graphics)), [paintImmediately](http://docs.google.com/javax/swing/JComponent.html#paintImmediately(int,%20int,%20int,%20int)), [paintImmediately](http://docs.google.com/javax/swing/JComponent.html#paintImmediately(java.awt.Rectangle)), [print](http://docs.google.com/javax/swing/JComponent.html#print(java.awt.Graphics)), [printAll](http://docs.google.com/javax/swing/JComponent.html#printAll(java.awt.Graphics)), [printBorder](http://docs.google.com/javax/swing/JComponent.html#printBorder(java.awt.Graphics)), [printChildren](http://docs.google.com/javax/swing/JComponent.html#printChildren(java.awt.Graphics)), [printComponent](http://docs.google.com/javax/swing/JComponent.html#printComponent(java.awt.Graphics)), [processComponentKeyEvent](http://docs.google.com/javax/swing/JComponent.html#processComponentKeyEvent(java.awt.event.KeyEvent)), [processKeyBinding](http://docs.google.com/javax/swing/JComponent.html#processKeyBinding(javax.swing.KeyStroke,%20java.awt.event.KeyEvent,%20int,%20boolean)), [processKeyEvent](http://docs.google.com/javax/swing/JComponent.html#processKeyEvent(java.awt.event.KeyEvent)), [processMouseEvent](http://docs.google.com/javax/swing/JComponent.html#processMouseEvent(java.awt.event.MouseEvent)), [processMouseMotionEvent](http://docs.google.com/javax/swing/JComponent.html#processMouseMotionEvent(java.awt.event.MouseEvent)), [putClientProperty](http://docs.google.com/javax/swing/JComponent.html#putClientProperty(java.lang.Object,%20java.lang.Object)), [registerKeyboardAction](http://docs.google.com/javax/swing/JComponent.html#registerKeyboardAction(java.awt.event.ActionListener,%20javax.swing.KeyStroke,%20int)), [registerKeyboardAction](http://docs.google.com/javax/swing/JComponent.html#registerKeyboardAction(java.awt.event.ActionListener,%20java.lang.String,%20javax.swing.KeyStroke,%20int)), [removeAncestorListener](http://docs.google.com/javax/swing/JComponent.html#removeAncestorListener(javax.swing.event.AncestorListener)), [removeVetoableChangeListener](http://docs.google.com/javax/swing/JComponent.html#removeVetoableChangeListener(java.beans.VetoableChangeListener)), [repaint](http://docs.google.com/javax/swing/JComponent.html#repaint(long,%20int,%20int,%20int,%20int)), [repaint](http://docs.google.com/javax/swing/JComponent.html#repaint(java.awt.Rectangle)), [requestDefaultFocus](http://docs.google.com/javax/swing/JComponent.html#requestDefaultFocus()), [requestFocus](http://docs.google.com/javax/swing/JComponent.html#requestFocus()), [requestFocus](http://docs.google.com/javax/swing/JComponent.html#requestFocus(boolean)), [requestFocusInWindow](http://docs.google.com/javax/swing/JComponent.html#requestFocusInWindow()), [requestFocusInWindow](http://docs.google.com/javax/swing/JComponent.html#requestFocusInWindow(boolean)), [resetKeyboardActions](http://docs.google.com/javax/swing/JComponent.html#resetKeyboardActions()), [reshape](http://docs.google.com/javax/swing/JComponent.html#reshape(int,%20int,%20int,%20int)), [revalidate](http://docs.google.com/javax/swing/JComponent.html#revalidate()), [scrollRectToVisible](http://docs.google.com/javax/swing/JComponent.html#scrollRectToVisible(java.awt.Rectangle)), [setActionMap](http://docs.google.com/javax/swing/JComponent.html#setActionMap(javax.swing.ActionMap)), [setAlignmentX](http://docs.google.com/javax/swing/JComponent.html#setAlignmentX(float)), [setAlignmentY](http://docs.google.com/javax/swing/JComponent.html#setAlignmentY(float)), [setAutoscrolls](http://docs.google.com/javax/swing/JComponent.html#setAutoscrolls(boolean)), [setBackground](http://docs.google.com/javax/swing/JComponent.html#setBackground(java.awt.Color)), [setBorder](http://docs.google.com/javax/swing/JComponent.html#setBorder(javax.swing.border.Border)), [setComponentPopupMenu](http://docs.google.com/javax/swing/JComponent.html#setComponentPopupMenu(javax.swing.JPopupMenu)), [setDebugGraphicsOptions](http://docs.google.com/javax/swing/JComponent.html#setDebugGraphicsOptions(int)), [setDefaultLocale](http://docs.google.com/javax/swing/JComponent.html#setDefaultLocale(java.util.Locale)), [setDoubleBuffered](http://docs.google.com/javax/swing/JComponent.html#setDoubleBuffered(boolean)), [setEnabled](http://docs.google.com/javax/swing/JComponent.html#setEnabled(boolean)), [setFocusTraversalKeys](http://docs.google.com/javax/swing/JComponent.html#setFocusTraversalKeys(int,%20java.util.Set)), [setFont](http://docs.google.com/javax/swing/JComponent.html#setFont(java.awt.Font)), [setForeground](http://docs.google.com/javax/swing/JComponent.html#setForeground(java.awt.Color)), [setInheritsPopupMenu](http://docs.google.com/javax/swing/JComponent.html#setInheritsPopupMenu(boolean)), [setInputMap](http://docs.google.com/javax/swing/JComponent.html#setInputMap(int,%20javax.swing.InputMap)), [setInputVerifier](http://docs.google.com/javax/swing/JComponent.html#setInputVerifier(javax.swing.InputVerifier)), [setMaximumSize](http://docs.google.com/javax/swing/JComponent.html#setMaximumSize(java.awt.Dimension)), [setMinimumSize](http://docs.google.com/javax/swing/JComponent.html#setMinimumSize(java.awt.Dimension)), [setNextFocusableComponent](http://docs.google.com/javax/swing/JComponent.html#setNextFocusableComponent(java.awt.Component)), [setOpaque](http://docs.google.com/javax/swing/JComponent.html#setOpaque(boolean)), [setPreferredSize](http://docs.google.com/javax/swing/JComponent.html#setPreferredSize(java.awt.Dimension)), [setRequestFocusEnabled](http://docs.google.com/javax/swing/JComponent.html#setRequestFocusEnabled(boolean)), [setToolTipText](http://docs.google.com/javax/swing/JComponent.html#setToolTipText(java.lang.String)), [setTransferHandler](http://docs.google.com/javax/swing/JComponent.html#setTransferHandler(javax.swing.TransferHandler)), [setUI](http://docs.google.com/javax/swing/JComponent.html#setUI(javax.swing.plaf.ComponentUI)), [setVerifyInputWhenFocusTarget](http://docs.google.com/javax/swing/JComponent.html#setVerifyInputWhenFocusTarget(boolean)), [setVisible](http://docs.google.com/javax/swing/JComponent.html#setVisible(boolean)), [unregisterKeyboardAction](http://docs.google.com/javax/swing/JComponent.html#unregisterKeyboardAction(javax.swing.KeyStroke)), [update](http://docs.google.com/javax/swing/JComponent.html#update(java.awt.Graphics)) |

| **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)), [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()), [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)), [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)), [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** |
| --- |

### FOCUS\_ACCELERATOR\_KEY

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

The bound property name for the focus accelerator.

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

### DEFAULT\_KEYMAP

public static final [String](http://docs.google.com/java/lang/String.html) **DEFAULT\_KEYMAP**

The default keymap that will be shared by all JTextComponent instances unless they have had a different keymap set.

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

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

### JTextComponent

public **JTextComponent**()

Creates a new JTextComponent. Listeners for caret events are established, and the pluggable UI installed. The component is marked as editable. No layout manager is used, because layout is managed by the view subsystem of text. The document model is set to null.

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

### getUI

public [TextUI](http://docs.google.com/javax/swing/plaf/TextUI.html) **getUI**()

Fetches the user-interface factory for this text-oriented editor.

**Returns:**the factory

### setUI

public void **setUI**([TextUI](http://docs.google.com/javax/swing/plaf/TextUI.html) ui)

Sets the user-interface factory for this text-oriented editor.

**Parameters:**ui - the factory

### updateUI

public void **updateUI**()

Reloads the pluggable UI. The key used to fetch the new interface is getUIClassID(). The type of the UI is TextUI. invalidate is called after setting the UI.

**Overrides:**[updateUI](http://docs.google.com/javax/swing/JComponent.html#updateUI()) in class [JComponent](http://docs.google.com/javax/swing/JComponent.html) **See Also:**[JComponent.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))

### addCaretListener

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

Adds a caret listener for notification of any changes to the caret.

**Parameters:**listener - the listener to be added**See Also:**[CaretEvent](http://docs.google.com/javax/swing/event/CaretEvent.html)

### removeCaretListener

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

Removes a caret listener.

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

### getCaretListeners

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

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

**Returns:**all of this component's CaretListeners or an empty array if no caret listeners are currently registered**Since:** 1.4 **See Also:**[addCaretListener(javax.swing.event.CaretListener)](http://docs.google.com/javax/swing/text/JTextComponent.html#addCaretListener(javax.swing.event.CaretListener)), [removeCaretListener(javax.swing.event.CaretListener)](http://docs.google.com/javax/swing/text/JTextComponent.html#removeCaretListener(javax.swing.event.CaretListener))

### fireCaretUpdate

protected void **fireCaretUpdate**([CaretEvent](http://docs.google.com/javax/swing/event/CaretEvent.html) e)

Notifies all listeners that have registered interest for notification on this event type. The event instance is lazily created using the parameters passed into the fire method. The listener list is processed in a last-to-first manner.

**Parameters:**e - the event**See Also:**[EventListenerList](http://docs.google.com/javax/swing/event/EventListenerList.html)

### setDocument

public void **setDocument**([Document](http://docs.google.com/javax/swing/text/Document.html) doc)

Associates the editor with a text document. The currently registered factory is used to build a view for the document, which gets displayed by the editor after revalidation. A PropertyChange event ("document") is propagated to each listener.

**Parameters:**doc - the document to display/edit**See Also:**[getDocument()](http://docs.google.com/javax/swing/text/JTextComponent.html#getDocument())

### getDocument

public [Document](http://docs.google.com/javax/swing/text/Document.html) **getDocument**()

Fetches the model associated with the editor. This is primarily for the UI to get at the minimal amount of state required to be a text editor. Subclasses will return the actual type of the model which will typically be something that extends Document.

**Returns:**the model

### setComponentOrientation

public void **setComponentOrientation**([ComponentOrientation](http://docs.google.com/java/awt/ComponentOrientation.html) o)

**Description copied from class:** [**Component**](http://docs.google.com/java/awt/Component.html#setComponentOrientation(java.awt.ComponentOrientation)) Sets the language-sensitive orientation that is to be used to order the elements or text within this component. Language-sensitive LayoutManager and Component subclasses will use this property to determine how to lay out and draw components.

At construction time, a component's orientation is set to ComponentOrientation.UNKNOWN, indicating that it has not been specified explicitly. The UNKNOWN orientation behaves the same as ComponentOrientation.LEFT\_TO\_RIGHT.

To set the orientation of a single component, use this method. To set the orientation of an entire component hierarchy, use [applyComponentOrientation](http://docs.google.com/java/awt/Component.html#applyComponentOrientation(java.awt.ComponentOrientation)).

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

### getActions

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

Fetches the command list for the editor. This is the list of commands supported by the plugged-in UI augmented by the collection of commands that the editor itself supports. These are useful for binding to events, such as in a keymap.

**Returns:**the command list

### setMargin

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

Sets margin space between the text component's border and its text. The text component's default Border object will use this value to create the proper margin. However, if a non-default border is set on the text component, it is that Border object's responsibility to create the appropriate margin space (else this property will effectively be ignored). This causes a redraw of the component. A PropertyChange event ("margin") is sent to all listeners.

**Parameters:**m - the space between the border and the text

### getMargin

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

Returns the margin between the text component's border and its text.

**Returns:**the margin

### setNavigationFilter

public void **setNavigationFilter**([NavigationFilter](http://docs.google.com/javax/swing/text/NavigationFilter.html) filter)

Sets the NavigationFilter. NavigationFilter is used by DefaultCaret and the default cursor movement actions as a way to restrict the cursor movement.

**Since:** 1.4

### getNavigationFilter

public [NavigationFilter](http://docs.google.com/javax/swing/text/NavigationFilter.html) **getNavigationFilter**()

Returns the NavigationFilter. NavigationFilter is used by DefaultCaret and the default cursor movement actions as a way to restrict the cursor movement. A null return value implies the cursor movement and selection should not be restricted.

**Returns:**the NavigationFilter**Since:** 1.4

### getCaret

public [Caret](http://docs.google.com/javax/swing/text/Caret.html) **getCaret**()

Fetches the caret that allows text-oriented navigation over the view.

**Returns:**the caret

### setCaret

public void **setCaret**([Caret](http://docs.google.com/javax/swing/text/Caret.html) c)

Sets the caret to be used. By default this will be set by the UI that gets installed. This can be changed to a custom caret if desired. Setting the caret results in a PropertyChange event ("caret") being fired.

**Parameters:**c - the caret**See Also:**[getCaret()](http://docs.google.com/javax/swing/text/JTextComponent.html#getCaret())

### getHighlighter

public [Highlighter](http://docs.google.com/javax/swing/text/Highlighter.html) **getHighlighter**()

Fetches the object responsible for making highlights.

**Returns:**the highlighter

### setHighlighter

public void **setHighlighter**([Highlighter](http://docs.google.com/javax/swing/text/Highlighter.html) h)

Sets the highlighter to be used. By default this will be set by the UI that gets installed. This can be changed to a custom highlighter if desired. The highlighter can be set to null to disable it. A PropertyChange event ("highlighter") is fired when a new highlighter is installed.

**Parameters:**h - the highlighter**See Also:**[getHighlighter()](http://docs.google.com/javax/swing/text/JTextComponent.html#getHighlighter())

### setKeymap

public void **setKeymap**([Keymap](http://docs.google.com/javax/swing/text/Keymap.html) map)

Sets the keymap to use for binding events to actions. Setting to null effectively disables keyboard input. A PropertyChange event ("keymap") is fired when a new keymap is installed.

**Parameters:**map - the keymap**See Also:**[getKeymap()](http://docs.google.com/javax/swing/text/JTextComponent.html#getKeymap())

### setDragEnabled

public void **setDragEnabled**(boolean b)

Turns on or off automatic drag handling. In order to enable automatic drag handling, this property should be set to true, and the component's TransferHandler needs to be non-null. The default value of the dragEnabled property is false.

The job of honoring this property, and recognizing a user drag gesture, lies with the look and feel implementation, and in particular, the component's TextUI. When automatic drag handling is enabled, most look and feels (including those that subclass BasicLookAndFeel) begin a drag and drop operation whenever the user presses the mouse button over a selection and then moves the mouse a few pixels. Setting this property to true can therefore have a subtle effect on how selections behave.

If a look and feel is used that ignores this property, you can still begin a drag and drop operation by calling exportAsDrag on the component's TransferHandler.

**Parameters:**b - whether or not to enable automatic drag handling **Throws:** [HeadlessException](http://docs.google.com/java/awt/HeadlessException.html) - if b is true and GraphicsEnvironment.isHeadless() returns true**Since:** 1.4 **See Also:**[GraphicsEnvironment.isHeadless()](http://docs.google.com/java/awt/GraphicsEnvironment.html#isHeadless()), [getDragEnabled()](http://docs.google.com/javax/swing/text/JTextComponent.html#getDragEnabled()), [JComponent.setTransferHandler(javax.swing.TransferHandler)](http://docs.google.com/javax/swing/JComponent.html#setTransferHandler(javax.swing.TransferHandler)), [TransferHandler](http://docs.google.com/javax/swing/TransferHandler.html)

### getDragEnabled

public boolean **getDragEnabled**()

Returns whether or not automatic drag handling is enabled.

**Returns:**the value of the dragEnabled property**Since:** 1.4 **See Also:**[setDragEnabled(boolean)](http://docs.google.com/javax/swing/text/JTextComponent.html#setDragEnabled(boolean))

### setDropMode

public final void **setDropMode**([DropMode](http://docs.google.com/javax/swing/DropMode.html) dropMode)

Sets the drop mode for this component. For backward compatibility, the default for this property is DropMode.USE\_SELECTION. Usage of DropMode.INSERT is recommended, however, for an improved user experience. It offers similar behavior of dropping between text locations, but does so without affecting the actual text selection and caret location.

JTextComponents support the following drop modes:

* DropMode.USE\_SELECTION
* DropMode.INSERT

The drop mode is only meaningful if this component has a TransferHandler that accepts drops.

**Parameters:**dropMode - the drop mode to use **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the drop mode is unsupported or null**Since:** 1.6 **See Also:**[getDropMode()](http://docs.google.com/javax/swing/text/JTextComponent.html#getDropMode()), [getDropLocation()](http://docs.google.com/javax/swing/text/JTextComponent.html#getDropLocation()), [JComponent.setTransferHandler(javax.swing.TransferHandler)](http://docs.google.com/javax/swing/JComponent.html#setTransferHandler(javax.swing.TransferHandler)), [TransferHandler](http://docs.google.com/javax/swing/TransferHandler.html)

### getDropMode

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

Returns the drop mode for this component.

**Returns:**the drop mode for this component**Since:** 1.6 **See Also:**[setDropMode(javax.swing.DropMode)](http://docs.google.com/javax/swing/text/JTextComponent.html#setDropMode(javax.swing.DropMode))

### getDropLocation

public final [JTextComponent.DropLocation](http://docs.google.com/javax/swing/text/JTextComponent.DropLocation.html) **getDropLocation**()

Returns the location that this component should visually indicate as the drop location during a DnD operation over the component, or null if no location is to currently be shown.

This method is not meant for querying the drop location from a TransferHandler, as the drop location is only set after the TransferHandler's canImport has returned and has allowed for the location to be shown.

When this property changes, a property change event with name "dropLocation" is fired by the component.

**Returns:**the drop location**Since:** 1.6 **See Also:**[setDropMode(javax.swing.DropMode)](http://docs.google.com/javax/swing/text/JTextComponent.html#setDropMode(javax.swing.DropMode)), [TransferHandler.canImport(TransferHandler.TransferSupport)](http://docs.google.com/javax/swing/TransferHandler.html#canImport(javax.swing.TransferHandler.TransferSupport))

### getKeymap

public [Keymap](http://docs.google.com/javax/swing/text/Keymap.html) **getKeymap**()

Fetches the keymap currently active in this text component.

**Returns:**the keymap

### addKeymap

public static [Keymap](http://docs.google.com/javax/swing/text/Keymap.html) **addKeymap**([String](http://docs.google.com/java/lang/String.html) nm,  
 [Keymap](http://docs.google.com/javax/swing/text/Keymap.html) parent)

Adds a new keymap into the keymap hierarchy. Keymap bindings resolve from bottom up so an attribute specified in a child will override an attribute specified in the parent.

**Parameters:**nm - the name of the keymap (must be unique within the collection of named keymaps in the document); the name may be null if the keymap is unnamed, but the caller is responsible for managing the reference returned as an unnamed keymap can't be fetched by nameparent - the parent keymap; this may be null if unspecified bindings need not be resolved in some other keymap **Returns:**the keymap

### removeKeymap

public static [Keymap](http://docs.google.com/javax/swing/text/Keymap.html) **removeKeymap**([String](http://docs.google.com/java/lang/String.html) nm)

Removes a named keymap previously added to the document. Keymaps with null names may not be removed in this way.

**Parameters:**nm - the name of the keymap to remove **Returns:**the keymap that was removed

### getKeymap

public static [Keymap](http://docs.google.com/javax/swing/text/Keymap.html) **getKeymap**([String](http://docs.google.com/java/lang/String.html) nm)

Fetches a named keymap previously added to the document. This does not work with null-named keymaps.

**Parameters:**nm - the name of the keymap **Returns:**the keymap

### loadKeymap

public static void **loadKeymap**([Keymap](http://docs.google.com/javax/swing/text/Keymap.html) map,  
 [JTextComponent.KeyBinding](http://docs.google.com/javax/swing/text/JTextComponent.KeyBinding.html)[] bindings,  
 [Action](http://docs.google.com/javax/swing/Action.html)[] actions)

Loads a keymap with a bunch of bindings. This can be used to take a static table of definitions and load them into some keymap. The following example illustrates an example of binding some keys to the cut, copy, and paste actions associated with a JTextComponent. A code fragment to accomplish this might look as follows:

static final JTextComponent.KeyBinding[] defaultBindings = {  
 new JTextComponent.KeyBinding(  
 KeyStroke.getKeyStroke(KeyEvent.VK\_C, InputEvent.CTRL\_MASK),  
 DefaultEditorKit.copyAction),  
 new JTextComponent.KeyBinding(  
 KeyStroke.getKeyStroke(KeyEvent.VK\_V, InputEvent.CTRL\_MASK),  
 DefaultEditorKit.pasteAction),  
 new JTextComponent.KeyBinding(  
 KeyStroke.getKeyStroke(KeyEvent.VK\_X, InputEvent.CTRL\_MASK),  
 DefaultEditorKit.cutAction),  
 };  
  
 JTextComponent c = new JTextPane();  
 Keymap k = c.getKeymap();  
 JTextComponent.loadKeymap(k, defaultBindings, c.getActions());

The sets of bindings and actions may be empty but must be non-null.

**Parameters:**map - the keymapbindings - the bindingsactions - the set of actions

### getCaretColor

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

Fetches the current color used to render the caret.

**Returns:**the color

### setCaretColor

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

Sets the current color used to render the caret. Setting to null effectively restores the default color. Setting the color results in a PropertyChange event ("caretColor") being fired.

**Parameters:**c - the color**See Also:**[getCaretColor()](http://docs.google.com/javax/swing/text/JTextComponent.html#getCaretColor())

### getSelectionColor

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

Fetches the current color used to render the selection.

**Returns:**the color

### setSelectionColor

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

Sets the current color used to render the selection. Setting the color to null is the same as setting Color.white. Setting the color results in a PropertyChange event ("selectionColor").

**Parameters:**c - the color**See Also:**[getSelectionColor()](http://docs.google.com/javax/swing/text/JTextComponent.html#getSelectionColor())

### getSelectedTextColor

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

Fetches the current color used to render the selected text.

**Returns:**the color

### setSelectedTextColor

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

Sets the current color used to render the selected text. Setting the color to null is the same as Color.black. Setting the color results in a PropertyChange event ("selectedTextColor") being fired.

**Parameters:**c - the color**See Also:**[getSelectedTextColor()](http://docs.google.com/javax/swing/text/JTextComponent.html#getSelectedTextColor())

### getDisabledTextColor

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

Fetches the current color used to render the disabled text.

**Returns:**the color

### setDisabledTextColor

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

Sets the current color used to render the disabled text. Setting the color fires off a PropertyChange event ("disabledTextColor").

**Parameters:**c - the color**See Also:**[getDisabledTextColor()](http://docs.google.com/javax/swing/text/JTextComponent.html#getDisabledTextColor())

### replaceSelection

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

Replaces the currently selected content with new content represented by the given string. If there is no selection this amounts to an insert of the given text. If there is no replacement text this amounts to a removal of the current selection.

This is the method that is used by the default implementation of the action for inserting content that gets bound to the keymap actions.

This method is thread safe, although most Swing methods are not. Please see [How to Use Threads](http://java.sun.com/docs/books/tutorial/uiswing/misc/threads.html) for more information.

**Parameters:**content - the content to replace the selection with

### getText

public [String](http://docs.google.com/java/lang/String.html) **getText**(int offs,  
 int len)  
 throws [BadLocationException](http://docs.google.com/javax/swing/text/BadLocationException.html)

Fetches a portion of the text represented by the component. Returns an empty string if length is 0.

**Parameters:**offs - the offset >= 0len - the length >= 0 **Returns:**the text **Throws:** [BadLocationException](http://docs.google.com/javax/swing/text/BadLocationException.html) - if the offset or length are invalid

### modelToView

public [Rectangle](http://docs.google.com/java/awt/Rectangle.html) **modelToView**(int pos)  
 throws [BadLocationException](http://docs.google.com/javax/swing/text/BadLocationException.html)

Converts the given location in the model to a place in the view coordinate system. The component must have a positive size for this translation to be computed (i.e. layout cannot be computed until the component has been sized). The component does not have to be visible or painted.

**Parameters:**pos - the position >= 0 **Returns:**the coordinates as a rectangle, with (r.x, r.y) as the location in the coordinate system, or null if the component does not yet have a positive size. **Throws:** [BadLocationException](http://docs.google.com/javax/swing/text/BadLocationException.html) - if the given position does not represent a valid location in the associated document**See Also:**[TextUI.modelToView(javax.swing.text.JTextComponent, int)](http://docs.google.com/javax/swing/plaf/TextUI.html#modelToView(javax.swing.text.JTextComponent,%20int))

### viewToModel

public int **viewToModel**([Point](http://docs.google.com/java/awt/Point.html) pt)

Converts the given place in the view coordinate system to the nearest representative location in the model. The component must have a positive size for this translation to be computed (i.e. layout cannot be computed until the component has been sized). The component does not have to be visible or painted.

**Parameters:**pt - the location in the view to translate **Returns:**the offset >= 0 from the start of the document, or -1 if the component does not yet have a positive size.**See Also:**[TextUI.viewToModel(javax.swing.text.JTextComponent, java.awt.Point)](http://docs.google.com/javax/swing/plaf/TextUI.html#viewToModel(javax.swing.text.JTextComponent,%20java.awt.Point))

### cut

public void **cut**()

Transfers the currently selected range in the associated text model to the system clipboard, removing the contents from the model. The current selection is reset. Does nothing for null selections.

**See Also:**[Toolkit.getSystemClipboard()](http://docs.google.com/java/awt/Toolkit.html#getSystemClipboard()), [Clipboard](http://docs.google.com/java/awt/datatransfer/Clipboard.html)

### copy

public void **copy**()

Transfers the currently selected range in the associated text model to the system clipboard, leaving the contents in the text model. The current selection remains intact. Does nothing for null selections.

**See Also:**[Toolkit.getSystemClipboard()](http://docs.google.com/java/awt/Toolkit.html#getSystemClipboard()), [Clipboard](http://docs.google.com/java/awt/datatransfer/Clipboard.html)

### paste

public void **paste**()

Transfers the contents of the system clipboard into the associated text model. If there is a selection in the associated view, it is replaced with the contents of the clipboard. If there is no selection, the clipboard contents are inserted in front of the current insert position in the associated view. If the clipboard is empty, does nothing.

**See Also:**[replaceSelection(java.lang.String)](http://docs.google.com/javax/swing/text/JTextComponent.html#replaceSelection(java.lang.String)), [Toolkit.getSystemClipboard()](http://docs.google.com/java/awt/Toolkit.html#getSystemClipboard()), [Clipboard](http://docs.google.com/java/awt/datatransfer/Clipboard.html)

### moveCaretPosition

public void **moveCaretPosition**(int pos)

Moves the caret to a new position, leaving behind a mark defined by the last time setCaretPosition was called. This forms a selection. If the document is null, does nothing. The position must be between 0 and the length of the component's text or else an exception is thrown.

**Parameters:**pos - the position **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the value supplied for position is less than zero or greater than the component's text length**See Also:**[setCaretPosition(int)](http://docs.google.com/javax/swing/text/JTextComponent.html#setCaretPosition(int))

### setFocusAccelerator

public void **setFocusAccelerator**(char aKey)

Sets the key accelerator that will cause the receiving text component to get the focus. The accelerator will be the key combination of the *alt* key and the character given (converted to upper case). By default, there is no focus accelerator key. Any previous key accelerator setting will be superseded. A '\0' key setting will be registered, and has the effect of turning off the focus accelerator. When the new key is set, a PropertyChange event (FOCUS\_ACCELERATOR\_KEY) will be fired.

**Parameters:**aKey - the key**See Also:**[getFocusAccelerator()](http://docs.google.com/javax/swing/text/JTextComponent.html#getFocusAccelerator())

### getFocusAccelerator

public char **getFocusAccelerator**()

Returns the key accelerator that will cause the receiving text component to get the focus. Return '\0' if no focus accelerator has been set.

**Returns:**the key

### read

public void **read**([Reader](http://docs.google.com/java/io/Reader.html) in,  
 [Object](http://docs.google.com/java/lang/Object.html) desc)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Initializes from a stream. This creates a model of the type appropriate for the component and initializes the model from the stream. By default this will load the model as plain text. Previous contents of the model are discarded.

**Parameters:**in - the stream to read fromdesc - an object describing the stream; this might be a string, a File, a URL, etc. Some kinds of documents (such as html for example) might be able to make use of this information; if non-null, it is added as a property of the document **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - as thrown by the stream being used to initialize**See Also:**[EditorKit.createDefaultDocument()](http://docs.google.com/javax/swing/text/EditorKit.html#createDefaultDocument()), [setDocument(javax.swing.text.Document)](http://docs.google.com/javax/swing/text/JTextComponent.html#setDocument(javax.swing.text.Document)), [PlainDocument](http://docs.google.com/javax/swing/text/PlainDocument.html)

### write

public void **write**([Writer](http://docs.google.com/java/io/Writer.html) out)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Stores the contents of the model into the given stream. By default this will store the model as plain text.

**Parameters:**out - the output stream **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - on any I/O error

### removeNotify

public void **removeNotify**()

**Description copied from class:** [**JComponent**](http://docs.google.com/javax/swing/JComponent.html#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/javax/swing/JComponent.html#removeNotify()) in class [JComponent](http://docs.google.com/javax/swing/JComponent.html) **See Also:**[JComponent.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))

### setCaretPosition

public void **setCaretPosition**(int position)

Sets the position of the text insertion caret for the TextComponent. Note that the caret tracks change, so this may move if the underlying text of the component is changed. If the document is null, does nothing. The position must be between 0 and the length of the component's text or else an exception is thrown.

**Parameters:**position - the position **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the value supplied for position is less than zero or greater than the component's text length

### getCaretPosition

public int **getCaretPosition**()

Returns the position of the text insertion caret for the text component.

**Returns:**the position of the text insertion caret for the text component >= 0

### setText

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

Sets the text of this TextComponent to the specified text. If the text is null or empty, has the effect of simply deleting the old text. When text has been inserted, the resulting caret location is determined by the implementation of the caret class.

This method is thread safe, although most Swing methods are not. Please see [How to Use Threads](http://java.sun.com/docs/books/tutorial/uiswing/misc/threads.html) for more information. Note that text is not a bound property, so no PropertyChangeEvent is fired when it changes. To listen for changes to the text, use DocumentListener.

**Parameters:**t - the new text to be set**See Also:**[getText(int, int)](http://docs.google.com/javax/swing/text/JTextComponent.html#getText(int,%20int)), [DefaultCaret](http://docs.google.com/javax/swing/text/DefaultCaret.html)

### getText

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

Returns the text contained in this TextComponent. If the underlying document is null, will give a NullPointerException. Note that text is not a bound property, so no PropertyChangeEvent is fired when it changes. To listen for changes to the text, use DocumentListener.

**Returns:**the text **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the document is null**See Also:**[setText(java.lang.String)](http://docs.google.com/javax/swing/text/JTextComponent.html#setText(java.lang.String))

### getSelectedText

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

Returns the selected text contained in this TextComponent. If the selection is null or the document empty, returns null.

**Returns:**the text **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the selection doesn't have a valid mapping into the document for some reason**See Also:**[setText(java.lang.String)](http://docs.google.com/javax/swing/text/JTextComponent.html#setText(java.lang.String))

### isEditable

public boolean **isEditable**()

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

**Returns:**the boolean value**See Also:**[setEditable(boolean)](http://docs.google.com/javax/swing/text/JTextComponent.html#setEditable(boolean))

### setEditable

public void **setEditable**(boolean b)

Sets the specified boolean to indicate whether or not this TextComponent should be editable. A PropertyChange event ("editable") is fired when the state is changed.

**Parameters:**b - the boolean to be set**See Also:**[isEditable()](http://docs.google.com/javax/swing/text/JTextComponent.html#isEditable())

### getSelectionStart

public int **getSelectionStart**()

Returns the selected text's start position. Return 0 for an empty document, or the value of dot if no selection.

**Returns:**the start position >= 0

### setSelectionStart

public void **setSelectionStart**(int selectionStart)

Sets the selection start to the specified position. The new starting point is constrained to be before or at the current selection end.

This is available for backward compatibility to code that called this method on java.awt.TextComponent. This is implemented to forward to the Caret implementation which is where the actual selection is maintained.

**Parameters:**selectionStart - the start position of the text >= 0

### getSelectionEnd

public int **getSelectionEnd**()

Returns the selected text's end position. Return 0 if the document is empty, or the value of dot if there is no selection.

**Returns:**the end position >= 0

### setSelectionEnd

public void **setSelectionEnd**(int selectionEnd)

Sets the selection end to the specified position. The new end point is constrained to be at or after the current selection start.

This is available for backward compatibility to code that called this method on java.awt.TextComponent. This is implemented to forward to the Caret implementation which is where the actual selection is maintained.

**Parameters:**selectionEnd - the end position of the text >= 0

### select

public void **select**(int selectionStart,  
 int selectionEnd)

Selects the text between the specified start and end positions.

This method sets the start and end positions of the selected text, enforcing the restriction that the start position must be greater than or equal to zero. The end position must be greater than or equal to the start position, and less than or equal to the length of the text component's text.

If the caller supplies values that are inconsistent or out of bounds, the method enforces these constraints silently, and without failure. Specifically, if the start position or end position is greater than the length of the text, it is reset to equal the text length. If the start position is less than zero, it is reset to zero, and if the end position is less than the start position, it is reset to the start position.

This call is provided for backward compatibility. It is routed to a call to setCaretPosition followed by a call to moveCaretPosition. The preferred way to manage selection is by calling those methods directly.

**Parameters:**selectionStart - the start position of the textselectionEnd - the end position of the text**See Also:**[setCaretPosition(int)](http://docs.google.com/javax/swing/text/JTextComponent.html#setCaretPosition(int)), [moveCaretPosition(int)](http://docs.google.com/javax/swing/text/JTextComponent.html#moveCaretPosition(int))

### selectAll

public void **selectAll**()

Selects all the text in the TextComponent. Does nothing on a null or empty document.

### 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. This will return one of:

1. If setToolTipText has been invoked with a non-null value, it will be returned, otherwise
2. The value from invoking getToolTipText on the UI will be returned.

By default JTextComponent does not register itself with the ToolTipManager. This means that tooltips will NOT be shown from the TextUI unless registerComponent has been invoked on the ToolTipManager.

**Overrides:**[getToolTipText](http://docs.google.com/javax/swing/JComponent.html#getToolTipText(java.awt.event.MouseEvent)) in class [JComponent](http://docs.google.com/javax/swing/JComponent.html) **Parameters:**event - the event in question **Returns:**the string to be used as the tooltip for event**See Also:**[JComponent.setToolTipText(java.lang.String)](http://docs.google.com/javax/swing/JComponent.html#setToolTipText(java.lang.String)), [TextUI.getToolTipText(javax.swing.text.JTextComponent, java.awt.Point)](http://docs.google.com/javax/swing/plaf/TextUI.html#getToolTipText(javax.swing.text.JTextComponent,%20java.awt.Point)), [ToolTipManager.registerComponent(javax.swing.JComponent)](http://docs.google.com/javax/swing/ToolTipManager.html#registerComponent(javax.swing.JComponent))

### getPreferredScrollableViewportSize

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

Returns the preferred size of the viewport for a view component. This is implemented to do the default behavior of returning the preferred size of the component.

**Specified by:**[getPreferredScrollableViewportSize](http://docs.google.com/javax/swing/Scrollable.html#getPreferredScrollableViewportSize()) in interface [Scrollable](http://docs.google.com/javax/swing/Scrollable.html) **Returns:**the preferredSize of a JViewport whose view is this Scrollable**See Also:**[JComponent.getPreferredSize()](http://docs.google.com/javax/swing/JComponent.html#getPreferredSize())

### getScrollableUnitIncrement

public int **getScrollableUnitIncrement**([Rectangle](http://docs.google.com/java/awt/Rectangle.html) visibleRect,  
 int orientation,  
 int direction)

Components that display logical rows or columns should compute the scroll increment that will completely expose one new row or column, depending on the value of orientation. Ideally, components should handle a partially exposed row or column by returning the distance required to completely expose the item.

The default implementation of this is to simply return 10% of the visible area. Subclasses are likely to be able to provide a much more reasonable value.

**Specified by:**[getScrollableUnitIncrement](http://docs.google.com/javax/swing/Scrollable.html#getScrollableUnitIncrement(java.awt.Rectangle,%20int,%20int)) in interface [Scrollable](http://docs.google.com/javax/swing/Scrollable.html) **Parameters:**visibleRect - the view area visible within the viewportorientation - either SwingConstants.VERTICAL or SwingConstants.HORIZONTALdirection - less than zero to scroll up/left, greater than zero for down/right **Returns:**the "unit" increment for scrolling in the specified direction **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - for an invalid orientation**See Also:**[JScrollBar.setUnitIncrement(int)](http://docs.google.com/javax/swing/JScrollBar.html#setUnitIncrement(int))

### getScrollableBlockIncrement

public int **getScrollableBlockIncrement**([Rectangle](http://docs.google.com/java/awt/Rectangle.html) visibleRect,  
 int orientation,  
 int direction)

Components that display logical rows or columns should compute the scroll increment that will completely expose one block of rows or columns, depending on the value of orientation.

The default implementation of this is to simply return the visible area. Subclasses will likely be able to provide a much more reasonable value.

**Specified by:**[getScrollableBlockIncrement](http://docs.google.com/javax/swing/Scrollable.html#getScrollableBlockIncrement(java.awt.Rectangle,%20int,%20int)) in interface [Scrollable](http://docs.google.com/javax/swing/Scrollable.html) **Parameters:**visibleRect - the view area visible within the viewportorientation - either SwingConstants.VERTICAL or SwingConstants.HORIZONTALdirection - less than zero to scroll up/left, greater than zero for down/right **Returns:**the "block" increment for scrolling in the specified direction **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - for an invalid orientation**See Also:**[JScrollBar.setBlockIncrement(int)](http://docs.google.com/javax/swing/JScrollBar.html#setBlockIncrement(int))

### getScrollableTracksViewportWidth

public boolean **getScrollableTracksViewportWidth**()

Returns true if a viewport should always force the width of this Scrollable to match the width of the viewport. For example a normal text view that supported line wrapping would return true here, since it would be undesirable for wrapped lines to disappear beyond the right edge of the viewport. Note that returning true for a Scrollable whose ancestor is a JScrollPane effectively disables horizontal scrolling.

Scrolling containers, like JViewport, will use this method each time they are validated.

**Specified by:**[getScrollableTracksViewportWidth](http://docs.google.com/javax/swing/Scrollable.html#getScrollableTracksViewportWidth()) in interface [Scrollable](http://docs.google.com/javax/swing/Scrollable.html) **Returns:**true if a viewport should force the Scrollables width to match its own

### getScrollableTracksViewportHeight

public boolean **getScrollableTracksViewportHeight**()

Returns true if a viewport should always force the height of this Scrollable to match the height of the viewport. For example a columnar text view that flowed text in left to right columns could effectively disable vertical scrolling by returning true here.

Scrolling containers, like JViewport, will use this method each time they are validated.

**Specified by:**[getScrollableTracksViewportHeight](http://docs.google.com/javax/swing/Scrollable.html#getScrollableTracksViewportHeight()) in interface [Scrollable](http://docs.google.com/javax/swing/Scrollable.html) **Returns:**true if a viewport should force the Scrollables height to match its own

### print

public boolean **print**()  
 throws [PrinterException](http://docs.google.com/java/awt/print/PrinterException.html)

A convenience print method that displays a print dialog, and then prints this JTextComponent in *interactive* mode with no header or footer text. Note: this method blocks until printing is done.

Note: In *headless* mode, no dialogs will be shown.

This method calls the full featured [print](http://docs.google.com/javax/swing/text/JTextComponent.html#print(java.text.MessageFormat,%20java.text.MessageFormat,%20boolean,%20javax.print.PrintService,%20javax.print.attribute.PrintRequestAttributeSet,%20boolean)) method to perform printing.

**Returns:**true, unless printing is canceled by the user **Throws:** [PrinterException](http://docs.google.com/java/awt/print/PrinterException.html) - if an error in the print system causes the job to be aborted [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if this thread is not allowed to initiate a print job request**Since:** 1.6 **See Also:**[print(MessageFormat, MessageFormat, boolean, PrintService, PrintRequestAttributeSet, boolean)](http://docs.google.com/javax/swing/text/JTextComponent.html#print(java.text.MessageFormat,%20java.text.MessageFormat,%20boolean,%20javax.print.PrintService,%20javax.print.attribute.PrintRequestAttributeSet,%20boolean))

### print

public boolean **print**([MessageFormat](http://docs.google.com/java/text/MessageFormat.html) headerFormat,  
 [MessageFormat](http://docs.google.com/java/text/MessageFormat.html) footerFormat)  
 throws [PrinterException](http://docs.google.com/java/awt/print/PrinterException.html)

A convenience print method that displays a print dialog, and then prints this JTextComponent in *interactive* mode with the specified header and footer text. Note: this method blocks until printing is done.

Note: In *headless* mode, no dialogs will be shown.

This method calls the full featured [print](http://docs.google.com/javax/swing/text/JTextComponent.html#print(java.text.MessageFormat,%20java.text.MessageFormat,%20boolean,%20javax.print.PrintService,%20javax.print.attribute.PrintRequestAttributeSet,%20boolean)) method to perform printing.

**Parameters:**headerFormat - the text, in MessageFormat, to be used as the header, or null for no headerfooterFormat - the text, in MessageFormat, to be used as the footer, or null for no footer **Returns:**true, unless printing is canceled by the user **Throws:** [PrinterException](http://docs.google.com/java/awt/print/PrinterException.html) - if an error in the print system causes the job to be aborted [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if this thread is not allowed to initiate a print job request**Since:** 1.6 **See Also:**[print(MessageFormat, MessageFormat, boolean, PrintService, PrintRequestAttributeSet, boolean)](http://docs.google.com/javax/swing/text/JTextComponent.html#print(java.text.MessageFormat,%20java.text.MessageFormat,%20boolean,%20javax.print.PrintService,%20javax.print.attribute.PrintRequestAttributeSet,%20boolean)), [MessageFormat](http://docs.google.com/java/text/MessageFormat.html)

### print

public boolean **print**([MessageFormat](http://docs.google.com/java/text/MessageFormat.html) headerFormat,  
 [MessageFormat](http://docs.google.com/java/text/MessageFormat.html) footerFormat,  
 boolean showPrintDialog,  
 [PrintService](http://docs.google.com/javax/print/PrintService.html) service,  
 [PrintRequestAttributeSet](http://docs.google.com/javax/print/attribute/PrintRequestAttributeSet.html) attributes,  
 boolean interactive)  
 throws [PrinterException](http://docs.google.com/java/awt/print/PrinterException.html)

Prints the content of this JTextComponent. Note: this method blocks until printing is done.

Page header and footer text can be added to the output by providing MessageFormat arguments. The printing code requests Strings from the formats, providing a single item which may be included in the formatted string: an Integer representing the current page number.

showPrintDialog boolean parameter allows you to specify whether a print dialog is displayed to the user. When it is, the user may use the dialog to change printing attributes or even cancel the print.

service allows you to provide the initial PrintService for the print dialog, or to specify PrintService to print to when the dialog is not shown.

attributes can be used to provide the initial values for the print dialog, or to supply any needed attributes when the dialog is not shown. attributes can be used to control how the job will print, for example *duplex* or *single-sided*.

interactive boolean parameter allows you to specify whether to perform printing in *interactive* mode. If true, a progress dialog, with an abort option, is displayed for the duration of printing. This dialog is *modal* when print is invoked on the *Event Dispatch Thread* and *non-modal* otherwise. **Warning**: calling this method on the *Event Dispatch Thread* with interactive false blocks *all* events, including repaints, from being processed until printing is complete. It is only recommended when printing from an application with no visible GUI.

Note: In *headless* mode, showPrintDialog and interactive parameters are ignored and no dialogs are shown.

This method ensures the document is not mutated during printing. To indicate it visually, setEnabled(false) is set for the duration of printing.

This method uses [getPrintable(java.text.MessageFormat, java.text.MessageFormat)](http://docs.google.com/javax/swing/text/JTextComponent.html#getPrintable(java.text.MessageFormat,%20java.text.MessageFormat)) to render document content.

This method is thread-safe, although most Swing methods are not. Please see  [How to Use Threads](http://java.sun.com/docs/books/tutorial/uiswing/misc/threads.html) for more information.

**Sample Usage**. This code snippet shows a cross-platform print dialog and then prints the JTextComponent in *interactive* mode unless the user cancels the dialog:

textComponent.print(new MessageFormat("My text component header"),  
 new MessageFormat("Footer. Page - {0}"), true, null, null, true);

Executing this code off the *Event Dispatch Thread* performs printing on the *background*. The following pattern might be used for *background* printing:

FutureTask<Boolean> future =  
 new FutureTask<Boolean>(  
 new Callable<Boolean>() {  
 public Boolean call() {  
 return textComponent.print(.....);  
 }   
 });  
 executor.execute(future);

**Parameters:**headerFormat - the text, in MessageFormat, to be used as the header, or null for no headerfooterFormat - the text, in MessageFormat, to be used as the footer, or null for no footershowPrintDialog - true to display a print dialog, false otherwiseservice - initial PrintService, or null for the defaultattributes - the job attributes to be applied to the print job, or null for noneinteractive - whether to print in an interactive mode **Returns:**true, unless printing is canceled by the user **Throws:** [PrinterException](http://docs.google.com/java/awt/print/PrinterException.html) - if an error in the print system causes the job to be aborted [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if this thread is not allowed to initiate a print job request**Since:** 1.6 **See Also:**[getPrintable(java.text.MessageFormat, java.text.MessageFormat)](http://docs.google.com/javax/swing/text/JTextComponent.html#getPrintable(java.text.MessageFormat,%20java.text.MessageFormat)), [MessageFormat](http://docs.google.com/java/text/MessageFormat.html), [GraphicsEnvironment.isHeadless()](http://docs.google.com/java/awt/GraphicsEnvironment.html#isHeadless()), [FutureTask](http://docs.google.com/java/util/concurrent/FutureTask.html)

### getPrintable

public [Printable](http://docs.google.com/java/awt/print/Printable.html) **getPrintable**([MessageFormat](http://docs.google.com/java/text/MessageFormat.html) headerFormat,  
 [MessageFormat](http://docs.google.com/java/text/MessageFormat.html) footerFormat)

Returns a Printable to use for printing the content of this JTextComponent. The returned Printable prints the document as it looks on the screen except being reformatted to fit the paper. The returned Printable can be wrapped inside another Printable in order to create complex reports and documents.

The returned Printable shares the document with this JTextComponent. It is the responsibility of the developer to ensure that the document is not mutated while this Printable is used. Printing behavior is undefined when the document is mutated during printing.

Page header and footer text can be added to the output by providing MessageFormat arguments. The printing code requests Strings from the formats, providing a single item which may be included in the formatted string: an Integer representing the current page number.

The returned Printable when printed, formats the document content appropriately for the page size. For correct line wrapping the imageable width of all pages must be the same. See [PageFormat.getImageableWidth()](http://docs.google.com/java/awt/print/PageFormat.html#getImageableWidth()).

This method is thread-safe, although most Swing methods are not. Please see  [How to Use Threads](http://java.sun.com/docs/books/tutorial/uiswing/misc/threads.html) for more information.

The returned Printable can be printed on any thread.

This implementation returned Printable performs all painting on the *Event Dispatch Thread*, regardless of what thread it is used on.

**Parameters:**headerFormat - the text, in MessageFormat, to be used as the header, or null for no headerfooterFormat - the text, in MessageFormat, to be used as the footer, or null for no footer **Returns:**a Printable for use in printing content of this JTextComponent**Since:** 1.6 **See Also:**[Printable](http://docs.google.com/java/awt/print/Printable.html), [PageFormat](http://docs.google.com/java/awt/print/PageFormat.html), [Document.render(java.lang.Runnable)](http://docs.google.com/javax/swing/text/Document.html#render(java.lang.Runnable))

### getAccessibleContext

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

Gets the AccessibleContext associated with this JTextComponent. For text components, the AccessibleContext takes the form of an AccessibleJTextComponent. A new AccessibleJTextComponent instance is created if necessary.

**Specified by:**[getAccessibleContext](http://docs.google.com/javax/accessibility/Accessible.html#getAccessibleContext()) in interface [Accessible](http://docs.google.com/javax/accessibility/Accessible.html)**Overrides:**[getAccessibleContext](http://docs.google.com/javax/swing/JComponent.html#getAccessibleContext()) in class [JComponent](http://docs.google.com/javax/swing/JComponent.html) **Returns:**an AccessibleJTextComponent that serves as the AccessibleContext of this JTextComponent

### paramString

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

Returns a string representation of this JTextComponent. 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.

Overriding paramString to provide information about the specific new aspects of the JFC components.

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

### processInputMethodEvent

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

**Description copied from class:** [**Component**](http://docs.google.com/java/awt/Component.html#processInputMethodEvent(java.awt.event.InputMethodEvent)) Processes input method events occurring on this component by dispatching them to any registered InputMethodListener objects.

This method is not called unless input method events are enabled for this component. Input method events are enabled when one of the following occurs:

* An InputMethodListener object is registered via addInputMethodListener.
* Input method events are enabled via enableEvents.

Note that if the event parameter is null the behavior is unspecified and may result in an exception.

**Overrides:**[processInputMethodEvent](http://docs.google.com/java/awt/Component.html#processInputMethodEvent(java.awt.event.InputMethodEvent)) in class [Component](http://docs.google.com/java/awt/Component.html) **Parameters:**e - the input method event**See Also:**[InputMethodEvent](http://docs.google.com/java/awt/event/InputMethodEvent.html), [InputMethodListener](http://docs.google.com/java/awt/event/InputMethodListener.html), [Component.addInputMethodListener(java.awt.event.InputMethodListener)](http://docs.google.com/java/awt/Component.html#addInputMethodListener(java.awt.event.InputMethodListener)), [Component.enableEvents(long)](http://docs.google.com/java/awt/Component.html#enableEvents(long))

### getInputMethodRequests

public [InputMethodRequests](http://docs.google.com/java/awt/im/InputMethodRequests.html) **getInputMethodRequests**()

**Description copied from class:** [**Component**](http://docs.google.com/java/awt/Component.html#getInputMethodRequests()) Gets the input method request handler which supports requests from input methods for this component. A component that supports on-the-spot text input must override this method to return an InputMethodRequests instance. At the same time, it also has to handle input method events.

**Overrides:**[getInputMethodRequests](http://docs.google.com/java/awt/Component.html#getInputMethodRequests()) in class [Component](http://docs.google.com/java/awt/Component.html) **Returns:**the input method request handler for this component, null by default**See Also:**[Component.addInputMethodListener(java.awt.event.InputMethodListener)](http://docs.google.com/java/awt/Component.html#addInputMethodListener(java.awt.event.InputMethodListener))

### addInputMethodListener

public void **addInputMethodListener**([InputMethodListener](http://docs.google.com/java/awt/event/InputMethodListener.html) l)

**Description copied from class:** [**Component**](http://docs.google.com/java/awt/Component.html#addInputMethodListener(java.awt.event.InputMethodListener)) Adds the specified input method listener to receive input method events from this component. A component will only receive input method events from input methods if it also overrides getInputMethodRequests to return an InputMethodRequests instance. If listener l is null, no exception is thrown and no action is performed.

Refer to [AWT Threading Issues](http://docs.google.com/doc-files/AWTThreadIssues.html#ListenersThreads) for details on AWT's threading model.

**Overrides:**[addInputMethodListener](http://docs.google.com/java/awt/Component.html#addInputMethodListener(java.awt.event.InputMethodListener)) in class [Component](http://docs.google.com/java/awt/Component.html) **Parameters:**l - the input method listener**See Also:**[InputMethodEvent](http://docs.google.com/java/awt/event/InputMethodEvent.html), [InputMethodListener](http://docs.google.com/java/awt/event/InputMethodListener.html), [Component.removeInputMethodListener(java.awt.event.InputMethodListener)](http://docs.google.com/java/awt/Component.html#removeInputMethodListener(java.awt.event.InputMethodListener)), [Component.getInputMethodListeners()](http://docs.google.com/java/awt/Component.html#getInputMethodListeners()), [Component.getInputMethodRequests()](http://docs.google.com/java/awt/Component.html#getInputMethodRequests())

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/JTextComponent.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/text/InternationalFormatter.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/text/JTextComponent.AccessibleJTextComponent.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/text/JTextComponent.html)    [**NO FRAMES**](http://docs.google.com/JTextComponent.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | [FIELD](#1t3h5sf) | [CONSTR](#3rdcrjn) | [METHOD](#26in1rg) | DETAIL: [FIELD](#2jxsxqh) | [CONSTR](#1y810tw) | [METHOD](#2xcytpi) |

[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).