| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/DefaultCaret.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/DateFormatter.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/text/DefaultEditorKit.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/text/DefaultCaret.html)    [**NO FRAMES**](http://docs.google.com/DefaultCaret.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#2et92p0) | [FIELD](#tyjcwt) | [CONSTR](#4d34og8) | [METHOD](#2s8eyo1) | DETAIL: [FIELD](#1ksv4uv) | [CONSTR](#4i7ojhp) | [METHOD](#1ci93xb) |

## **javax.swing.text**

Class DefaultCaret

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 [java.awt.geom.RectangularShape](http://docs.google.com/java/awt/geom/RectangularShape.html)  
 [java.awt.geom.Rectangle2D](http://docs.google.com/java/awt/geom/Rectangle2D.html)  
 [java.awt.Rectangle](http://docs.google.com/java/awt/Rectangle.html)  
 **javax.swing.text.DefaultCaret**

**All Implemented Interfaces:** [FocusListener](http://docs.google.com/java/awt/event/FocusListener.html), [MouseListener](http://docs.google.com/java/awt/event/MouseListener.html), [MouseMotionListener](http://docs.google.com/java/awt/event/MouseMotionListener.html), [Shape](http://docs.google.com/java/awt/Shape.html), [Serializable](http://docs.google.com/java/io/Serializable.html), [Cloneable](http://docs.google.com/java/lang/Cloneable.html), [EventListener](http://docs.google.com/java/util/EventListener.html), [Caret](http://docs.google.com/javax/swing/text/Caret.html) **Direct Known Subclasses:** [BasicTextUI.BasicCaret](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.BasicCaret.html)

public class **DefaultCaret**extends [Rectangle](http://docs.google.com/java/awt/Rectangle.html)implements [Caret](http://docs.google.com/javax/swing/text/Caret.html), [FocusListener](http://docs.google.com/java/awt/event/FocusListener.html), [MouseListener](http://docs.google.com/java/awt/event/MouseListener.html), [MouseMotionListener](http://docs.google.com/java/awt/event/MouseMotionListener.html)

A default implementation of Caret. The caret is rendered as a vertical line in the color specified by the CaretColor property of the associated JTextComponent. It can blink at the rate specified by the BlinkRate property.

This implementation expects two sources of asynchronous notification. The timer thread fires asynchronously, and causes the caret to simply repaint the most recent bounding box. The caret also tracks change as the document is modified. Typically this will happen on the event dispatch thread as a result of some mouse or keyboard event. The caret behavior on both synchronous and asynchronous documents updates is controlled by UpdatePolicy property. The repaint of the new caret location will occur on the event thread in any case, as calls to modelToView are only safe on the event thread.

The caret acts as a mouse and focus listener on the text component it has been installed in, and defines the caret semantics based upon those events. The listener methods can be reimplemented to change the semantics. By default, the first mouse button will be used to set focus and caret position. Dragging the mouse pointer with the first mouse button will sweep out a selection that is contiguous in the model. If the associated text component is editable, the caret will become visible when focus is gained, and invisible when focus is lost.

The Highlighter bound to the associated text component is used to render the selection by default. Selection appearance can be customized by supplying a painter to use for the highlights. By default a painter is used that will render a solid color as specified in the associated text component in the SelectionColor property. This can easily be changed by reimplementing the [getSelectionHighlighter](#2iq8gzs) method.

A customized caret appearance can be achieved by reimplementing the paint method. If the paint method is changed, the damage method should also be reimplemented to cause a repaint for the area needed to render the caret. The caret extends the Rectangle class which is used to hold the bounding box for where the caret was last rendered. This enables the caret to repaint in a thread-safe manner when the caret moves without making a call to modelToView which is unstable between model updates and view repair (i.e. the order of delivery to DocumentListeners is not guaranteed).

The magic caret position is set to null when the caret position changes. A timer is used to determine the new location (after the caret change). When the timer fires, if the magic caret position is still null it is reset to the current caret position. Any actions that change the caret position and want the magic caret position to remain the same, must remember the magic caret position, change the cursor, and then set the magic caret position to its original value. This has the benefit that only actions that want the magic caret position to persist (such as open/down) need to know about it.

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

| **Nested Class Summary** | |
| --- | --- |

| **Nested classes/interfaces inherited from class java.awt.geom.**[**Rectangle2D**](http://docs.google.com/java/awt/geom/Rectangle2D.html) |
| --- |
| [Rectangle2D.Double](http://docs.google.com/java/awt/geom/Rectangle2D.Double.html), [Rectangle2D.Float](http://docs.google.com/java/awt/geom/Rectangle2D.Float.html) |

| **Field Summary** | |
| --- | --- |
| static int | [**ALWAYS\_UPDATE**](http://docs.google.com/javax/swing/text/DefaultCaret.html#ALWAYS_UPDATE)            Indicates that the caret position is to be **always** updated accordingly to the document changes regardless whether the document updates are performed on the Event Dispatching Thread or not. |
| protected  [ChangeEvent](http://docs.google.com/javax/swing/event/ChangeEvent.html) | [**changeEvent**](http://docs.google.com/javax/swing/text/DefaultCaret.html#changeEvent)            The change event for the model. |
| protected  [EventListenerList](http://docs.google.com/javax/swing/event/EventListenerList.html) | [**listenerList**](http://docs.google.com/javax/swing/text/DefaultCaret.html#listenerList)            The event listener list. |
| static int | [**NEVER\_UPDATE**](http://docs.google.com/javax/swing/text/DefaultCaret.html#NEVER_UPDATE)            Indicates that the caret should remain at the same absolute position in the document regardless of any document updates, except when the document length becomes less than the current caret position due to removal. |
| static int | [**UPDATE\_WHEN\_ON\_EDT**](http://docs.google.com/javax/swing/text/DefaultCaret.html#UPDATE_WHEN_ON_EDT)            Indicates that the caret position is to be updated only when document changes are performed on the Event Dispatching Thread. |

| **Fields inherited from class java.awt.**[**Rectangle**](http://docs.google.com/java/awt/Rectangle.html) |
| --- |
| [height](http://docs.google.com/java/awt/Rectangle.html#height), [width](http://docs.google.com/java/awt/Rectangle.html#width), [x](http://docs.google.com/java/awt/Rectangle.html#x), [y](http://docs.google.com/java/awt/Rectangle.html#y) |

| **Fields inherited from class java.awt.geom.**[**Rectangle2D**](http://docs.google.com/java/awt/geom/Rectangle2D.html) |
| --- |
| [OUT\_BOTTOM](http://docs.google.com/java/awt/geom/Rectangle2D.html#OUT_BOTTOM), [OUT\_LEFT](http://docs.google.com/java/awt/geom/Rectangle2D.html#OUT_LEFT), [OUT\_RIGHT](http://docs.google.com/java/awt/geom/Rectangle2D.html#OUT_RIGHT), [OUT\_TOP](http://docs.google.com/java/awt/geom/Rectangle2D.html#OUT_TOP) |

| **Constructor Summary** | |
| --- | --- |
| [**DefaultCaret**](http://docs.google.com/javax/swing/text/DefaultCaret.html#DefaultCaret())()            Constructs a default caret. |

| **Method Summary** | |
| --- | --- |
| void | [**addChangeListener**](http://docs.google.com/javax/swing/text/DefaultCaret.html#addChangeListener(javax.swing.event.ChangeListener))([ChangeListener](http://docs.google.com/javax/swing/event/ChangeListener.html) l)            Adds a listener to track whenever the caret position has been changed. |
| protected  void | [**adjustVisibility**](http://docs.google.com/javax/swing/text/DefaultCaret.html#adjustVisibility(java.awt.Rectangle))([Rectangle](http://docs.google.com/java/awt/Rectangle.html) nloc)            Scrolls the associated view (if necessary) to make the caret visible. |
| protected  void | [**damage**](http://docs.google.com/javax/swing/text/DefaultCaret.html#damage(java.awt.Rectangle))([Rectangle](http://docs.google.com/java/awt/Rectangle.html) r)            Damages the area surrounding the caret to cause it to be repainted in a new location. |
| void | [**deinstall**](http://docs.google.com/javax/swing/text/DefaultCaret.html#deinstall(javax.swing.text.JTextComponent))([JTextComponent](http://docs.google.com/javax/swing/text/JTextComponent.html) c)            Called when the UI is being removed from the interface of a JTextComponent. |
| boolean | [**equals**](http://docs.google.com/javax/swing/text/DefaultCaret.html#equals(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) obj)            Compares this object to the specified object. |
| protected  void | [**fireStateChanged**](http://docs.google.com/javax/swing/text/DefaultCaret.html#fireStateChanged())()            Notifies all listeners that have registered interest for notification on this event type. |
| void | [**focusGained**](http://docs.google.com/javax/swing/text/DefaultCaret.html#focusGained(java.awt.event.FocusEvent))([FocusEvent](http://docs.google.com/java/awt/event/FocusEvent.html) e)            Called when the component containing the caret gains focus. |
| void | [**focusLost**](http://docs.google.com/javax/swing/text/DefaultCaret.html#focusLost(java.awt.event.FocusEvent))([FocusEvent](http://docs.google.com/java/awt/event/FocusEvent.html) e)            Called when the component containing the caret loses focus. |
| int | [**getBlinkRate**](http://docs.google.com/javax/swing/text/DefaultCaret.html#getBlinkRate())()            Gets the caret blink rate. |
| [ChangeListener](http://docs.google.com/javax/swing/event/ChangeListener.html)[] | [**getChangeListeners**](http://docs.google.com/javax/swing/text/DefaultCaret.html#getChangeListeners())()            Returns an array of all the change listeners registered on this caret. |
| protected  [JTextComponent](http://docs.google.com/javax/swing/text/JTextComponent.html) | [**getComponent**](http://docs.google.com/javax/swing/text/DefaultCaret.html#getComponent())()            Gets the text editor component that this caret is is bound to. |
| int | [**getDot**](http://docs.google.com/javax/swing/text/DefaultCaret.html#getDot())()            Fetches the current position of the caret. |
| [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html) | [**getDotBias**](http://docs.google.com/javax/swing/text/DefaultCaret.html#getDotBias())()            Returns the bias of the caret position. |
| | <T extends [EventListener](http://docs.google.com/java/util/EventListener.html)>  T[] | | --- | | [**getListeners**](http://docs.google.com/javax/swing/text/DefaultCaret.html#getListeners(java.lang.Class))([Class](http://docs.google.com/java/lang/Class.html)<T> listenerType)            Returns an array of all the objects currently registered as *Foo*Listeners upon this caret. |
| [Point](http://docs.google.com/java/awt/Point.html) | [**getMagicCaretPosition**](http://docs.google.com/javax/swing/text/DefaultCaret.html#getMagicCaretPosition())()            Gets the saved caret position. |
| int | [**getMark**](http://docs.google.com/javax/swing/text/DefaultCaret.html#getMark())()            Fetches the current position of the mark. |
| [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html) | [**getMarkBias**](http://docs.google.com/javax/swing/text/DefaultCaret.html#getMarkBias())()            Returns the bias of the mark. |
| protected  [Highlighter.HighlightPainter](http://docs.google.com/javax/swing/text/Highlighter.HighlightPainter.html) | [**getSelectionPainter**](http://docs.google.com/javax/swing/text/DefaultCaret.html#getSelectionPainter())()            Gets the painter for the Highlighter. |
| int | [**getUpdatePolicy**](http://docs.google.com/javax/swing/text/DefaultCaret.html#getUpdatePolicy())()            Gets the caret movement policy on document updates. |
| void | [**install**](http://docs.google.com/javax/swing/text/DefaultCaret.html#install(javax.swing.text.JTextComponent))([JTextComponent](http://docs.google.com/javax/swing/text/JTextComponent.html) c)            Called when the UI is being installed into the interface of a JTextComponent. |
| boolean | [**isActive**](http://docs.google.com/javax/swing/text/DefaultCaret.html#isActive())()            Determines if the caret is currently active. |
| boolean | [**isSelectionVisible**](http://docs.google.com/javax/swing/text/DefaultCaret.html#isSelectionVisible())()            Checks whether the current selection is visible. |
| boolean | [**isVisible**](http://docs.google.com/javax/swing/text/DefaultCaret.html#isVisible())()            Indicates whether or not the caret is currently visible. |
| void | [**mouseClicked**](http://docs.google.com/javax/swing/text/DefaultCaret.html#mouseClicked(java.awt.event.MouseEvent))([MouseEvent](http://docs.google.com/java/awt/event/MouseEvent.html) e)            Called when the mouse is clicked. |
| void | [**mouseDragged**](http://docs.google.com/javax/swing/text/DefaultCaret.html#mouseDragged(java.awt.event.MouseEvent))([MouseEvent](http://docs.google.com/java/awt/event/MouseEvent.html) e)            Moves the caret position according to the mouse pointer's current location. |
| void | [**mouseEntered**](http://docs.google.com/javax/swing/text/DefaultCaret.html#mouseEntered(java.awt.event.MouseEvent))([MouseEvent](http://docs.google.com/java/awt/event/MouseEvent.html) e)            Called when the mouse enters a region. |
| void | [**mouseExited**](http://docs.google.com/javax/swing/text/DefaultCaret.html#mouseExited(java.awt.event.MouseEvent))([MouseEvent](http://docs.google.com/java/awt/event/MouseEvent.html) e)            Called when the mouse exits a region. |
| void | [**mouseMoved**](http://docs.google.com/javax/swing/text/DefaultCaret.html#mouseMoved(java.awt.event.MouseEvent))([MouseEvent](http://docs.google.com/java/awt/event/MouseEvent.html) e)            Called when the mouse is moved. |
| void | [**mousePressed**](http://docs.google.com/javax/swing/text/DefaultCaret.html#mousePressed(java.awt.event.MouseEvent))([MouseEvent](http://docs.google.com/java/awt/event/MouseEvent.html) e)            If button 1 is pressed, this is implemented to request focus on the associated text component, and to set the caret position. |
| void | [**mouseReleased**](http://docs.google.com/javax/swing/text/DefaultCaret.html#mouseReleased(java.awt.event.MouseEvent))([MouseEvent](http://docs.google.com/java/awt/event/MouseEvent.html) e)            Called when the mouse is released. |
| protected  void | [**moveCaret**](http://docs.google.com/javax/swing/text/DefaultCaret.html#moveCaret(java.awt.event.MouseEvent))([MouseEvent](http://docs.google.com/java/awt/event/MouseEvent.html) e)            Tries to move the position of the caret from the coordinates of a mouse event, using viewToModel(). |
| void | [**moveDot**](http://docs.google.com/javax/swing/text/DefaultCaret.html#moveDot(int))(int dot)            Moves the caret position to the specified position, with a forward bias. |
| void | [**moveDot**](http://docs.google.com/javax/swing/text/DefaultCaret.html#moveDot(int,%20javax.swing.text.Position.Bias))(int dot, [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html) dotBias)            Moves the caret position to the specified position, with the specified bias. |
| void | [**paint**](http://docs.google.com/javax/swing/text/DefaultCaret.html#paint(java.awt.Graphics))([Graphics](http://docs.google.com/java/awt/Graphics.html) g)            Renders the caret as a vertical line. |
| protected  void | [**positionCaret**](http://docs.google.com/javax/swing/text/DefaultCaret.html#positionCaret(java.awt.event.MouseEvent))([MouseEvent](http://docs.google.com/java/awt/event/MouseEvent.html) e)            Tries to set the position of the caret from the coordinates of a mouse event, using viewToModel(). |
| void | [**removeChangeListener**](http://docs.google.com/javax/swing/text/DefaultCaret.html#removeChangeListener(javax.swing.event.ChangeListener))([ChangeListener](http://docs.google.com/javax/swing/event/ChangeListener.html) l)            Removes a listener that was tracking caret position changes. |
| protected  void | [**repaint**](http://docs.google.com/javax/swing/text/DefaultCaret.html#repaint())()            Cause the caret to be painted. |
| void | [**setBlinkRate**](http://docs.google.com/javax/swing/text/DefaultCaret.html#setBlinkRate(int))(int rate)            Sets the caret blink rate. |
| void | [**setDot**](http://docs.google.com/javax/swing/text/DefaultCaret.html#setDot(int))(int dot)            Sets the caret position and mark to the specified position, with a forward bias. |
| void | [**setDot**](http://docs.google.com/javax/swing/text/DefaultCaret.html#setDot(int,%20javax.swing.text.Position.Bias))(int dot, [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html) dotBias)            Sets the caret position and mark to the specified position, with the specified bias. |
| void | [**setMagicCaretPosition**](http://docs.google.com/javax/swing/text/DefaultCaret.html#setMagicCaretPosition(java.awt.Point))([Point](http://docs.google.com/java/awt/Point.html) p)            Saves the current caret position. |
| void | [**setSelectionVisible**](http://docs.google.com/javax/swing/text/DefaultCaret.html#setSelectionVisible(boolean))(boolean vis)            Changes the selection visibility. |
| void | [**setUpdatePolicy**](http://docs.google.com/javax/swing/text/DefaultCaret.html#setUpdatePolicy(int))(int policy)            Sets the caret movement policy on the document updates. |
| void | [**setVisible**](http://docs.google.com/javax/swing/text/DefaultCaret.html#setVisible(boolean))(boolean e)            Sets the caret visibility, and repaints the caret. |
| [String](http://docs.google.com/java/lang/String.html) | [**toString**](http://docs.google.com/javax/swing/text/DefaultCaret.html#toString())()            Returns a String representing this Rectangle and its values. |

| **Methods inherited from class java.awt.**[**Rectangle**](http://docs.google.com/java/awt/Rectangle.html) |
| --- |
| [add](http://docs.google.com/java/awt/Rectangle.html#add(int,%20int)), [add](http://docs.google.com/java/awt/Rectangle.html#add(java.awt.Point)), [add](http://docs.google.com/java/awt/Rectangle.html#add(java.awt.Rectangle)), [contains](http://docs.google.com/java/awt/Rectangle.html#contains(int,%20int)), [contains](http://docs.google.com/java/awt/Rectangle.html#contains(int,%20int,%20int,%20int)), [contains](http://docs.google.com/java/awt/Rectangle.html#contains(java.awt.Point)), [contains](http://docs.google.com/java/awt/Rectangle.html#contains(java.awt.Rectangle)), [createIntersection](http://docs.google.com/java/awt/Rectangle.html#createIntersection(java.awt.geom.Rectangle2D)), [createUnion](http://docs.google.com/java/awt/Rectangle.html#createUnion(java.awt.geom.Rectangle2D)), [getBounds](http://docs.google.com/java/awt/Rectangle.html#getBounds()), [getBounds2D](http://docs.google.com/java/awt/Rectangle.html#getBounds2D()), [getHeight](http://docs.google.com/java/awt/Rectangle.html#getHeight()), [getLocation](http://docs.google.com/java/awt/Rectangle.html#getLocation()), [getSize](http://docs.google.com/java/awt/Rectangle.html#getSize()), [getWidth](http://docs.google.com/java/awt/Rectangle.html#getWidth()), [getX](http://docs.google.com/java/awt/Rectangle.html#getX()), [getY](http://docs.google.com/java/awt/Rectangle.html#getY()), [grow](http://docs.google.com/java/awt/Rectangle.html#grow(int,%20int)), [inside](http://docs.google.com/java/awt/Rectangle.html#inside(int,%20int)), [intersection](http://docs.google.com/java/awt/Rectangle.html#intersection(java.awt.Rectangle)), [intersects](http://docs.google.com/java/awt/Rectangle.html#intersects(java.awt.Rectangle)), [isEmpty](http://docs.google.com/java/awt/Rectangle.html#isEmpty()), [move](http://docs.google.com/java/awt/Rectangle.html#move(int,%20int)), [outcode](http://docs.google.com/java/awt/Rectangle.html#outcode(double,%20double)), [reshape](http://docs.google.com/java/awt/Rectangle.html#reshape(int,%20int,%20int,%20int)), [resize](http://docs.google.com/java/awt/Rectangle.html#resize(int,%20int)), [setBounds](http://docs.google.com/java/awt/Rectangle.html#setBounds(int,%20int,%20int,%20int)), [setBounds](http://docs.google.com/java/awt/Rectangle.html#setBounds(java.awt.Rectangle)), [setLocation](http://docs.google.com/java/awt/Rectangle.html#setLocation(int,%20int)), [setLocation](http://docs.google.com/java/awt/Rectangle.html#setLocation(java.awt.Point)), [setRect](http://docs.google.com/java/awt/Rectangle.html#setRect(double,%20double,%20double,%20double)), [setSize](http://docs.google.com/java/awt/Rectangle.html#setSize(java.awt.Dimension)), [setSize](http://docs.google.com/java/awt/Rectangle.html#setSize(int,%20int)), [translate](http://docs.google.com/java/awt/Rectangle.html#translate(int,%20int)), [union](http://docs.google.com/java/awt/Rectangle.html#union(java.awt.Rectangle)) |

| **Methods inherited from class java.awt.geom.**[**Rectangle2D**](http://docs.google.com/java/awt/geom/Rectangle2D.html) |
| --- |
| [add](http://docs.google.com/java/awt/geom/Rectangle2D.html#add(double,%20double)), [add](http://docs.google.com/java/awt/geom/Rectangle2D.html#add(java.awt.geom.Point2D)), [add](http://docs.google.com/java/awt/geom/Rectangle2D.html#add(java.awt.geom.Rectangle2D)), [contains](http://docs.google.com/java/awt/geom/Rectangle2D.html#contains(double,%20double)), [contains](http://docs.google.com/java/awt/geom/Rectangle2D.html#contains(double,%20double,%20double,%20double)), [getPathIterator](http://docs.google.com/java/awt/geom/Rectangle2D.html#getPathIterator(java.awt.geom.AffineTransform)), [getPathIterator](http://docs.google.com/java/awt/geom/Rectangle2D.html#getPathIterator(java.awt.geom.AffineTransform,%20double)), [hashCode](http://docs.google.com/java/awt/geom/Rectangle2D.html#hashCode()), [intersect](http://docs.google.com/java/awt/geom/Rectangle2D.html#intersect(java.awt.geom.Rectangle2D,%20java.awt.geom.Rectangle2D,%20java.awt.geom.Rectangle2D)), [intersects](http://docs.google.com/java/awt/geom/Rectangle2D.html#intersects(double,%20double,%20double,%20double)), [intersectsLine](http://docs.google.com/java/awt/geom/Rectangle2D.html#intersectsLine(double,%20double,%20double,%20double)), [intersectsLine](http://docs.google.com/java/awt/geom/Rectangle2D.html#intersectsLine(java.awt.geom.Line2D)), [outcode](http://docs.google.com/java/awt/geom/Rectangle2D.html#outcode(java.awt.geom.Point2D)), [setFrame](http://docs.google.com/java/awt/geom/Rectangle2D.html#setFrame(double,%20double,%20double,%20double)), [setRect](http://docs.google.com/java/awt/geom/Rectangle2D.html#setRect(java.awt.geom.Rectangle2D)), [union](http://docs.google.com/java/awt/geom/Rectangle2D.html#union(java.awt.geom.Rectangle2D,%20java.awt.geom.Rectangle2D,%20java.awt.geom.Rectangle2D)) |

| **Methods inherited from class java.awt.geom.**[**RectangularShape**](http://docs.google.com/java/awt/geom/RectangularShape.html) |
| --- |
| [clone](http://docs.google.com/java/awt/geom/RectangularShape.html#clone()), [contains](http://docs.google.com/java/awt/geom/RectangularShape.html#contains(java.awt.geom.Point2D)), [contains](http://docs.google.com/java/awt/geom/RectangularShape.html#contains(java.awt.geom.Rectangle2D)), [getCenterX](http://docs.google.com/java/awt/geom/RectangularShape.html#getCenterX()), [getCenterY](http://docs.google.com/java/awt/geom/RectangularShape.html#getCenterY()), [getFrame](http://docs.google.com/java/awt/geom/RectangularShape.html#getFrame()), [getMaxX](http://docs.google.com/java/awt/geom/RectangularShape.html#getMaxX()), [getMaxY](http://docs.google.com/java/awt/geom/RectangularShape.html#getMaxY()), [getMinX](http://docs.google.com/java/awt/geom/RectangularShape.html#getMinX()), [getMinY](http://docs.google.com/java/awt/geom/RectangularShape.html#getMinY()), [intersects](http://docs.google.com/java/awt/geom/RectangularShape.html#intersects(java.awt.geom.Rectangle2D)), [setFrame](http://docs.google.com/java/awt/geom/RectangularShape.html#setFrame(java.awt.geom.Point2D,%20java.awt.geom.Dimension2D)), [setFrame](http://docs.google.com/java/awt/geom/RectangularShape.html#setFrame(java.awt.geom.Rectangle2D)), [setFrameFromCenter](http://docs.google.com/java/awt/geom/RectangularShape.html#setFrameFromCenter(double,%20double,%20double,%20double)), [setFrameFromCenter](http://docs.google.com/java/awt/geom/RectangularShape.html#setFrameFromCenter(java.awt.geom.Point2D,%20java.awt.geom.Point2D)), [setFrameFromDiagonal](http://docs.google.com/java/awt/geom/RectangularShape.html#setFrameFromDiagonal(double,%20double,%20double,%20double)), [setFrameFromDiagonal](http://docs.google.com/java/awt/geom/RectangularShape.html#setFrameFromDiagonal(java.awt.geom.Point2D,%20java.awt.geom.Point2D)) |

| **Methods inherited from class java.lang.**[**Object**](http://docs.google.com/java/lang/Object.html) |
| --- |
| [finalize](http://docs.google.com/java/lang/Object.html#finalize()), [getClass](http://docs.google.com/java/lang/Object.html#getClass()), [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)) |

| **Methods inherited from interface java.awt.**[**Shape**](http://docs.google.com/java/awt/Shape.html) |
| --- |
| [contains](http://docs.google.com/java/awt/Shape.html#contains(double,%20double)), [contains](http://docs.google.com/java/awt/Shape.html#contains(double,%20double,%20double,%20double)), [contains](http://docs.google.com/java/awt/Shape.html#contains(java.awt.geom.Point2D)), [contains](http://docs.google.com/java/awt/Shape.html#contains(java.awt.geom.Rectangle2D)), [getPathIterator](http://docs.google.com/java/awt/Shape.html#getPathIterator(java.awt.geom.AffineTransform)), [getPathIterator](http://docs.google.com/java/awt/Shape.html#getPathIterator(java.awt.geom.AffineTransform,%20double)), [intersects](http://docs.google.com/java/awt/Shape.html#intersects(double,%20double,%20double,%20double)), [intersects](http://docs.google.com/java/awt/Shape.html#intersects(java.awt.geom.Rectangle2D)) |

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

### UPDATE\_WHEN\_ON\_EDT

public static final int **UPDATE\_WHEN\_ON\_EDT**

Indicates that the caret position is to be updated only when document changes are performed on the Event Dispatching Thread.

**Since:** 1.5 **See Also:**[setUpdatePolicy(int)](http://docs.google.com/javax/swing/text/DefaultCaret.html#setUpdatePolicy(int)), [getUpdatePolicy()](http://docs.google.com/javax/swing/text/DefaultCaret.html#getUpdatePolicy()), [Constant Field Values](http://docs.google.com/constant-values.html#javax.swing.text.DefaultCaret.UPDATE_WHEN_ON_EDT)

### NEVER\_UPDATE

public static final int **NEVER\_UPDATE**

Indicates that the caret should remain at the same absolute position in the document regardless of any document updates, except when the document length becomes less than the current caret position due to removal. In that case the caret position is adjusted to the end of the document.

**Since:** 1.5 **See Also:**[setUpdatePolicy(int)](http://docs.google.com/javax/swing/text/DefaultCaret.html#setUpdatePolicy(int)), [getUpdatePolicy()](http://docs.google.com/javax/swing/text/DefaultCaret.html#getUpdatePolicy()), [Constant Field Values](http://docs.google.com/constant-values.html#javax.swing.text.DefaultCaret.NEVER_UPDATE)

### ALWAYS\_UPDATE

public static final int **ALWAYS\_UPDATE**

Indicates that the caret position is to be **always** updated accordingly to the document changes regardless whether the document updates are performed on the Event Dispatching Thread or not.

**Since:** 1.5 **See Also:**[setUpdatePolicy(int)](http://docs.google.com/javax/swing/text/DefaultCaret.html#setUpdatePolicy(int)), [getUpdatePolicy()](http://docs.google.com/javax/swing/text/DefaultCaret.html#getUpdatePolicy()), [Constant Field Values](http://docs.google.com/constant-values.html#javax.swing.text.DefaultCaret.ALWAYS_UPDATE)

### listenerList

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

The event listener list.

### changeEvent

protected transient [ChangeEvent](http://docs.google.com/javax/swing/event/ChangeEvent.html) **changeEvent**

The change event for the model. Only one ChangeEvent is needed per model instance since the event's only (read-only) state is the source property. The source of events generated here is always "this".

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

### DefaultCaret

public **DefaultCaret**()

Constructs a default caret.

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

### setUpdatePolicy

public void **setUpdatePolicy**(int policy)

Sets the caret movement policy on the document updates. Normally the caret updates its absolute position within the document on insertions occurred before or at the caret position and on removals before the caret position. 'Absolute position' means here the position relative to the start of the document. For example if a character is typed within editable text component it is inserted at the caret position and the caret moves to the next absolute position within the document due to insertion and if BACKSPACE is typed then caret decreases its absolute position due to removal of a character before it. Sometimes it may be useful to turn off the caret position updates so that the caret stays at the same absolute position within the document position regardless of any document updates.

The following update policies are allowed:

* NEVER\_UPDATE: the caret stays at the same absolute position in the document regardless of any document updates, except when document length becomes less than the current caret position due to removal. In that case caret position is adjusted to the end of the document. The caret doesn't try to keep itself visible by scrolling the associated view when using this policy.
* ALWAYS\_UPDATE: the caret always tracks document changes. For regular changes it increases its position if an insertion occurs before or at its current position, and decreases position if a removal occurs before its current position. For undo/redo updates it is always moved to the position where update occurred. The caret also tries to keep itself visible by calling adjustVisibility method.
* UPDATE\_WHEN\_ON\_EDT: acts like ALWAYS\_UPDATE if the document updates are performed on the Event Dispatching Thread and like NEVER\_UPDATE if updates are performed on other thread.

The default property value is UPDATE\_WHEN\_ON\_EDT.

**Parameters:**policy - one of the following values : UPDATE\_WHEN\_ON\_EDT, NEVER\_UPDATE, ALWAYS\_UPDATE **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if invalid value is passed**Since:** 1.5 **See Also:**[getUpdatePolicy()](http://docs.google.com/javax/swing/text/DefaultCaret.html#getUpdatePolicy()), [adjustVisibility(java.awt.Rectangle)](http://docs.google.com/javax/swing/text/DefaultCaret.html#adjustVisibility(java.awt.Rectangle)), [UPDATE\_WHEN\_ON\_EDT](http://docs.google.com/javax/swing/text/DefaultCaret.html#UPDATE_WHEN_ON_EDT), [NEVER\_UPDATE](http://docs.google.com/javax/swing/text/DefaultCaret.html#NEVER_UPDATE), [ALWAYS\_UPDATE](http://docs.google.com/javax/swing/text/DefaultCaret.html#ALWAYS_UPDATE)

### getUpdatePolicy

public int **getUpdatePolicy**()

Gets the caret movement policy on document updates.

**Returns:**one of the following values : UPDATE\_WHEN\_ON\_EDT, NEVER\_UPDATE, ALWAYS\_UPDATE**Since:** 1.5 **See Also:**[setUpdatePolicy(int)](http://docs.google.com/javax/swing/text/DefaultCaret.html#setUpdatePolicy(int)), [UPDATE\_WHEN\_ON\_EDT](http://docs.google.com/javax/swing/text/DefaultCaret.html#UPDATE_WHEN_ON_EDT), [NEVER\_UPDATE](http://docs.google.com/javax/swing/text/DefaultCaret.html#NEVER_UPDATE), [ALWAYS\_UPDATE](http://docs.google.com/javax/swing/text/DefaultCaret.html#ALWAYS_UPDATE)

### getComponent

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

Gets the text editor component that this caret is is bound to.

**Returns:**the component

### repaint

protected final void **repaint**()

Cause the caret to be painted. The repaint area is the bounding box of the caret (i.e. the caret rectangle or *this*).

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.

### damage

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

Damages the area surrounding the caret to cause it to be repainted in a new location. If paint() is reimplemented, this method should also be reimplemented. This method should update the caret bounds (x, y, width, and height).

**Parameters:**r - the current location of the caret**See Also:**[paint(java.awt.Graphics)](http://docs.google.com/javax/swing/text/DefaultCaret.html#paint(java.awt.Graphics))

### adjustVisibility

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

Scrolls the associated view (if necessary) to make the caret visible. Since how this should be done is somewhat of a policy, this method can be reimplemented to change the behavior. By default the scrollRectToVisible method is called on the associated component.

**Parameters:**nloc - the new position to scroll to

### getSelectionPainter

protected [Highlighter.HighlightPainter](http://docs.google.com/javax/swing/text/Highlighter.HighlightPainter.html) **getSelectionPainter**()

Gets the painter for the Highlighter.

**Returns:**the painter

### positionCaret

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

Tries to set the position of the caret from the coordinates of a mouse event, using viewToModel().

**Parameters:**e - the mouse event

### moveCaret

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

Tries to move the position of the caret from the coordinates of a mouse event, using viewToModel(). This will cause a selection if the dot and mark are different.

**Parameters:**e - the mouse event

### focusGained

public void **focusGained**([FocusEvent](http://docs.google.com/java/awt/event/FocusEvent.html) e)

Called when the component containing the caret gains focus. This is implemented to set the caret to visible if the component is editable.

**Specified by:**[focusGained](http://docs.google.com/java/awt/event/FocusListener.html#focusGained(java.awt.event.FocusEvent)) in interface [FocusListener](http://docs.google.com/java/awt/event/FocusListener.html) **Parameters:**e - the focus event**See Also:**[FocusListener.focusGained(java.awt.event.FocusEvent)](http://docs.google.com/java/awt/event/FocusListener.html#focusGained(java.awt.event.FocusEvent))

### focusLost

public void **focusLost**([FocusEvent](http://docs.google.com/java/awt/event/FocusEvent.html) e)

Called when the component containing the caret loses focus. This is implemented to set the caret to visibility to false.

**Specified by:**[focusLost](http://docs.google.com/java/awt/event/FocusListener.html#focusLost(java.awt.event.FocusEvent)) in interface [FocusListener](http://docs.google.com/java/awt/event/FocusListener.html) **Parameters:**e - the focus event**See Also:**[FocusListener.focusLost(java.awt.event.FocusEvent)](http://docs.google.com/java/awt/event/FocusListener.html#focusLost(java.awt.event.FocusEvent))

### mouseClicked

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

Called when the mouse is clicked. If the click was generated from button1, a double click selects a word, and a triple click the current line.

**Specified by:**[mouseClicked](http://docs.google.com/java/awt/event/MouseListener.html#mouseClicked(java.awt.event.MouseEvent)) in interface [MouseListener](http://docs.google.com/java/awt/event/MouseListener.html) **Parameters:**e - the mouse event**See Also:**[MouseListener.mouseClicked(java.awt.event.MouseEvent)](http://docs.google.com/java/awt/event/MouseListener.html#mouseClicked(java.awt.event.MouseEvent))

### mousePressed

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

If button 1 is pressed, this is implemented to request focus on the associated text component, and to set the caret position. If the shift key is held down, the caret will be moved, potentially resulting in a selection, otherwise the caret position will be set to the new location. If the component is not enabled, there will be no request for focus.

**Specified by:**[mousePressed](http://docs.google.com/java/awt/event/MouseListener.html#mousePressed(java.awt.event.MouseEvent)) in interface [MouseListener](http://docs.google.com/java/awt/event/MouseListener.html) **Parameters:**e - the mouse event**See Also:**[MouseListener.mousePressed(java.awt.event.MouseEvent)](http://docs.google.com/java/awt/event/MouseListener.html#mousePressed(java.awt.event.MouseEvent))

### mouseReleased

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

Called when the mouse is released.

**Specified by:**[mouseReleased](http://docs.google.com/java/awt/event/MouseListener.html#mouseReleased(java.awt.event.MouseEvent)) in interface [MouseListener](http://docs.google.com/java/awt/event/MouseListener.html) **Parameters:**e - the mouse event**See Also:**[MouseListener.mouseReleased(java.awt.event.MouseEvent)](http://docs.google.com/java/awt/event/MouseListener.html#mouseReleased(java.awt.event.MouseEvent))

### mouseEntered

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

Called when the mouse enters a region.

**Specified by:**[mouseEntered](http://docs.google.com/java/awt/event/MouseListener.html#mouseEntered(java.awt.event.MouseEvent)) in interface [MouseListener](http://docs.google.com/java/awt/event/MouseListener.html) **Parameters:**e - the mouse event**See Also:**[MouseListener.mouseEntered(java.awt.event.MouseEvent)](http://docs.google.com/java/awt/event/MouseListener.html#mouseEntered(java.awt.event.MouseEvent))

### mouseExited

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

Called when the mouse exits a region.

**Specified by:**[mouseExited](http://docs.google.com/java/awt/event/MouseListener.html#mouseExited(java.awt.event.MouseEvent)) in interface [MouseListener](http://docs.google.com/java/awt/event/MouseListener.html) **Parameters:**e - the mouse event**See Also:**[MouseListener.mouseExited(java.awt.event.MouseEvent)](http://docs.google.com/java/awt/event/MouseListener.html#mouseExited(java.awt.event.MouseEvent))

### mouseDragged

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

Moves the caret position according to the mouse pointer's current location. This effectively extends the selection. By default, this is only done for mouse button 1.

**Specified by:**[mouseDragged](http://docs.google.com/java/awt/event/MouseMotionListener.html#mouseDragged(java.awt.event.MouseEvent)) in interface [MouseMotionListener](http://docs.google.com/java/awt/event/MouseMotionListener.html) **Parameters:**e - the mouse event**See Also:**[MouseMotionListener.mouseDragged(java.awt.event.MouseEvent)](http://docs.google.com/java/awt/event/MouseMotionListener.html#mouseDragged(java.awt.event.MouseEvent))

### mouseMoved

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

Called when the mouse is moved.

**Specified by:**[mouseMoved](http://docs.google.com/java/awt/event/MouseMotionListener.html#mouseMoved(java.awt.event.MouseEvent)) in interface [MouseMotionListener](http://docs.google.com/java/awt/event/MouseMotionListener.html) **Parameters:**e - the mouse event**See Also:**[MouseMotionListener.mouseMoved(java.awt.event.MouseEvent)](http://docs.google.com/java/awt/event/MouseMotionListener.html#mouseMoved(java.awt.event.MouseEvent))

### paint

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

Renders the caret as a vertical line. If this is reimplemented the damage method should also be reimplemented as it assumes the shape of the caret is a vertical line. Sets the caret color to the value returned by getCaretColor().

If there are multiple text directions present in the associated document, a flag indicating the caret bias will be rendered. This will occur only if the associated document is a subclass of AbstractDocument and there are multiple bidi levels present in the bidi element structure (i.e. the text has multiple directions associated with it).

**Specified by:**[paint](http://docs.google.com/javax/swing/text/Caret.html#paint(java.awt.Graphics)) in interface [Caret](http://docs.google.com/javax/swing/text/Caret.html) **Parameters:**g - the graphics context**See Also:**[damage(java.awt.Rectangle)](http://docs.google.com/javax/swing/text/DefaultCaret.html#damage(java.awt.Rectangle))

### install

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

Called when the UI is being installed into the interface of a JTextComponent. This can be used to gain access to the model that is being navigated by the implementation of this interface. Sets the dot and mark to 0, and establishes document, property change, focus, mouse, and mouse motion listeners.

**Specified by:**[install](http://docs.google.com/javax/swing/text/Caret.html#install(javax.swing.text.JTextComponent)) in interface [Caret](http://docs.google.com/javax/swing/text/Caret.html) **Parameters:**c - the component**See Also:**[Caret.install(javax.swing.text.JTextComponent)](http://docs.google.com/javax/swing/text/Caret.html#install(javax.swing.text.JTextComponent))

### deinstall

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

Called when the UI is being removed from the interface of a JTextComponent. This is used to unregister any listeners that were attached.

**Specified by:**[deinstall](http://docs.google.com/javax/swing/text/Caret.html#deinstall(javax.swing.text.JTextComponent)) in interface [Caret](http://docs.google.com/javax/swing/text/Caret.html) **Parameters:**c - the component**See Also:**[Caret.deinstall(javax.swing.text.JTextComponent)](http://docs.google.com/javax/swing/text/Caret.html#deinstall(javax.swing.text.JTextComponent))

### addChangeListener

public void **addChangeListener**([ChangeListener](http://docs.google.com/javax/swing/event/ChangeListener.html) l)

Adds a listener to track whenever the caret position has been changed.

**Specified by:**[addChangeListener](http://docs.google.com/javax/swing/text/Caret.html#addChangeListener(javax.swing.event.ChangeListener)) in interface [Caret](http://docs.google.com/javax/swing/text/Caret.html) **Parameters:**l - the listener**See Also:**[Caret.addChangeListener(javax.swing.event.ChangeListener)](http://docs.google.com/javax/swing/text/Caret.html#addChangeListener(javax.swing.event.ChangeListener))

### removeChangeListener

public void **removeChangeListener**([ChangeListener](http://docs.google.com/javax/swing/event/ChangeListener.html) l)

Removes a listener that was tracking caret position changes.

**Specified by:**[removeChangeListener](http://docs.google.com/javax/swing/text/Caret.html#removeChangeListener(javax.swing.event.ChangeListener)) in interface [Caret](http://docs.google.com/javax/swing/text/Caret.html) **Parameters:**l - the listener**See Also:**[Caret.removeChangeListener(javax.swing.event.ChangeListener)](http://docs.google.com/javax/swing/text/Caret.html#removeChangeListener(javax.swing.event.ChangeListener))

### getChangeListeners

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

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

**Returns:**all of this caret's ChangeListeners or an empty array if no change listeners are currently registered**Since:** 1.4 **See Also:**[addChangeListener(javax.swing.event.ChangeListener)](http://docs.google.com/javax/swing/text/DefaultCaret.html#addChangeListener(javax.swing.event.ChangeListener)), [removeChangeListener(javax.swing.event.ChangeListener)](http://docs.google.com/javax/swing/text/DefaultCaret.html#removeChangeListener(javax.swing.event.ChangeListener))

### fireStateChanged

protected void **fireStateChanged**()

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 last to first.

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

### getListeners

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

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

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

ChangeListener[] cls = (ChangeListener[])(c.getListeners(ChangeListener.class));

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

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

### setSelectionVisible

public void **setSelectionVisible**(boolean vis)

Changes the selection visibility.

**Specified by:**[setSelectionVisible](http://docs.google.com/javax/swing/text/Caret.html#setSelectionVisible(boolean)) in interface [Caret](http://docs.google.com/javax/swing/text/Caret.html) **Parameters:**vis - the new visibility

### isSelectionVisible

public boolean **isSelectionVisible**()

Checks whether the current selection is visible.

**Specified by:**[isSelectionVisible](http://docs.google.com/javax/swing/text/Caret.html#isSelectionVisible()) in interface [Caret](http://docs.google.com/javax/swing/text/Caret.html) **Returns:**true if the selection is visible

### isActive

public boolean **isActive**()

Determines if the caret is currently active.

This method returns whether or not the Caret is currently in a blinking state. It does not provide information as to whether it is currently blinked on or off. To determine if the caret is currently painted use the isVisible method.

**Returns:**true if active else false**Since:** 1.5 **See Also:**[isVisible()](http://docs.google.com/javax/swing/text/DefaultCaret.html#isVisible())

### isVisible

public boolean **isVisible**()

Indicates whether or not the caret is currently visible. As the caret flashes on and off the return value of this will change between true, when the caret is painted, and false, when the caret is not painted. isActive indicates whether or not the caret is in a blinking state, such that it **can** be visible, and isVisible indicates whether or not the caret **is** actually visible.

Subclasses that wish to render a different flashing caret should override paint and only paint the caret if this method returns true.

**Specified by:**[isVisible](http://docs.google.com/javax/swing/text/Caret.html#isVisible()) in interface [Caret](http://docs.google.com/javax/swing/text/Caret.html) **Returns:**true if visible else false**See Also:**[Caret.isVisible()](http://docs.google.com/javax/swing/text/Caret.html#isVisible()), [isActive()](http://docs.google.com/javax/swing/text/DefaultCaret.html#isActive())

### setVisible

public void **setVisible**(boolean e)

Sets the caret visibility, and repaints the caret. It is important to understand the relationship between this method, isVisible and isActive. Calling this method with a value of true activates the caret blinking. Setting it to false turns it completely off. To determine whether the blinking is active, you should call isActive. In effect, isActive is an appropriate corresponding "getter" method for this one. isVisible can be used to fetch the current visibility status of the caret, meaning whether or not it is currently painted. This status will change as the caret blinks on and off.

Here's a list showing the potential return values of both isActive and isVisible after calling this method:

**setVisible(true)**:

* isActive(): true
* isVisible(): true or false depending on whether or not the caret is blinked on or off

**setVisible(false)**:

* isActive(): false
* isVisible(): false

**Specified by:**[setVisible](http://docs.google.com/javax/swing/text/Caret.html#setVisible(boolean)) in interface [Caret](http://docs.google.com/javax/swing/text/Caret.html) **Parameters:**e - the visibility specifier**See Also:**[isActive()](http://docs.google.com/javax/swing/text/DefaultCaret.html#isActive()), [Caret.setVisible(boolean)](http://docs.google.com/javax/swing/text/Caret.html#setVisible(boolean))

### setBlinkRate

public void **setBlinkRate**(int rate)

Sets the caret blink rate.

**Specified by:**[setBlinkRate](http://docs.google.com/javax/swing/text/Caret.html#setBlinkRate(int)) in interface [Caret](http://docs.google.com/javax/swing/text/Caret.html) **Parameters:**rate - the rate in milliseconds, 0 to stop blinking**See Also:**[Caret.setBlinkRate(int)](http://docs.google.com/javax/swing/text/Caret.html#setBlinkRate(int))

### getBlinkRate

public int **getBlinkRate**()

Gets the caret blink rate.

**Specified by:**[getBlinkRate](http://docs.google.com/javax/swing/text/Caret.html#getBlinkRate()) in interface [Caret](http://docs.google.com/javax/swing/text/Caret.html) **Returns:**the delay in milliseconds. If this is zero the caret will not blink.**See Also:**[Caret.getBlinkRate()](http://docs.google.com/javax/swing/text/Caret.html#getBlinkRate())

### getDot

public int **getDot**()

Fetches the current position of the caret.

**Specified by:**[getDot](http://docs.google.com/javax/swing/text/Caret.html#getDot()) in interface [Caret](http://docs.google.com/javax/swing/text/Caret.html) **Returns:**the position >= 0**See Also:**[Caret.getDot()](http://docs.google.com/javax/swing/text/Caret.html#getDot())

### getMark

public int **getMark**()

Fetches the current position of the mark. If there is a selection, the dot and mark will not be the same.

**Specified by:**[getMark](http://docs.google.com/javax/swing/text/Caret.html#getMark()) in interface [Caret](http://docs.google.com/javax/swing/text/Caret.html) **Returns:**the position >= 0**See Also:**[Caret.getMark()](http://docs.google.com/javax/swing/text/Caret.html#getMark())

### setDot

public void **setDot**(int dot)

Sets the caret position and mark to the specified position, with a forward bias. This implicitly sets the selection range to zero.

**Specified by:**[setDot](http://docs.google.com/javax/swing/text/Caret.html#setDot(int)) in interface [Caret](http://docs.google.com/javax/swing/text/Caret.html) **Parameters:**dot - the position >= 0**See Also:**[setDot(int, Position.Bias)](http://docs.google.com/javax/swing/text/DefaultCaret.html#setDot(int,%20javax.swing.text.Position.Bias)), [Caret.setDot(int)](http://docs.google.com/javax/swing/text/Caret.html#setDot(int))

### moveDot

public void **moveDot**(int dot)

Moves the caret position to the specified position, with a forward bias.

**Specified by:**[moveDot](http://docs.google.com/javax/swing/text/Caret.html#moveDot(int)) in interface [Caret](http://docs.google.com/javax/swing/text/Caret.html) **Parameters:**dot - the position >= 0**See Also:**[moveDot(int, javax.swing.text.Position.Bias)](http://docs.google.com/javax/swing/text/DefaultCaret.html#moveDot(int,%20javax.swing.text.Position.Bias)), [Caret.moveDot(int)](http://docs.google.com/javax/swing/text/Caret.html#moveDot(int))

### moveDot

public void **moveDot**(int dot,  
 [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html) dotBias)

Moves the caret position to the specified position, with the specified bias.

**Parameters:**dot - the position >= 0dotBias - the bias for this position, not null **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the bias is null**Since:** 1.6 **See Also:**[Caret.moveDot(int)](http://docs.google.com/javax/swing/text/Caret.html#moveDot(int))

### setDot

public void **setDot**(int dot,  
 [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html) dotBias)

Sets the caret position and mark to the specified position, with the specified bias. This implicitly sets the selection range to zero.

**Parameters:**dot - the position >= 0dotBias - the bias for this position, not null **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the bias is null**Since:** 1.6 **See Also:**[Caret.setDot(int)](http://docs.google.com/javax/swing/text/Caret.html#setDot(int))

### getDotBias

public [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html) **getDotBias**()

Returns the bias of the caret position.

**Returns:**the bias of the caret position**Since:** 1.6

### getMarkBias

public [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html) **getMarkBias**()

Returns the bias of the mark.

**Returns:**the bias of the mark**Since:** 1.6

### setMagicCaretPosition

public void **setMagicCaretPosition**([Point](http://docs.google.com/java/awt/Point.html) p)

Saves the current caret position. This is used when caret up/down actions occur, moving between lines that have uneven end positions.

**Specified by:**[setMagicCaretPosition](http://docs.google.com/javax/swing/text/Caret.html#setMagicCaretPosition(java.awt.Point)) in interface [Caret](http://docs.google.com/javax/swing/text/Caret.html) **Parameters:**p - the position**See Also:**[getMagicCaretPosition()](http://docs.google.com/javax/swing/text/DefaultCaret.html#getMagicCaretPosition())

### getMagicCaretPosition

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

Gets the saved caret position.

**Specified by:**[getMagicCaretPosition](http://docs.google.com/javax/swing/text/Caret.html#getMagicCaretPosition()) in interface [Caret](http://docs.google.com/javax/swing/text/Caret.html) **Returns:**the position see #setMagicCaretPosition**See Also:**[Caret.setMagicCaretPosition(java.awt.Point)](http://docs.google.com/javax/swing/text/Caret.html#setMagicCaretPosition(java.awt.Point))

### equals

public boolean **equals**([Object](http://docs.google.com/java/lang/Object.html) obj)

Compares this object to the specified object. The superclass behavior of comparing rectangles is not desired, so this is changed to the Object behavior.

**Overrides:**[equals](http://docs.google.com/java/awt/Rectangle.html#equals(java.lang.Object)) in class [Rectangle](http://docs.google.com/java/awt/Rectangle.html) **Parameters:**obj - the object to compare this font with **Returns:**true if the objects are equal; false otherwise**See Also:**[Object.hashCode()](http://docs.google.com/java/lang/Object.html#hashCode()), [Hashtable](http://docs.google.com/java/util/Hashtable.html)

### toString

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

**Description copied from class:** [**Rectangle**](http://docs.google.com/java/awt/Rectangle.html#toString()) Returns a String representing this Rectangle and its values.

**Overrides:**[toString](http://docs.google.com/java/awt/Rectangle.html#toString()) in class [Rectangle](http://docs.google.com/java/awt/Rectangle.html) **Returns:**a String representing this Rectangle object's coordinate and size values.

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/DefaultCaret.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/DateFormatter.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/text/DefaultEditorKit.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/text/DefaultCaret.html)    [**NO FRAMES**](http://docs.google.com/DefaultCaret.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#2et92p0) | [FIELD](#tyjcwt) | [CONSTR](#4d34og8) | [METHOD](#2s8eyo1) | DETAIL: [FIELD](#1ksv4uv) | [CONSTR](#4i7ojhp) | [METHOD](#1ci93xb) |

[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).