| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Utilities.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/TextAction.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/text/View.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/text/Utilities.html)    [**NO FRAMES**](http://docs.google.com/Utilities.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#3dy6vkm) | [METHOD](#4d34og8) |

## **javax.swing.text**

Class Utilities

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 **javax.swing.text.Utilities**

public class **Utilities**extends [Object](http://docs.google.com/java/lang/Object.html)

A collection of methods to deal with various text related activities.

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

| **Method Summary** | |
| --- | --- |
| static int | [**drawTabbedText**](http://docs.google.com/javax/swing/text/Utilities.html#drawTabbedText(javax.swing.text.Segment,%20int,%20int,%20java.awt.Graphics,%20javax.swing.text.TabExpander,%20int))([Segment](http://docs.google.com/javax/swing/text/Segment.html) s, int x, int y, [Graphics](http://docs.google.com/java/awt/Graphics.html) g, [TabExpander](http://docs.google.com/javax/swing/text/TabExpander.html) e, int startOffset)            Draws the given text, expanding any tabs that are contained using the given tab expansion technique. |
| static int | [**getBreakLocation**](http://docs.google.com/javax/swing/text/Utilities.html#getBreakLocation(javax.swing.text.Segment,%20java.awt.FontMetrics,%20int,%20int,%20javax.swing.text.TabExpander,%20int))([Segment](http://docs.google.com/javax/swing/text/Segment.html) s, [FontMetrics](http://docs.google.com/java/awt/FontMetrics.html) metrics, int x0, int x, [TabExpander](http://docs.google.com/javax/swing/text/TabExpander.html) e, int startOffset)            Determine where to break the given text to fit within the given span. |
| static int | [**getNextWord**](http://docs.google.com/javax/swing/text/Utilities.html#getNextWord(javax.swing.text.JTextComponent,%20int))([JTextComponent](http://docs.google.com/javax/swing/text/JTextComponent.html) c, int offs)            Determines the start of the next word for the given location. |
| static [Element](http://docs.google.com/javax/swing/text/Element.html) | [**getParagraphElement**](http://docs.google.com/javax/swing/text/Utilities.html#getParagraphElement(javax.swing.text.JTextComponent,%20int))([JTextComponent](http://docs.google.com/javax/swing/text/JTextComponent.html) c, int offs)            Determines the element to use for a paragraph/line. |
| static int | [**getPositionAbove**](http://docs.google.com/javax/swing/text/Utilities.html#getPositionAbove(javax.swing.text.JTextComponent,%20int,%20int))([JTextComponent](http://docs.google.com/javax/swing/text/JTextComponent.html) c, int offs, int x)            Determines the position in the model that is closest to the given view location in the row above. |
| static int | [**getPositionBelow**](http://docs.google.com/javax/swing/text/Utilities.html#getPositionBelow(javax.swing.text.JTextComponent,%20int,%20int))([JTextComponent](http://docs.google.com/javax/swing/text/JTextComponent.html) c, int offs, int x)            Determines the position in the model that is closest to the given view location in the row below. |
| static int | [**getPreviousWord**](http://docs.google.com/javax/swing/text/Utilities.html#getPreviousWord(javax.swing.text.JTextComponent,%20int))([JTextComponent](http://docs.google.com/javax/swing/text/JTextComponent.html) c, int offs)            Determine the start of the prev word for the given location. |
| static int | [**getRowEnd**](http://docs.google.com/javax/swing/text/Utilities.html#getRowEnd(javax.swing.text.JTextComponent,%20int))([JTextComponent](http://docs.google.com/javax/swing/text/JTextComponent.html) c, int offs)            Determines the ending row model position of the row that contains the specified model position. |
| static int | [**getRowStart**](http://docs.google.com/javax/swing/text/Utilities.html#getRowStart(javax.swing.text.JTextComponent,%20int))([JTextComponent](http://docs.google.com/javax/swing/text/JTextComponent.html) c, int offs)            Determines the starting row model position of the row that contains the specified model position. |
| static int | [**getTabbedTextOffset**](http://docs.google.com/javax/swing/text/Utilities.html#getTabbedTextOffset(javax.swing.text.Segment,%20java.awt.FontMetrics,%20int,%20int,%20javax.swing.text.TabExpander,%20int))([Segment](http://docs.google.com/javax/swing/text/Segment.html) s, [FontMetrics](http://docs.google.com/java/awt/FontMetrics.html) metrics, int x0, int x, [TabExpander](http://docs.google.com/javax/swing/text/TabExpander.html) e, int startOffset)            Determines the relative offset into the given text that best represents the given span in the view coordinate system. |
| static int | [**getTabbedTextOffset**](http://docs.google.com/javax/swing/text/Utilities.html#getTabbedTextOffset(javax.swing.text.Segment,%20java.awt.FontMetrics,%20int,%20int,%20javax.swing.text.TabExpander,%20int,%20boolean))([Segment](http://docs.google.com/javax/swing/text/Segment.html) s, [FontMetrics](http://docs.google.com/java/awt/FontMetrics.html) metrics, int x0, int x, [TabExpander](http://docs.google.com/javax/swing/text/TabExpander.html) e, int startOffset, boolean round) |
| static int | [**getTabbedTextWidth**](http://docs.google.com/javax/swing/text/Utilities.html#getTabbedTextWidth(javax.swing.text.Segment,%20java.awt.FontMetrics,%20int,%20javax.swing.text.TabExpander,%20int))([Segment](http://docs.google.com/javax/swing/text/Segment.html) s, [FontMetrics](http://docs.google.com/java/awt/FontMetrics.html) metrics, int x, [TabExpander](http://docs.google.com/javax/swing/text/TabExpander.html) e, int startOffset)            Determines the width of the given segment of text taking tabs into consideration. |
| static int | [**getWordEnd**](http://docs.google.com/javax/swing/text/Utilities.html#getWordEnd(javax.swing.text.JTextComponent,%20int))([JTextComponent](http://docs.google.com/javax/swing/text/JTextComponent.html) c, int offs)            Determines the end of a word for the given location. |
| static int | [**getWordStart**](http://docs.google.com/javax/swing/text/Utilities.html#getWordStart(javax.swing.text.JTextComponent,%20int))([JTextComponent](http://docs.google.com/javax/swing/text/JTextComponent.html) c, int offs)            Determines the start of a word for the given model location. |

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

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

### Utilities

public **Utilities**()

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

### drawTabbedText

public static final int **drawTabbedText**([Segment](http://docs.google.com/javax/swing/text/Segment.html) s,  
 int x,  
 int y,  
 [Graphics](http://docs.google.com/java/awt/Graphics.html) g,  
 [TabExpander](http://docs.google.com/javax/swing/text/TabExpander.html) e,  
 int startOffset)

Draws the given text, expanding any tabs that are contained using the given tab expansion technique. This particular implementation renders in a 1.1 style coordinate system where ints are used and 72dpi is assumed.

**Parameters:**s - the source of the textx - the X origin >= 0y - the Y origin >= 0g - the graphics contexte - how to expand the tabs. If this value is null, tabs will be expanded as a space character.startOffset - starting offset of the text in the document >= 0 **Returns:**the X location at the end of the rendered text

### getTabbedTextWidth

public static final int **getTabbedTextWidth**([Segment](http://docs.google.com/javax/swing/text/Segment.html) s,  
 [FontMetrics](http://docs.google.com/java/awt/FontMetrics.html) metrics,  
 int x,  
 [TabExpander](http://docs.google.com/javax/swing/text/TabExpander.html) e,  
 int startOffset)

Determines the width of the given segment of text taking tabs into consideration. This is implemented in a 1.1 style coordinate system where ints are used and 72dpi is assumed.

**Parameters:**s - the source of the textmetrics - the font metrics to use for the calculationx - the X origin >= 0e - how to expand the tabs. If this value is null, tabs will be expanded as a space character.startOffset - starting offset of the text in the document >= 0 **Returns:**the width of the text

### getTabbedTextOffset

public static final int **getTabbedTextOffset**([Segment](http://docs.google.com/javax/swing/text/Segment.html) s,  
 [FontMetrics](http://docs.google.com/java/awt/FontMetrics.html) metrics,  
 int x0,  
 int x,  
 [TabExpander](http://docs.google.com/javax/swing/text/TabExpander.html) e,  
 int startOffset)

Determines the relative offset into the given text that best represents the given span in the view coordinate system. This is implemented in a 1.1 style coordinate system where ints are used and 72dpi is assumed.

**Parameters:**s - the source of the textmetrics - the font metrics to use for the calculationx0 - the starting view location representing the start of the given text >= 0.x - the target view location to translate to an offset into the text >= 0.e - how to expand the tabs. If this value is null, tabs will be expanded as a space character.startOffset - starting offset of the text in the document >= 0 **Returns:**the offset into the text >= 0

### getTabbedTextOffset

public static final int **getTabbedTextOffset**([Segment](http://docs.google.com/javax/swing/text/Segment.html) s,  
 [FontMetrics](http://docs.google.com/java/awt/FontMetrics.html) metrics,  
 int x0,  
 int x,  
 [TabExpander](http://docs.google.com/javax/swing/text/TabExpander.html) e,  
 int startOffset,  
 boolean round)

### getBreakLocation

public static final int **getBreakLocation**([Segment](http://docs.google.com/javax/swing/text/Segment.html) s,  
 [FontMetrics](http://docs.google.com/java/awt/FontMetrics.html) metrics,  
 int x0,  
 int x,  
 [TabExpander](http://docs.google.com/javax/swing/text/TabExpander.html) e,  
 int startOffset)

Determine where to break the given text to fit within the given span. This tries to find a word boundary.

**Parameters:**s - the source of the textmetrics - the font metrics to use for the calculationx0 - the starting view location representing the start of the given text.x - the target view location to translate to an offset into the text.e - how to expand the tabs. If this value is null, tabs will be expanded as a space character.startOffset - starting offset in the document of the text **Returns:**the offset into the given text

### getRowStart

public static final int **getRowStart**([JTextComponent](http://docs.google.com/javax/swing/text/JTextComponent.html) c,  
 int offs)  
 throws [BadLocationException](http://docs.google.com/javax/swing/text/BadLocationException.html)

Determines the starting row model position of the row that contains the specified model position. The component given must have a size to compute the result. If the component doesn't have a size a value of -1 will be returned.

**Parameters:**c - the editoroffs - the offset in the document >= 0 **Returns:**the position >= 0 if the request can be computed, otherwise a value of -1 will be returned. **Throws:** [BadLocationException](http://docs.google.com/javax/swing/text/BadLocationException.html) - if the offset is out of range

### getRowEnd

public static final int **getRowEnd**([JTextComponent](http://docs.google.com/javax/swing/text/JTextComponent.html) c,  
 int offs)  
 throws [BadLocationException](http://docs.google.com/javax/swing/text/BadLocationException.html)

Determines the ending row model position of the row that contains the specified model position. The component given must have a size to compute the result. If the component doesn't have a size a value of -1 will be returned.

**Parameters:**c - the editoroffs - the offset in the document >= 0 **Returns:**the position >= 0 if the request can be computed, otherwise a value of -1 will be returned. **Throws:** [BadLocationException](http://docs.google.com/javax/swing/text/BadLocationException.html) - if the offset is out of range

### getPositionAbove

public static final int **getPositionAbove**([JTextComponent](http://docs.google.com/javax/swing/text/JTextComponent.html) c,  
 int offs,  
 int x)  
 throws [BadLocationException](http://docs.google.com/javax/swing/text/BadLocationException.html)

Determines the position in the model that is closest to the given view location in the row above. The component given must have a size to compute the result. If the component doesn't have a size a value of -1 will be returned.

**Parameters:**c - the editoroffs - the offset in the document >= 0x - the X coordinate >= 0 **Returns:**the position >= 0 if the request can be computed, otherwise a value of -1 will be returned. **Throws:** [BadLocationException](http://docs.google.com/javax/swing/text/BadLocationException.html) - if the offset is out of range

### getPositionBelow

public static final int **getPositionBelow**([JTextComponent](http://docs.google.com/javax/swing/text/JTextComponent.html) c,  
 int offs,  
 int x)  
 throws [BadLocationException](http://docs.google.com/javax/swing/text/BadLocationException.html)

Determines the position in the model that is closest to the given view location in the row below. The component given must have a size to compute the result. If the component doesn't have a size a value of -1 will be returned.

**Parameters:**c - the editoroffs - the offset in the document >= 0x - the X coordinate >= 0 **Returns:**the position >= 0 if the request can be computed, otherwise a value of -1 will be returned. **Throws:** [BadLocationException](http://docs.google.com/javax/swing/text/BadLocationException.html) - if the offset is out of range

### getWordStart

public static final int **getWordStart**([JTextComponent](http://docs.google.com/javax/swing/text/JTextComponent.html) c,  
 int offs)  
 throws [BadLocationException](http://docs.google.com/javax/swing/text/BadLocationException.html)

Determines the start of a word for the given model location. Uses BreakIterator.getWordInstance() to actually get the words.

**Parameters:**c - the editoroffs - the offset in the document >= 0 **Returns:**the location in the model of the word start >= 0 **Throws:** [BadLocationException](http://docs.google.com/javax/swing/text/BadLocationException.html) - if the offset is out of range

### getWordEnd

public static final int **getWordEnd**([JTextComponent](http://docs.google.com/javax/swing/text/JTextComponent.html) c,  
 int offs)  
 throws [BadLocationException](http://docs.google.com/javax/swing/text/BadLocationException.html)

Determines the end of a word for the given location. Uses BreakIterator.getWordInstance() to actually get the words.

**Parameters:**c - the editoroffs - the offset in the document >= 0 **Returns:**the location in the model of the word end >= 0 **Throws:** [BadLocationException](http://docs.google.com/javax/swing/text/BadLocationException.html) - if the offset is out of range

### getNextWord

public static final int **getNextWord**([JTextComponent](http://docs.google.com/javax/swing/text/JTextComponent.html) c,  
 int offs)  
 throws [BadLocationException](http://docs.google.com/javax/swing/text/BadLocationException.html)

Determines the start of the next word for the given location. Uses BreakIterator.getWordInstance() to actually get the words.

**Parameters:**c - the editoroffs - the offset in the document >= 0 **Returns:**the location in the model of the word start >= 0 **Throws:** [BadLocationException](http://docs.google.com/javax/swing/text/BadLocationException.html) - if the offset is out of range

### getPreviousWord

public static final int **getPreviousWord**([JTextComponent](http://docs.google.com/javax/swing/text/JTextComponent.html) c,  
 int offs)  
 throws [BadLocationException](http://docs.google.com/javax/swing/text/BadLocationException.html)

Determine the start of the prev word for the given location. Uses BreakIterator.getWordInstance() to actually get the words.

**Parameters:**c - the editoroffs - the offset in the document >= 0 **Returns:**the location in the model of the word start >= 0 **Throws:** [BadLocationException](http://docs.google.com/javax/swing/text/BadLocationException.html) - if the offset is out of range

### getParagraphElement

public static final [Element](http://docs.google.com/javax/swing/text/Element.html) **getParagraphElement**([JTextComponent](http://docs.google.com/javax/swing/text/JTextComponent.html) c,  
 int offs)

Determines the element to use for a paragraph/line.

**Parameters:**c - the editoroffs - the starting offset in the document >= 0 **Returns:**the element

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Utilities.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/TextAction.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/text/View.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/text/Utilities.html)    [**NO FRAMES**](http://docs.google.com/Utilities.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#3dy6vkm) | [METHOD](#4d34og8) |

[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).