| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/LineBreakMeasurer.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/java/awt/font/LayoutPath.html)   [**NEXT CLASS**](http://docs.google.com/java/awt/font/LineMetrics.html) | [**FRAMES**](http://docs.google.com/index.html?java/awt/font/LineBreakMeasurer.html)    [**NO FRAMES**](http://docs.google.com/LineBreakMeasurer.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#3dy6vkm) | [METHOD](#2s8eyo1) |

## **java.awt.font**

Class LineBreakMeasurer

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 **java.awt.font.LineBreakMeasurer**

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

The LineBreakMeasurer class allows styled text to be broken into lines (or segments) that fit within a particular visual advance. This is useful for clients who wish to display a paragraph of text that fits within a specific width, called the **wrapping width**.

LineBreakMeasurer is constructed with an iterator over styled text. The iterator's range should be a single paragraph in the text. LineBreakMeasurer maintains a position in the text for the start of the next text segment. Initially, this position is the start of text. Paragraphs are assigned an overall direction (either left-to-right or right-to-left) according to the bidirectional formatting rules. All segments obtained from a paragraph have the same direction as the paragraph.

Segments of text are obtained by calling the method nextLayout, which returns a [TextLayout](http://docs.google.com/java/awt/font/TextLayout.html) representing the text that fits within the wrapping width. The nextLayout method moves the current position to the end of the layout returned from nextLayout.

LineBreakMeasurer implements the most commonly used line-breaking policy: Every word that fits within the wrapping width is placed on the line. If the first word does not fit, then all of the characters that fit within the wrapping width are placed on the line. At least one character is placed on each line.

The TextLayout instances returned by LineBreakMeasurer treat tabs like 0-width spaces. Clients who wish to obtain tab-delimited segments for positioning should use the overload of nextLayout which takes a limiting offset in the text. The limiting offset should be the first character after the tab. The TextLayout objects returned from this method end at the limit provided (or before, if the text between the current position and the limit won't fit entirely within the wrapping width).

Clients who are laying out tab-delimited text need a slightly different line-breaking policy after the first segment has been placed on a line. Instead of fitting partial words in the remaining space, they should place words which don't fit in the remaining space entirely on the next line. This change of policy can be requested in the overload of nextLayout which takes a boolean parameter. If this parameter is true, nextLayout returns null if the first word won't fit in the given space. See the tab sample below.

In general, if the text used to construct the LineBreakMeasurer changes, a new LineBreakMeasurer must be constructed to reflect the change. (The old LineBreakMeasurer continues to function properly, but it won't be aware of the text change.) Nevertheless, if the text change is the insertion or deletion of a single character, an existing LineBreakMeasurer can be 'updated' by calling insertChar or deleteChar. Updating an existing LineBreakMeasurer is much faster than creating a new one. Clients who modify text based on user typing should take advantage of these methods.

**Examples**:

Rendering a paragraph in a component

public void paint(Graphics graphics) {  
  
 Point2D pen = new Point2D(10, 20);  
 Graphics2D g2d = (Graphics2D)graphics;  
 FontRenderContext frc = g2d.getFontRenderContext();  
  
 // let styledText be an AttributedCharacterIterator containing at least  
 // one character  
  
 LineBreakMeasurer measurer = new LineBreakMeasurer(styledText, frc);  
 float wrappingWidth = getSize().width - 15;  
  
 while (measurer.getPosition() < fStyledText.length()) {  
  
 TextLayout layout = measurer.nextLayout(wrappingWidth);  
  
 pen.y += (layout.getAscent());  
 float dx = layout.isLeftToRight() ?  
 0 : (wrappingWidth - layout.getAdvance());  
  
 layout.draw(graphics, pen.x + dx, pen.y);  
 pen.y += layout.getDescent() + layout.getLeading();  
 }  
 }

Rendering text with tabs. For simplicity, the overall text direction is assumed to be left-to-right

public void paint(Graphics graphics) {  
  
 float leftMargin = 10, rightMargin = 310;  
 float[] tabStops = { 100, 250 };  
  
 // assume styledText is an AttributedCharacterIterator, and the number  
 // of tabs in styledText is tabCount  
  
 int[] tabLocations = new int[tabCount+1];  
  
 int i = 0;  
 for (char c = styledText.first(); c != styledText.DONE; c = styledText.next()) {  
 if (c == '\t') {  
 tabLocations[i++] = styledText.getIndex();  
 }  
 }  
 tabLocations[tabCount] = styledText.getEndIndex() - 1;  
  
 // Now tabLocations has an entry for every tab's offset in  
 // the text. For convenience, the last entry is tabLocations  
 // is the offset of the last character in the text.  
  
 LineBreakMeasurer measurer = new LineBreakMeasurer(styledText);  
 int currentTab = 0;  
 float verticalPos = 20;  
  
 while (measurer.getPosition() < styledText.getEndIndex()) {  
  
 // Lay out and draw each line. All segments on a line  
 // must be computed before any drawing can occur, since  
 // we must know the largest ascent on the line.  
 // TextLayouts are computed and stored in a Vector;  
 // their horizontal positions are stored in a parallel  
 // Vector.  
  
 // lineContainsText is true after first segment is drawn  
 boolean lineContainsText = false;  
 boolean lineComplete = false;  
 float maxAscent = 0, maxDescent = 0;  
 float horizontalPos = leftMargin;  
 Vector layouts = new Vector(1);  
 Vector penPositions = new Vector(1);  
  
 while (!lineComplete) {  
 float wrappingWidth = rightMargin - horizontalPos;  
 TextLayout layout =  
 measurer.nextLayout(wrappingWidth,  
 tabLocations[currentTab]+1,  
 lineContainsText);  
  
 // layout can be null if lineContainsText is true  
 if (layout != null) {  
 layouts.addElement(layout);  
 penPositions.addElement(new Float(horizontalPos));  
 horizontalPos += layout.getAdvance();  
 maxAscent = Math.max(maxAscent, layout.getAscent());  
 maxDescent = Math.max(maxDescent,  
 layout.getDescent() + layout.getLeading());  
 } else {  
 lineComplete = true;  
 }  
  
 lineContainsText = true;  
  
 if (measurer.getPosition() == tabLocations[currentTab]+1) {  
 currentTab++;  
 }  
  
 if (measurer.getPosition() == styledText.getEndIndex())  
 lineComplete = true;  
 else if (horizontalPos >= tabStops[tabStops.length-1])  
 lineComplete = true;  
  
 if (!lineComplete) {  
 // move to next tab stop  
 int j;  
 for (j=0; horizontalPos >= tabStops[j]; j++) {}  
 horizontalPos = tabStops[j];  
 }  
 }  
  
 verticalPos += maxAscent;  
  
 Enumeration layoutEnum = layouts.elements();  
 Enumeration positionEnum = penPositions.elements();  
  
 // now iterate through layouts and draw them  
 while (layoutEnum.hasMoreElements()) {  
 TextLayout nextLayout = (TextLayout) layoutEnum.nextElement();  
 Float nextPosition = (Float) positionEnum.nextElement();  
 nextLayout.draw(graphics, nextPosition.floatValue(), verticalPos);  
 }  
  
 verticalPos += maxDescent;  
 }  
 }

**See Also:**[TextLayout](http://docs.google.com/java/awt/font/TextLayout.html)

| **Constructor Summary** | |
| --- | --- |
| [**LineBreakMeasurer**](http://docs.google.com/java/awt/font/LineBreakMeasurer.html#LineBreakMeasurer(java.text.AttributedCharacterIterator,%20java.text.BreakIterator,%20java.awt.font.FontRenderContext))([AttributedCharacterIterator](http://docs.google.com/java/text/AttributedCharacterIterator.html) text, [BreakIterator](http://docs.google.com/java/text/BreakIterator.html) breakIter, [FontRenderContext](http://docs.google.com/java/awt/font/FontRenderContext.html) frc)            Constructs a LineBreakMeasurer for the specified text. |
| [**LineBreakMeasurer**](http://docs.google.com/java/awt/font/LineBreakMeasurer.html#LineBreakMeasurer(java.text.AttributedCharacterIterator,%20java.awt.font.FontRenderContext))([AttributedCharacterIterator](http://docs.google.com/java/text/AttributedCharacterIterator.html) text, [FontRenderContext](http://docs.google.com/java/awt/font/FontRenderContext.html) frc)            Constructs a LineBreakMeasurer for the specified text. |

| **Method Summary** | |
| --- | --- |
| void | [**deleteChar**](http://docs.google.com/java/awt/font/LineBreakMeasurer.html#deleteChar(java.text.AttributedCharacterIterator,%20int))([AttributedCharacterIterator](http://docs.google.com/java/text/AttributedCharacterIterator.html) newParagraph, int deletePos)            Updates this LineBreakMeasurer after a single character is deleted from the text, and sets the current position to the beginning of the paragraph. |
| int | [**getPosition**](http://docs.google.com/java/awt/font/LineBreakMeasurer.html#getPosition())()            Returns the current position of this LineBreakMeasurer. |
| void | [**insertChar**](http://docs.google.com/java/awt/font/LineBreakMeasurer.html#insertChar(java.text.AttributedCharacterIterator,%20int))([AttributedCharacterIterator](http://docs.google.com/java/text/AttributedCharacterIterator.html) newParagraph, int insertPos)            Updates this LineBreakMeasurer after a single character is inserted into the text, and sets the current position to the beginning of the paragraph. |
| [TextLayout](http://docs.google.com/java/awt/font/TextLayout.html) | [**nextLayout**](http://docs.google.com/java/awt/font/LineBreakMeasurer.html#nextLayout(float))(float wrappingWidth)            Returns the next layout, and updates the current position. |
| [TextLayout](http://docs.google.com/java/awt/font/TextLayout.html) | [**nextLayout**](http://docs.google.com/java/awt/font/LineBreakMeasurer.html#nextLayout(float,%20int,%20boolean))(float wrappingWidth, int offsetLimit, boolean requireNextWord)            Returns the next layout, and updates the current position. |
| int | [**nextOffset**](http://docs.google.com/java/awt/font/LineBreakMeasurer.html#nextOffset(float))(float wrappingWidth)            Returns the position at the end of the next layout. |
| int | [**nextOffset**](http://docs.google.com/java/awt/font/LineBreakMeasurer.html#nextOffset(float,%20int,%20boolean))(float wrappingWidth, int offsetLimit, boolean requireNextWord)            Returns the position at the end of the next layout. |
| void | [**setPosition**](http://docs.google.com/java/awt/font/LineBreakMeasurer.html#setPosition(int))(int newPosition)            Sets the current position of this LineBreakMeasurer. |

| **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** |
| --- |

### LineBreakMeasurer

public **LineBreakMeasurer**([AttributedCharacterIterator](http://docs.google.com/java/text/AttributedCharacterIterator.html) text,  
 [FontRenderContext](http://docs.google.com/java/awt/font/FontRenderContext.html) frc)

Constructs a LineBreakMeasurer for the specified text.

**Parameters:**text - the text for which this LineBreakMeasurer produces TextLayout objects; the text must contain at least one character; if the text available through iter changes, further calls to this LineBreakMeasurer instance are undefined (except, in some cases, when insertChar or deleteChar are invoked afterward - see below)frc - contains information about a graphics device which is needed to measure the text correctly; text measurements can vary slightly depending on the device resolution, and attributes such as antialiasing; this parameter does not specify a translation between the LineBreakMeasurer and user space**See Also:**[insertChar(java.text.AttributedCharacterIterator, int)](http://docs.google.com/java/awt/font/LineBreakMeasurer.html#insertChar(java.text.AttributedCharacterIterator,%20int)), [deleteChar(java.text.AttributedCharacterIterator, int)](http://docs.google.com/java/awt/font/LineBreakMeasurer.html#deleteChar(java.text.AttributedCharacterIterator,%20int))

### LineBreakMeasurer

public **LineBreakMeasurer**([AttributedCharacterIterator](http://docs.google.com/java/text/AttributedCharacterIterator.html) text,  
 [BreakIterator](http://docs.google.com/java/text/BreakIterator.html) breakIter,  
 [FontRenderContext](http://docs.google.com/java/awt/font/FontRenderContext.html) frc)

Constructs a LineBreakMeasurer for the specified text.

**Parameters:**text - the text for which this LineBreakMeasurer produces TextLayout objects; the text must contain at least one character; if the text available through iter changes, further calls to this LineBreakMeasurer instance are undefined (except, in some cases, when insertChar or deleteChar are invoked afterward - see below)breakIter - the [BreakIterator](http://docs.google.com/java/text/BreakIterator.html) which defines line breaksfrc - contains information about a graphics device which is needed to measure the text correctly; text measurements can vary slightly depending on the device resolution, and attributes such as antialiasing; this parameter does not specify a translation between the LineBreakMeasurer and user space **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the text has less than one character**See Also:**[insertChar(java.text.AttributedCharacterIterator, int)](http://docs.google.com/java/awt/font/LineBreakMeasurer.html#insertChar(java.text.AttributedCharacterIterator,%20int)), [deleteChar(java.text.AttributedCharacterIterator, int)](http://docs.google.com/java/awt/font/LineBreakMeasurer.html#deleteChar(java.text.AttributedCharacterIterator,%20int))

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

### nextOffset

public int **nextOffset**(float wrappingWidth)

Returns the position at the end of the next layout. Does NOT update the current position of this LineBreakMeasurer.

**Parameters:**wrappingWidth - the maximum visible advance permitted for the text in the next layout **Returns:**an offset in the text representing the limit of the next TextLayout.

### nextOffset

public int **nextOffset**(float wrappingWidth,  
 int offsetLimit,  
 boolean requireNextWord)

Returns the position at the end of the next layout. Does NOT update the current position of this LineBreakMeasurer.

**Parameters:**wrappingWidth - the maximum visible advance permitted for the text in the next layoutoffsetLimit - the first character that can not be included in the next layout, even if the text after the limit would fit within the wrapping width; offsetLimit must be greater than the current positionrequireNextWord - if true, the current position that is returned if the entire next word does not fit within wrappingWidth; if false, the offset returned is at least one greater than the current position **Returns:**an offset in the text representing the limit of the next TextLayout

### nextLayout

public [TextLayout](http://docs.google.com/java/awt/font/TextLayout.html) **nextLayout**(float wrappingWidth)

Returns the next layout, and updates the current position.

**Parameters:**wrappingWidth - the maximum visible advance permitted for the text in the next layout **Returns:**a TextLayout, beginning at the current position, which represents the next line fitting within wrappingWidth

### nextLayout

public [TextLayout](http://docs.google.com/java/awt/font/TextLayout.html) **nextLayout**(float wrappingWidth,  
 int offsetLimit,  
 boolean requireNextWord)

Returns the next layout, and updates the current position.

**Parameters:**wrappingWidth - the maximum visible advance permitted for the text in the next layoutoffsetLimit - the first character that can not be included in the next layout, even if the text after the limit would fit within the wrapping width; offsetLimit must be greater than the current positionrequireNextWord - if true, and if the entire word at the current position does not fit within the wrapping width, null is returned. If false, a valid layout is returned that includes at least the character at the current position **Returns:**a TextLayout, beginning at the current position, that represents the next line fitting within wrappingWidth. If the current position is at the end of the text used by this LineBreakMeasurer, null is returned

### getPosition

public int **getPosition**()

Returns the current position of this LineBreakMeasurer.

**Returns:**the current position of this LineBreakMeasurer**See Also:**[setPosition(int)](http://docs.google.com/java/awt/font/LineBreakMeasurer.html#setPosition(int))

### setPosition

public void **setPosition**(int newPosition)

Sets the current position of this LineBreakMeasurer.

**Parameters:**newPosition - the current position of this LineBreakMeasurer; the position should be within the text used to construct this LineBreakMeasurer (or in the text most recently passed to insertChar or deleteChar**See Also:**[getPosition()](http://docs.google.com/java/awt/font/LineBreakMeasurer.html#getPosition())

### insertChar

public void **insertChar**([AttributedCharacterIterator](http://docs.google.com/java/text/AttributedCharacterIterator.html) newParagraph,  
 int insertPos)

Updates this LineBreakMeasurer after a single character is inserted into the text, and sets the current position to the beginning of the paragraph.

**Parameters:**newParagraph - the text after the insertioninsertPos - the position in the text at which the character is inserted **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if insertPos is less than the start of newParagraph or greater than or equal to the end of newParagraph [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if newParagraph is null**See Also:**[deleteChar(java.text.AttributedCharacterIterator, int)](http://docs.google.com/java/awt/font/LineBreakMeasurer.html#deleteChar(java.text.AttributedCharacterIterator,%20int))

### deleteChar

public void **deleteChar**([AttributedCharacterIterator](http://docs.google.com/java/text/AttributedCharacterIterator.html) newParagraph,  
 int deletePos)

Updates this LineBreakMeasurer after a single character is deleted from the text, and sets the current position to the beginning of the paragraph.

**Parameters:**newParagraph - the text after the deletiondeletePos - the position in the text at which the character is deleted **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if deletePos is less than the start of newParagraph or greater than the end of newParagraph [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if newParagraph is null**See Also:**[insertChar(java.text.AttributedCharacterIterator, int)](http://docs.google.com/java/awt/font/LineBreakMeasurer.html#insertChar(java.text.AttributedCharacterIterator,%20int))

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/LineBreakMeasurer.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/java/awt/font/LayoutPath.html)   [**NEXT CLASS**](http://docs.google.com/java/awt/font/LineMetrics.html) | [**FRAMES**](http://docs.google.com/index.html?java/awt/font/LineBreakMeasurer.html)    [**NO FRAMES**](http://docs.google.com/LineBreakMeasurer.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#3dy6vkm) | [METHOD](#2s8eyo1) |

[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).