| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/TextLayout.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/TextHitInfo.html)   [**NEXT CLASS**](http://docs.google.com/java/awt/font/TextLayout.CaretPolicy.html) | [**FRAMES**](http://docs.google.com/index.html?java/awt/font/TextLayout.html)    [**NO FRAMES**](http://docs.google.com/TextLayout.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | [FIELD](#2et92p0) | [CONSTR](#tyjcwt) | [METHOD](#3dy6vkm) | DETAIL: [FIELD](#4d34og8) | [CONSTR](#17dp8vu) | [METHOD](#35nkun2) |

## **java.awt.font**

Class TextLayout

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

**All Implemented Interfaces:** [Cloneable](http://docs.google.com/java/lang/Cloneable.html)

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

TextLayout is an immutable graphical representation of styled character data.

It provides the following capabilities:

* implicit bidirectional analysis and reordering,
* cursor positioning and movement, including split cursors for mixed directional text,
* highlighting, including both logical and visual highlighting for mixed directional text,
* multiple baselines (roman, hanging, and centered),
* hit testing,
* justification,
* default font substitution,
* metric information such as ascent, descent, and advance, and
* rendering

A TextLayout object can be rendered using its draw method.

TextLayout can be constructed either directly or through the use of a [LineBreakMeasurer](http://docs.google.com/java/awt/font/LineBreakMeasurer.html). When constructed directly, the source text represents a single paragraph. LineBreakMeasurer allows styled text to be broken into lines that fit within a particular width. See the LineBreakMeasurer documentation for more information.

TextLayout construction logically proceeds as follows:

* paragraph attributes are extracted and examined,
* text is analyzed for bidirectional reordering, and reordering information is computed if needed,
* text is segmented into style runs
* fonts are chosen for style runs, first by using a font if the attribute [TextAttribute.FONT](http://docs.google.com/java/awt/font/TextAttribute.html#FONT) is present, otherwise by computing a default font using the attributes that have been defined
* if text is on multiple baselines, the runs or subruns are further broken into subruns sharing a common baseline,
* glyphvectors are generated for each run using the chosen font,
* final bidirectional reordering is performed on the glyphvectors

All graphical information returned from a TextLayout object's methods is relative to the origin of the TextLayout, which is the intersection of the TextLayout object's baseline with its left edge. Also, coordinates passed into a TextLayout object's methods are assumed to be relative to the TextLayout object's origin. Clients usually need to translate between a TextLayout object's coordinate system and the coordinate system in another object (such as a [Graphics](http://docs.google.com/java/awt/Graphics.html) object).

TextLayout objects are constructed from styled text, but they do not retain a reference to their source text. Thus, changes in the text previously used to generate a TextLayout do not affect the TextLayout.

Three methods on a TextLayout object (getNextRightHit, getNextLeftHit, and hitTestChar) return instances of [TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html). The offsets contained in these TextHitInfo objects are relative to the start of the TextLayout, **not** to the text used to create the TextLayout. Similarly, TextLayout methods that accept TextHitInfo instances as parameters expect the TextHitInfo object's offsets to be relative to the TextLayout, not to any underlying text storage model.

**Examples**:

Constructing and drawing a TextLayout and its bounding rectangle:

Graphics2D g = ...;  
 Point2D loc = ...;  
 Font font = Font.getFont("Helvetica-bold-italic");  
 FontRenderContext frc = g.getFontRenderContext();  
 TextLayout layout = new TextLayout("This is a string", font, frc);  
 layout.draw(g, (float)loc.getX(), (float)loc.getY());  
  
 Rectangle2D bounds = layout.getBounds();  
 bounds.setRect(bounds.getX()+loc.getX(),  
 bounds.getY()+loc.getY(),  
 bounds.getWidth(),  
 bounds.getHeight());  
 g.draw(bounds);

Hit-testing a TextLayout (determining which character is at a particular graphical location):

Point2D click = ...;  
 TextHitInfo hit = layout.hitTestChar(  
 (float) (click.getX() - loc.getX()),  
 (float) (click.getY() - loc.getY()));

Responding to a right-arrow key press:

int insertionIndex = ...;  
 TextHitInfo next = layout.getNextRightHit(insertionIndex);  
 if (next != null) {  
 // translate graphics to origin of layout on screen  
 g.translate(loc.getX(), loc.getY());  
 Shape[] carets = layout.getCaretShapes(next.getInsertionIndex());  
 g.draw(carets[0]);  
 if (carets[1] != null) {  
 g.draw(carets[1]);  
 }  
 }

Drawing a selection range corresponding to a substring in the source text. The selected area may not be visually contiguous:

// selStart, selLimit should be relative to the layout,  
 // not to the source text  
  
 int selStart = ..., selLimit = ...;  
 Color selectionColor = ...;  
 Shape selection = layout.getLogicalHighlightShape(selStart, selLimit);  
 // selection may consist of disjoint areas  
 // graphics is assumed to be tranlated to origin of layout  
 g.setColor(selectionColor);  
 g.fill(selection);

Drawing a visually contiguous selection range. The selection range may correspond to more than one substring in the source text. The ranges of the corresponding source text substrings can be obtained with getLogicalRangesForVisualSelection():

TextHitInfo selStart = ..., selLimit = ...;  
 Shape selection = layout.getVisualHighlightShape(selStart, selLimit);  
 g.setColor(selectionColor);  
 g.fill(selection);  
 int[] ranges = getLogicalRangesForVisualSelection(selStart, selLimit);  
 // ranges[0], ranges[1] is the first selection range,  
 // ranges[2], ranges[3] is the second selection range, etc.

Note: Font rotations can cause text baselines to be rotated, and multiple runs with different rotations can cause the baseline to bend or zig-zag. In order to account for this (rare) possibility, some APIs are specified to return metrics and take parameters 'in baseline-relative coordinates' (e.g. ascent, advance), and others are in 'in standard coordinates' (e.g. getBounds). Values in baseline-relative coordinates map the 'x' coordinate to the distance along the baseline, (positive x is forward along the baseline), and the 'y' coordinate to a distance along the perpendicular to the baseline at 'x' (postitive y is 90 degrees clockwise from the baseline vector). Values in standard coordinates are measured along the x and y axes, with 0,0 at the origin of the TextLayout. Documentation for each relevant API indicates what values are in what coordinate system. In general, measurement-related APIs are in baseline-relative coordinates, while display-related APIs are in standard coordinates.

**See Also:**[LineBreakMeasurer](http://docs.google.com/java/awt/font/LineBreakMeasurer.html), [TextAttribute](http://docs.google.com/java/awt/font/TextAttribute.html), [TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html), [LayoutPath](http://docs.google.com/java/awt/font/LayoutPath.html)

| **Nested Class Summary** | |
| --- | --- |
| static class | [**TextLayout.CaretPolicy**](http://docs.google.com/java/awt/font/TextLayout.CaretPolicy.html)            Defines a policy for determining the strong caret location. |

| **Field Summary** | |
| --- | --- |
| static [TextLayout.CaretPolicy](http://docs.google.com/java/awt/font/TextLayout.CaretPolicy.html) | [**DEFAULT\_CARET\_POLICY**](http://docs.google.com/java/awt/font/TextLayout.html#DEFAULT_CARET_POLICY)            This CaretPolicy is used when a policy is not specified by the client. |

| **Constructor Summary** | |
| --- | --- |
| [**TextLayout**](http://docs.google.com/java/awt/font/TextLayout.html#TextLayout(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 TextLayout from an iterator over styled text. |
| [**TextLayout**](http://docs.google.com/java/awt/font/TextLayout.html#TextLayout(java.lang.String,%20java.awt.Font,%20java.awt.font.FontRenderContext))([String](http://docs.google.com/java/lang/String.html) string, [Font](http://docs.google.com/java/awt/Font.html) font, [FontRenderContext](http://docs.google.com/java/awt/font/FontRenderContext.html) frc)            Constructs a TextLayout from a String and a [Font](http://docs.google.com/java/awt/Font.html). |
| [**TextLayout**](http://docs.google.com/java/awt/font/TextLayout.html#TextLayout(java.lang.String,%20java.util.Map,%20java.awt.font.FontRenderContext))([String](http://docs.google.com/java/lang/String.html) string, [Map](http://docs.google.com/java/util/Map.html)<? extends [AttributedCharacterIterator.Attribute](http://docs.google.com/java/text/AttributedCharacterIterator.Attribute.html),?> attributes, [FontRenderContext](http://docs.google.com/java/awt/font/FontRenderContext.html) frc)            Constructs a TextLayout from a String and an attribute set. |

| **Method Summary** | |
| --- | --- |
| protected  [Object](http://docs.google.com/java/lang/Object.html) | [**clone**](http://docs.google.com/java/awt/font/TextLayout.html#clone())()            Creates a copy of this TextLayout. |
| void | [**draw**](http://docs.google.com/java/awt/font/TextLayout.html#draw(java.awt.Graphics2D,%20float,%20float))([Graphics2D](http://docs.google.com/java/awt/Graphics2D.html) g2, float x, float y)            Renders this TextLayout at the specified location in the specified [Graphics2D](http://docs.google.com/java/awt/Graphics2D.html) context. |
| boolean | [**equals**](http://docs.google.com/java/awt/font/TextLayout.html#equals(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) obj)            Returns true if the specified Object is a TextLayout object and if the specified Object equals this TextLayout. |
| boolean | [**equals**](http://docs.google.com/java/awt/font/TextLayout.html#equals(java.awt.font.TextLayout))([TextLayout](http://docs.google.com/java/awt/font/TextLayout.html) rhs)            Returns true if the two layouts are equal. |
| float | [**getAdvance**](http://docs.google.com/java/awt/font/TextLayout.html#getAdvance())()            Returns the advance of this TextLayout. |
| float | [**getAscent**](http://docs.google.com/java/awt/font/TextLayout.html#getAscent())()            Returns the ascent of this TextLayout. |
| byte | [**getBaseline**](http://docs.google.com/java/awt/font/TextLayout.html#getBaseline())()            Returns the baseline for this TextLayout. |
| float[] | [**getBaselineOffsets**](http://docs.google.com/java/awt/font/TextLayout.html#getBaselineOffsets())()            Returns the offsets array for the baselines used for this TextLayout. |
| [Shape](http://docs.google.com/java/awt/Shape.html) | [**getBlackBoxBounds**](http://docs.google.com/java/awt/font/TextLayout.html#getBlackBoxBounds(int,%20int))(int firstEndpoint, int secondEndpoint)            Returns the black box bounds of the characters in the specified range. |
| [Rectangle2D](http://docs.google.com/java/awt/geom/Rectangle2D.html) | [**getBounds**](http://docs.google.com/java/awt/font/TextLayout.html#getBounds())()            Returns the bounds of this TextLayout. |
| float[] | [**getCaretInfo**](http://docs.google.com/java/awt/font/TextLayout.html#getCaretInfo(java.awt.font.TextHitInfo))([TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html) hit)            Returns information about the caret corresponding to hit. |
| float[] | [**getCaretInfo**](http://docs.google.com/java/awt/font/TextLayout.html#getCaretInfo(java.awt.font.TextHitInfo,%20java.awt.geom.Rectangle2D))([TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html) hit, [Rectangle2D](http://docs.google.com/java/awt/geom/Rectangle2D.html) bounds)            Returns information about the caret corresponding to hit. |
| [Shape](http://docs.google.com/java/awt/Shape.html) | [**getCaretShape**](http://docs.google.com/java/awt/font/TextLayout.html#getCaretShape(java.awt.font.TextHitInfo))([TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html) hit)            Returns a Shape representing the caret at the specified hit inside the natural bounds of this TextLayout. |
| [Shape](http://docs.google.com/java/awt/Shape.html) | [**getCaretShape**](http://docs.google.com/java/awt/font/TextLayout.html#getCaretShape(java.awt.font.TextHitInfo,%20java.awt.geom.Rectangle2D))([TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html) hit, [Rectangle2D](http://docs.google.com/java/awt/geom/Rectangle2D.html) bounds)            Returns a [Shape](http://docs.google.com/java/awt/Shape.html) representing the caret at the specified hit inside the specified bounds. |
| [Shape](http://docs.google.com/java/awt/Shape.html)[] | [**getCaretShapes**](http://docs.google.com/java/awt/font/TextLayout.html#getCaretShapes(int))(int offset)            Returns two paths corresponding to the strong and weak caret. |
| [Shape](http://docs.google.com/java/awt/Shape.html)[] | [**getCaretShapes**](http://docs.google.com/java/awt/font/TextLayout.html#getCaretShapes(int,%20java.awt.geom.Rectangle2D))(int offset, [Rectangle2D](http://docs.google.com/java/awt/geom/Rectangle2D.html) bounds)            Returns two paths corresponding to the strong and weak caret. |
| [Shape](http://docs.google.com/java/awt/Shape.html)[] | [**getCaretShapes**](http://docs.google.com/java/awt/font/TextLayout.html#getCaretShapes(int,%20java.awt.geom.Rectangle2D,%20java.awt.font.TextLayout.CaretPolicy))(int offset, [Rectangle2D](http://docs.google.com/java/awt/geom/Rectangle2D.html) bounds, [TextLayout.CaretPolicy](http://docs.google.com/java/awt/font/TextLayout.CaretPolicy.html) policy)            Returns two paths corresponding to the strong and weak caret. |
| int | [**getCharacterCount**](http://docs.google.com/java/awt/font/TextLayout.html#getCharacterCount())()            Returns the number of characters represented by this TextLayout. |
| byte | [**getCharacterLevel**](http://docs.google.com/java/awt/font/TextLayout.html#getCharacterLevel(int))(int index)            Returns the level of the character at index. |
| float | [**getDescent**](http://docs.google.com/java/awt/font/TextLayout.html#getDescent())()            Returns the descent of this TextLayout. |
| [TextLayout](http://docs.google.com/java/awt/font/TextLayout.html) | [**getJustifiedLayout**](http://docs.google.com/java/awt/font/TextLayout.html#getJustifiedLayout(float))(float justificationWidth)            Creates a copy of this TextLayout justified to the specified width. |
| [LayoutPath](http://docs.google.com/java/awt/font/LayoutPath.html) | [**getLayoutPath**](http://docs.google.com/java/awt/font/TextLayout.html#getLayoutPath())()            Return the LayoutPath, or null if the layout path is the default path (x maps to advance, y maps to offset). |
| float | [**getLeading**](http://docs.google.com/java/awt/font/TextLayout.html#getLeading())()            Returns the leading of the TextLayout. |
| [Shape](http://docs.google.com/java/awt/Shape.html) | [**getLogicalHighlightShape**](http://docs.google.com/java/awt/font/TextLayout.html#getLogicalHighlightShape(int,%20int))(int firstEndpoint, int secondEndpoint)            Returns a Shape enclosing the logical selection in the specified range, extended to the natural bounds of this TextLayout. |
| [Shape](http://docs.google.com/java/awt/Shape.html) | [**getLogicalHighlightShape**](http://docs.google.com/java/awt/font/TextLayout.html#getLogicalHighlightShape(int,%20int,%20java.awt.geom.Rectangle2D))(int firstEndpoint, int secondEndpoint, [Rectangle2D](http://docs.google.com/java/awt/geom/Rectangle2D.html) bounds)            Returns a Shape enclosing the logical selection in the specified range, extended to the specified bounds. |
| int[] | [**getLogicalRangesForVisualSelection**](http://docs.google.com/java/awt/font/TextLayout.html#getLogicalRangesForVisualSelection(java.awt.font.TextHitInfo,%20java.awt.font.TextHitInfo))([TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html) firstEndpoint, [TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html) secondEndpoint)            Returns the logical ranges of text corresponding to a visual selection. |
| [TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html) | [**getNextLeftHit**](http://docs.google.com/java/awt/font/TextLayout.html#getNextLeftHit(int))(int offset)            Returns the hit for the next caret to the left (top); if no such hit, returns null. |
| [TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html) | [**getNextLeftHit**](http://docs.google.com/java/awt/font/TextLayout.html#getNextLeftHit(int,%20java.awt.font.TextLayout.CaretPolicy))(int offset, [TextLayout.CaretPolicy](http://docs.google.com/java/awt/font/TextLayout.CaretPolicy.html) policy)            Returns the hit for the next caret to the left (top); if no such hit, returns null. |
| [TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html) | [**getNextLeftHit**](http://docs.google.com/java/awt/font/TextLayout.html#getNextLeftHit(java.awt.font.TextHitInfo))([TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html) hit)            Returns the hit for the next caret to the left (top); if no such hit, returns null. |
| [TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html) | [**getNextRightHit**](http://docs.google.com/java/awt/font/TextLayout.html#getNextRightHit(int))(int offset)            Returns the hit for the next caret to the right (bottom); if no such hit, returns null. |
| [TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html) | [**getNextRightHit**](http://docs.google.com/java/awt/font/TextLayout.html#getNextRightHit(int,%20java.awt.font.TextLayout.CaretPolicy))(int offset, [TextLayout.CaretPolicy](http://docs.google.com/java/awt/font/TextLayout.CaretPolicy.html) policy)            Returns the hit for the next caret to the right (bottom); if no such hit, returns null. |
| [TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html) | [**getNextRightHit**](http://docs.google.com/java/awt/font/TextLayout.html#getNextRightHit(java.awt.font.TextHitInfo))([TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html) hit)            Returns the hit for the next caret to the right (bottom); if there is no such hit, returns null. |
| [Shape](http://docs.google.com/java/awt/Shape.html) | [**getOutline**](http://docs.google.com/java/awt/font/TextLayout.html#getOutline(java.awt.geom.AffineTransform))([AffineTransform](http://docs.google.com/java/awt/geom/AffineTransform.html) tx)            Returns a Shape representing the outline of this TextLayout. |
| [Rectangle](http://docs.google.com/java/awt/Rectangle.html) | [**getPixelBounds**](http://docs.google.com/java/awt/font/TextLayout.html#getPixelBounds(java.awt.font.FontRenderContext,%20float,%20float))([FontRenderContext](http://docs.google.com/java/awt/font/FontRenderContext.html) frc, float x, float y)            Returns the pixel bounds of this TextLayout when rendered in a graphics with the given FontRenderContext at the given location. |
| float | [**getVisibleAdvance**](http://docs.google.com/java/awt/font/TextLayout.html#getVisibleAdvance())()            Returns the advance of this TextLayout, minus trailing whitespace. |
| [Shape](http://docs.google.com/java/awt/Shape.html) | [**getVisualHighlightShape**](http://docs.google.com/java/awt/font/TextLayout.html#getVisualHighlightShape(java.awt.font.TextHitInfo,%20java.awt.font.TextHitInfo))([TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html) firstEndpoint, [TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html) secondEndpoint)            Returns a Shape enclosing the visual selection in the specified range, extended to the bounds. |
| [Shape](http://docs.google.com/java/awt/Shape.html) | [**getVisualHighlightShape**](http://docs.google.com/java/awt/font/TextLayout.html#getVisualHighlightShape(java.awt.font.TextHitInfo,%20java.awt.font.TextHitInfo,%20java.awt.geom.Rectangle2D))([TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html) firstEndpoint, [TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html) secondEndpoint, [Rectangle2D](http://docs.google.com/java/awt/geom/Rectangle2D.html) bounds)            Returns a path enclosing the visual selection in the specified range, extended to bounds. |
| [TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html) | [**getVisualOtherHit**](http://docs.google.com/java/awt/font/TextLayout.html#getVisualOtherHit(java.awt.font.TextHitInfo))([TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html) hit)            Returns the hit on the opposite side of the specified hit's caret. |
| protected  void | [**handleJustify**](http://docs.google.com/java/awt/font/TextLayout.html#handleJustify(float))(float justificationWidth)            Justify this layout. |
| int | [**hashCode**](http://docs.google.com/java/awt/font/TextLayout.html#hashCode())()            Returns the hash code of this TextLayout. |
| [TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html) | [**hitTestChar**](http://docs.google.com/java/awt/font/TextLayout.html#hitTestChar(float,%20float))(float x, float y)            Returns a TextHitInfo corresponding to the specified point. |
| [TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html) | [**hitTestChar**](http://docs.google.com/java/awt/font/TextLayout.html#hitTestChar(float,%20float,%20java.awt.geom.Rectangle2D))(float x, float y, [Rectangle2D](http://docs.google.com/java/awt/geom/Rectangle2D.html) bounds)            Returns a TextHitInfo corresponding to the specified point. |
| void | [**hitToPoint**](http://docs.google.com/java/awt/font/TextLayout.html#hitToPoint(java.awt.font.TextHitInfo,%20java.awt.geom.Point2D))([TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html) hit, [Point2D](http://docs.google.com/java/awt/geom/Point2D.html) point)            Convert a hit to a point in standard coordinates. |
| boolean | [**isLeftToRight**](http://docs.google.com/java/awt/font/TextLayout.html#isLeftToRight())()            Returns true if this TextLayout has a left-to-right base direction or false if it has a right-to-left base direction. |
| boolean | [**isVertical**](http://docs.google.com/java/awt/font/TextLayout.html#isVertical())()            Returns true if this TextLayout is vertical. |
| [String](http://docs.google.com/java/lang/String.html) | [**toString**](http://docs.google.com/java/awt/font/TextLayout.html#toString())()            Returns debugging information for this TextLayout. |

| **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)) |

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

### DEFAULT\_CARET\_POLICY

public static final [TextLayout.CaretPolicy](http://docs.google.com/java/awt/font/TextLayout.CaretPolicy.html) **DEFAULT\_CARET\_POLICY**

This CaretPolicy is used when a policy is not specified by the client. With this policy, a hit on a character whose direction is the same as the line direction is stronger than a hit on a counterdirectional character. If the characters' directions are the same, a hit on the leading edge of a character is stronger than a hit on the trailing edge of a character.

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

### TextLayout

public **TextLayout**([String](http://docs.google.com/java/lang/String.html) string,  
 [Font](http://docs.google.com/java/awt/Font.html) font,  
 [FontRenderContext](http://docs.google.com/java/awt/font/FontRenderContext.html) frc)

Constructs a TextLayout from a String and a [Font](http://docs.google.com/java/awt/Font.html). All the text is styled using the specified Font.

The String must specify a single paragraph of text, because an entire paragraph is required for the bidirectional algorithm.

**Parameters:**string - the text to displayfont - a Font used to style the textfrc - 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 TextLayout and user space.

### TextLayout

public **TextLayout**([String](http://docs.google.com/java/lang/String.html) string,  
 [Map](http://docs.google.com/java/util/Map.html)<? extends [AttributedCharacterIterator.Attribute](http://docs.google.com/java/text/AttributedCharacterIterator.Attribute.html),?> attributes,  
 [FontRenderContext](http://docs.google.com/java/awt/font/FontRenderContext.html) frc)

Constructs a TextLayout from a String and an attribute set.

All the text is styled using the provided attributes.

string must specify a single paragraph of text because an entire paragraph is required for the bidirectional algorithm.

**Parameters:**string - the text to displayattributes - the attributes used to style the textfrc - 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 TextLayout and user space.

### TextLayout

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

Constructs a TextLayout from an iterator over styled text.

The iterator must specify a single paragraph of text because an entire paragraph is required for the bidirectional algorithm.

**Parameters:**text - the styled text to displayfrc - 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 TextLayout and user space.

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

### clone

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

Creates a copy of this TextLayout.

**Overrides:**[clone](http://docs.google.com/java/lang/Object.html#clone()) in class [Object](http://docs.google.com/java/lang/Object.html) **Returns:**a clone of this instance.**See Also:**[Cloneable](http://docs.google.com/java/lang/Cloneable.html)

### getJustifiedLayout

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

Creates a copy of this TextLayout justified to the specified width.

If this TextLayout has already been justified, an exception is thrown. If this TextLayout object's justification ratio is zero, a TextLayout identical to this TextLayout is returned.

**Parameters:**justificationWidth - the width to use when justifying the line. For best results, it should not be too different from the current advance of the line. **Returns:**a TextLayout justified to the specified width. **Throws:** [Error](http://docs.google.com/java/lang/Error.html) - if this layout has already been justified, an Error is thrown.

### handleJustify

protected void **handleJustify**(float justificationWidth)

Justify this layout. Overridden by subclassers to control justification (if there were subclassers, that is...) The layout will only justify if the paragraph attributes (from the source text, possibly defaulted by the layout attributes) indicate a non-zero justification ratio. The text will be justified to the indicated width. The current implementation also adjusts hanging punctuation and trailing whitespace to overhang the justification width. Once justified, the layout may not be rejustified.

Some code may rely on immutablity of layouts. Subclassers should not call this directly, but instead should call getJustifiedLayout, which will call this method on a clone of this layout, preserving the original.

**Parameters:**justificationWidth - the width to use when justifying the line. For best results, it should not be too different from the current advance of the line.**See Also:**[getJustifiedLayout(float)](http://docs.google.com/java/awt/font/TextLayout.html#getJustifiedLayout(float))

### getBaseline

public byte **getBaseline**()

Returns the baseline for this TextLayout. The baseline is one of the values defined in Font, which are roman, centered and hanging. Ascent and descent are relative to this baseline. The baselineOffsets are also relative to this baseline.

**Returns:**the baseline of this TextLayout.**See Also:**[getBaselineOffsets()](http://docs.google.com/java/awt/font/TextLayout.html#getBaselineOffsets()), [Font](http://docs.google.com/java/awt/Font.html)

### getBaselineOffsets

public float[] **getBaselineOffsets**()

Returns the offsets array for the baselines used for this TextLayout.

The array is indexed by one of the values defined in Font, which are roman, centered and hanging. The values are relative to this TextLayout object's baseline, so that getBaselineOffsets[getBaseline()] == 0. Offsets are added to the position of the TextLayout object's baseline to get the position for the new baseline.

**Returns:**the offsets array containing the baselines used for this TextLayout.**See Also:**[getBaseline()](http://docs.google.com/java/awt/font/TextLayout.html#getBaseline()), [Font](http://docs.google.com/java/awt/Font.html)

### getAdvance

public float **getAdvance**()

Returns the advance of this TextLayout. The advance is the distance from the origin to the advance of the rightmost (bottommost) character. This is in baseline-relative coordinates.

**Returns:**the advance of this TextLayout.

### getVisibleAdvance

public float **getVisibleAdvance**()

Returns the advance of this TextLayout, minus trailing whitespace. This is in baseline-relative coordinates.

**Returns:**the advance of this TextLayout without the trailing whitespace.**See Also:**[getAdvance()](http://docs.google.com/java/awt/font/TextLayout.html#getAdvance())

### getAscent

public float **getAscent**()

Returns the ascent of this TextLayout. The ascent is the distance from the top (right) of the TextLayout to the baseline. It is always either positive or zero. The ascent is sufficient to accomodate superscripted text and is the maximum of the sum of the ascent, offset, and baseline of each glyph. The ascent is the maximum ascent from the baseline of all the text in the TextLayout. It is in baseline-relative coordinates.

**Returns:**the ascent of this TextLayout.

### getDescent

public float **getDescent**()

Returns the descent of this TextLayout. The descent is the distance from the baseline to the bottom (left) of the TextLayout. It is always either positive or zero. The descent is sufficient to accomodate subscripted text and is the maximum of the sum of the descent, offset, and baseline of each glyph. This is the maximum descent from the baseline of all the text in the TextLayout. It is in baseline-relative coordinates.

**Returns:**the descent of this TextLayout.

### getLeading

public float **getLeading**()

Returns the leading of the TextLayout. The leading is the suggested interline spacing for this TextLayout. This is in baseline-relative coordinates.

The leading is computed from the leading, descent, and baseline of all glyphvectors in the TextLayout. The algorithm is roughly as follows:

maxD = 0;  
 maxDL = 0;  
 for (GlyphVector g in all glyphvectors) {  
 maxD = max(maxD, g.getDescent() + offsets[g.getBaseline()]);  
 maxDL = max(maxDL, g.getDescent() + g.getLeading() +  
 offsets[g.getBaseline()]);  
 }  
 return maxDL - maxD;

**Returns:**the leading of this TextLayout.

### getBounds

public [Rectangle2D](http://docs.google.com/java/awt/geom/Rectangle2D.html) **getBounds**()

Returns the bounds of this TextLayout. The bounds are in standard coordinates.

Due to rasterization effects, this bounds might not enclose all of the pixels rendered by the TextLayout.

It might not coincide exactly with the ascent, descent, origin or advance of the TextLayout.

**Returns:**a [Rectangle2D](http://docs.google.com/java/awt/geom/Rectangle2D.html) that is the bounds of this TextLayout.

### getPixelBounds

public [Rectangle](http://docs.google.com/java/awt/Rectangle.html) **getPixelBounds**([FontRenderContext](http://docs.google.com/java/awt/font/FontRenderContext.html) frc,  
 float x,  
 float y)

Returns the pixel bounds of this TextLayout when rendered in a graphics with the given FontRenderContext at the given location. The graphics render context need not be the same as the FontRenderContext used to create this TextLayout, and can be null. If it is null, the FontRenderContext of this TextLayout is used.

**Parameters:**frc - the FontRenderContext of the Graphics.x - the x-coordinate at which to render this TextLayout.y - the y-coordinate at which to render this TextLayout. **Returns:**a Rectangle bounding the pixels that would be affected.**Since:** 1.6 **See Also:**[GlyphVector.getPixelBounds(java.awt.font.FontRenderContext, float, float)](http://docs.google.com/java/awt/font/GlyphVector.html#getPixelBounds(java.awt.font.FontRenderContext,%20float,%20float))

### isLeftToRight

public boolean **isLeftToRight**()

Returns true if this TextLayout has a left-to-right base direction or false if it has a right-to-left base direction. The TextLayout has a base direction of either left-to-right (LTR) or right-to-left (RTL). The base direction is independent of the actual direction of text on the line, which may be either LTR, RTL, or mixed. Left-to-right layouts by default should position flush left. If the layout is on a tabbed line, the tabs run left to right, so that logically successive layouts position left to right. The opposite is true for RTL layouts. By default they should position flush left, and tabs run right-to-left.

**Returns:**true if the base direction of this TextLayout is left-to-right; false otherwise.

### isVertical

public boolean **isVertical**()

Returns true if this TextLayout is vertical.

**Returns:**true if this TextLayout is vertical; false otherwise.

### getCharacterCount

public int **getCharacterCount**()

Returns the number of characters represented by this TextLayout.

**Returns:**the number of characters in this TextLayout.

### getCaretInfo

public float[] **getCaretInfo**([TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html) hit,  
 [Rectangle2D](http://docs.google.com/java/awt/geom/Rectangle2D.html) bounds)

Returns information about the caret corresponding to hit. The first element of the array is the intersection of the caret with the baseline, as a distance along the baseline. The second element of the array is the inverse slope (run/rise) of the caret, measured with respect to the baseline at that point.

This method is meant for informational use. To display carets, it is better to use getCaretShapes.

**Parameters:**hit - a hit on a character in this TextLayoutbounds - the bounds to which the caret info is constructed. The bounds is in baseline-relative coordinates. **Returns:**a two-element array containing the position and slope of the caret. The returned caret info is in baseline-relative coordinates.**See Also:**[getCaretShapes(int, Rectangle2D, TextLayout.CaretPolicy)](http://docs.google.com/java/awt/font/TextLayout.html#getCaretShapes(int,%20java.awt.geom.Rectangle2D,%20java.awt.font.TextLayout.CaretPolicy)), [Font.getItalicAngle()](http://docs.google.com/java/awt/Font.html#getItalicAngle())

### getCaretInfo

public float[] **getCaretInfo**([TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html) hit)

Returns information about the caret corresponding to hit. This method is a convenience overload of getCaretInfo and uses the natural bounds of this TextLayout.

**Parameters:**hit - a hit on a character in this TextLayout **Returns:**the information about a caret corresponding to a hit. The returned caret info is in baseline-relative coordinates.

### getNextRightHit

public [TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html) **getNextRightHit**([TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html) hit)

Returns the hit for the next caret to the right (bottom); if there is no such hit, returns null. If the hit character index is out of bounds, an [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) is thrown.

**Parameters:**hit - a hit on a character in this layout **Returns:**a hit whose caret appears at the next position to the right (bottom) of the caret of the provided hit or null.

### getNextRightHit

public [TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html) **getNextRightHit**(int offset,  
 [TextLayout.CaretPolicy](http://docs.google.com/java/awt/font/TextLayout.CaretPolicy.html) policy)

Returns the hit for the next caret to the right (bottom); if no such hit, returns null. The hit is to the right of the strong caret at the specified offset, as determined by the specified policy. The returned hit is the stronger of the two possible hits, as determined by the specified policy.

**Parameters:**offset - an insertion offset in this TextLayout. Cannot be less than 0 or greater than this TextLayout object's character count.policy - the policy used to select the strong caret **Returns:**a hit whose caret appears at the next position to the right (bottom) of the caret of the provided hit, or null.

### getNextRightHit

public [TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html) **getNextRightHit**(int offset)

Returns the hit for the next caret to the right (bottom); if no such hit, returns null. The hit is to the right of the strong caret at the specified offset, as determined by the default policy. The returned hit is the stronger of the two possible hits, as determined by the default policy.

**Parameters:**offset - an insertion offset in this TextLayout. Cannot be less than 0 or greater than the TextLayout object's character count. **Returns:**a hit whose caret appears at the next position to the right (bottom) of the caret of the provided hit, or null.

### getNextLeftHit

public [TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html) **getNextLeftHit**([TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html) hit)

Returns the hit for the next caret to the left (top); if no such hit, returns null. If the hit character index is out of bounds, an IllegalArgumentException is thrown.

**Parameters:**hit - a hit on a character in this TextLayout. **Returns:**a hit whose caret appears at the next position to the left (top) of the caret of the provided hit, or null.

### getNextLeftHit

public [TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html) **getNextLeftHit**(int offset,  
 [TextLayout.CaretPolicy](http://docs.google.com/java/awt/font/TextLayout.CaretPolicy.html) policy)

Returns the hit for the next caret to the left (top); if no such hit, returns null. The hit is to the left of the strong caret at the specified offset, as determined by the specified policy. The returned hit is the stronger of the two possible hits, as determined by the specified policy.

**Parameters:**offset - an insertion offset in this TextLayout. Cannot be less than 0 or greater than this TextLayout object's character count.policy - the policy used to select the strong caret **Returns:**a hit whose caret appears at the next position to the left (top) of the caret of the provided hit, or null.

### getNextLeftHit

public [TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html) **getNextLeftHit**(int offset)

Returns the hit for the next caret to the left (top); if no such hit, returns null. The hit is to the left of the strong caret at the specified offset, as determined by the default policy. The returned hit is the stronger of the two possible hits, as determined by the default policy.

**Parameters:**offset - an insertion offset in this TextLayout. Cannot be less than 0 or greater than this TextLayout object's character count. **Returns:**a hit whose caret appears at the next position to the left (top) of the caret of the provided hit, or null.

### getVisualOtherHit

public [TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html) **getVisualOtherHit**([TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html) hit)

Returns the hit on the opposite side of the specified hit's caret.

**Parameters:**hit - the specified hit **Returns:**a hit that is on the opposite side of the specified hit's caret.

### getCaretShape

public [Shape](http://docs.google.com/java/awt/Shape.html) **getCaretShape**([TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html) hit,  
 [Rectangle2D](http://docs.google.com/java/awt/geom/Rectangle2D.html) bounds)

Returns a [Shape](http://docs.google.com/java/awt/Shape.html) representing the caret at the specified hit inside the specified bounds.

**Parameters:**hit - the hit at which to generate the caretbounds - the bounds of the TextLayout to use in generating the caret. The bounds is in baseline-relative coordinates. **Returns:**a Shape representing the caret. The returned shape is in standard coordinates.

### getCaretShape

public [Shape](http://docs.google.com/java/awt/Shape.html) **getCaretShape**([TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html) hit)

Returns a Shape representing the caret at the specified hit inside the natural bounds of this TextLayout.

**Parameters:**hit - the hit at which to generate the caret **Returns:**a Shape representing the caret. The returned shape is in standard coordinates.

### getCharacterLevel

public byte **getCharacterLevel**(int index)

Returns the level of the character at index. Indices -1 and characterCount are assigned the base level of this TextLayout.

**Parameters:**index - the index of the character from which to get the level **Returns:**the level of the character at the specified index.

### getCaretShapes

public [Shape](http://docs.google.com/java/awt/Shape.html)[] **getCaretShapes**(int offset,  
 [Rectangle2D](http://docs.google.com/java/awt/geom/Rectangle2D.html) bounds,  
 [TextLayout.CaretPolicy](http://docs.google.com/java/awt/font/TextLayout.CaretPolicy.html) policy)

Returns two paths corresponding to the strong and weak caret.

**Parameters:**offset - an offset in this TextLayoutbounds - the bounds to which to extend the carets. The bounds is in baseline-relative coordinates.policy - the specified CaretPolicy **Returns:**an array of two paths. Element zero is the strong caret. If there are two carets, element one is the weak caret, otherwise it is null. The returned shapes are in standard coordinates.

### getCaretShapes

public [Shape](http://docs.google.com/java/awt/Shape.html)[] **getCaretShapes**(int offset,  
 [Rectangle2D](http://docs.google.com/java/awt/geom/Rectangle2D.html) bounds)

Returns two paths corresponding to the strong and weak caret. This method is a convenience overload of getCaretShapes that uses the default caret policy.

**Parameters:**offset - an offset in this TextLayoutbounds - the bounds to which to extend the carets. This is in baseline-relative coordinates. **Returns:**two paths corresponding to the strong and weak caret as defined by the DEFAULT\_CARET\_POLICY. These are in standard coordinates.

### getCaretShapes

public [Shape](http://docs.google.com/java/awt/Shape.html)[] **getCaretShapes**(int offset)

Returns two paths corresponding to the strong and weak caret. This method is a convenience overload of getCaretShapes that uses the default caret policy and this TextLayout object's natural bounds.

**Parameters:**offset - an offset in this TextLayout **Returns:**two paths corresponding to the strong and weak caret as defined by the DEFAULT\_CARET\_POLICY. These are in standard coordinates.

### getLogicalRangesForVisualSelection

public int[] **getLogicalRangesForVisualSelection**([TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html) firstEndpoint,  
 [TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html) secondEndpoint)

Returns the logical ranges of text corresponding to a visual selection.

**Parameters:**firstEndpoint - an endpoint of the visual rangesecondEndpoint - the other endpoint of the visual range. This endpoint can be less than firstEndpoint. **Returns:**an array of integers representing start/limit pairs for the selected ranges.**See Also:**[getVisualHighlightShape(TextHitInfo, TextHitInfo, Rectangle2D)](http://docs.google.com/java/awt/font/TextLayout.html#getVisualHighlightShape(java.awt.font.TextHitInfo,%20java.awt.font.TextHitInfo,%20java.awt.geom.Rectangle2D))

### getVisualHighlightShape

public [Shape](http://docs.google.com/java/awt/Shape.html) **getVisualHighlightShape**([TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html) firstEndpoint,  
 [TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html) secondEndpoint,  
 [Rectangle2D](http://docs.google.com/java/awt/geom/Rectangle2D.html) bounds)

Returns a path enclosing the visual selection in the specified range, extended to bounds.

If the selection includes the leftmost (topmost) position, the selection is extended to the left (top) of bounds. If the selection includes the rightmost (bottommost) position, the selection is extended to the right (bottom) of the bounds. The height (width on vertical lines) of the selection is always extended to bounds.

Although the selection is always contiguous, the logically selected text can be discontiguous on lines with mixed-direction text. The logical ranges of text selected can be retrieved using getLogicalRangesForVisualSelection. For example, consider the text 'ABCdef' where capital letters indicate right-to-left text, rendered on a right-to-left line, with a visual selection from 0L (the leading edge of 'A') to 3T (the trailing edge of 'd'). The text appears as follows, with bold underlined areas representing the selection:

d**efCBA**

The logical selection ranges are 0-3, 4-6 (ABC, ef) because the visually contiguous text is logically discontiguous. Also note that since the rightmost position on the layout (to the right of 'A') is selected, the selection is extended to the right of the bounds.

**Parameters:**firstEndpoint - one end of the visual selectionsecondEndpoint - the other end of the visual selectionbounds - the bounding rectangle to which to extend the selection. This is in baseline-relative coordinates. **Returns:**a Shape enclosing the selection. This is in standard coordinates.**See Also:**[getLogicalRangesForVisualSelection(TextHitInfo, TextHitInfo)](http://docs.google.com/java/awt/font/TextLayout.html#getLogicalRangesForVisualSelection(java.awt.font.TextHitInfo,%20java.awt.font.TextHitInfo)), [getLogicalHighlightShape(int, int, Rectangle2D)](http://docs.google.com/java/awt/font/TextLayout.html#getLogicalHighlightShape(int,%20int,%20java.awt.geom.Rectangle2D))

### getVisualHighlightShape

public [Shape](http://docs.google.com/java/awt/Shape.html) **getVisualHighlightShape**([TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html) firstEndpoint,  
 [TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html) secondEndpoint)

Returns a Shape enclosing the visual selection in the specified range, extended to the bounds. This method is a convenience overload of getVisualHighlightShape that uses the natural bounds of this TextLayout.

**Parameters:**firstEndpoint - one end of the visual selectionsecondEndpoint - the other end of the visual selection **Returns:**a Shape enclosing the selection. This is in standard coordinates.

### getLogicalHighlightShape

public [Shape](http://docs.google.com/java/awt/Shape.html) **getLogicalHighlightShape**(int firstEndpoint,  
 int secondEndpoint,  
 [Rectangle2D](http://docs.google.com/java/awt/geom/Rectangle2D.html) bounds)

Returns a Shape enclosing the logical selection in the specified range, extended to the specified bounds.

If the selection range includes the first logical character, the selection is extended to the portion of bounds before the start of this TextLayout. If the range includes the last logical character, the selection is extended to the portion of bounds after the end of this TextLayout. The height (width on vertical lines) of the selection is always extended to bounds.

The selection can be discontiguous on lines with mixed-direction text. Only those characters in the logical range between start and limit appear selected. For example, consider the text 'ABCdef' where capital letters indicate right-to-left text, rendered on a right-to-left line, with a logical selection from 0 to 4 ('ABCd'). The text appears as follows, with bold standing in for the selection, and underlining for the extension:

**d**ef**CBA**

The selection is discontiguous because the selected characters are visually discontiguous. Also note that since the range includes the first logical character (A), the selection is extended to the portion of the bounds before the start of the layout, which in this case (a right-to-left line) is the right portion of the bounds.

**Parameters:**firstEndpoint - an endpoint in the range of characters to selectsecondEndpoint - the other endpoint of the range of characters to select. Can be less than firstEndpoint. The range includes the character at min(firstEndpoint, secondEndpoint), but excludes max(firstEndpoint, secondEndpoint).bounds - the bounding rectangle to which to extend the selection. This is in baseline-relative coordinates. **Returns:**an area enclosing the selection. This is in standard coordinates.**See Also:**[getVisualHighlightShape(TextHitInfo, TextHitInfo, Rectangle2D)](http://docs.google.com/java/awt/font/TextLayout.html#getVisualHighlightShape(java.awt.font.TextHitInfo,%20java.awt.font.TextHitInfo,%20java.awt.geom.Rectangle2D))

### getLogicalHighlightShape

public [Shape](http://docs.google.com/java/awt/Shape.html) **getLogicalHighlightShape**(int firstEndpoint,  
 int secondEndpoint)

Returns a Shape enclosing the logical selection in the specified range, extended to the natural bounds of this TextLayout. This method is a convenience overload of getLogicalHighlightShape that uses the natural bounds of this TextLayout.

**Parameters:**firstEndpoint - an endpoint in the range of characters to selectsecondEndpoint - the other endpoint of the range of characters to select. Can be less than firstEndpoint. The range includes the character at min(firstEndpoint, secondEndpoint), but excludes max(firstEndpoint, secondEndpoint). **Returns:**a Shape enclosing the selection. This is in standard coordinates.

### getBlackBoxBounds

public [Shape](http://docs.google.com/java/awt/Shape.html) **getBlackBoxBounds**(int firstEndpoint,  
 int secondEndpoint)

Returns the black box bounds of the characters in the specified range. The black box bounds is an area consisting of the union of the bounding boxes of all the glyphs corresponding to the characters between start and limit. This area can be disjoint.

**Parameters:**firstEndpoint - one end of the character rangesecondEndpoint - the other end of the character range. Can be less than firstEndpoint. **Returns:**a Shape enclosing the black box bounds. This is in standard coordinates.

### hitTestChar

public [TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html) **hitTestChar**(float x,  
 float y,  
 [Rectangle2D](http://docs.google.com/java/awt/geom/Rectangle2D.html) bounds)

Returns a TextHitInfo corresponding to the specified point. Coordinates outside the bounds of the TextLayout map to hits on the leading edge of the first logical character, or the trailing edge of the last logical character, as appropriate, regardless of the position of that character in the line. Only the direction along the baseline is used to make this evaluation.

**Parameters:**x - the x offset from the origin of this TextLayout. This is in standard coordinates.y - the y offset from the origin of this TextLayout. This is in standard coordinates.bounds - the bounds of the TextLayout. This is in baseline-relative coordinates. **Returns:**a hit describing the character and edge (leading or trailing) under the specified point.

### hitTestChar

public [TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html) **hitTestChar**(float x,  
 float y)

Returns a TextHitInfo corresponding to the specified point. This method is a convenience overload of hitTestChar that uses the natural bounds of this TextLayout.

**Parameters:**x - the x offset from the origin of this TextLayout. This is in standard coordinates.y - the y offset from the origin of this TextLayout. This is in standard coordinates. **Returns:**a hit describing the character and edge (leading or trailing) under the specified point.

### hashCode

public int **hashCode**()

Returns the hash code of this TextLayout.

**Overrides:**[hashCode](http://docs.google.com/java/lang/Object.html#hashCode()) in class [Object](http://docs.google.com/java/lang/Object.html) **Returns:**the hash code of this TextLayout.**See Also:**[Object.equals(java.lang.Object)](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [Hashtable](http://docs.google.com/java/util/Hashtable.html)

### equals

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

Returns true if the specified Object is a TextLayout object and if the specified Object equals this TextLayout.

**Overrides:**[equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)) in class [Object](http://docs.google.com/java/lang/Object.html) **Parameters:**obj - an Object to test for equality **Returns:**true if the specified Object equals this TextLayout; false otherwise.**See Also:**[Object.hashCode()](http://docs.google.com/java/lang/Object.html#hashCode()), [Hashtable](http://docs.google.com/java/util/Hashtable.html)

### equals

public boolean **equals**([TextLayout](http://docs.google.com/java/awt/font/TextLayout.html) rhs)

Returns true if the two layouts are equal. Two layouts are equal if they contain equal glyphvectors in the same order.

**Parameters:**rhs - the TextLayout to compare to this TextLayout **Returns:**true if the specified TextLayout equals this TextLayout.

### toString

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

Returns debugging information for this TextLayout.

**Overrides:**[toString](http://docs.google.com/java/lang/Object.html#toString()) in class [Object](http://docs.google.com/java/lang/Object.html) **Returns:**the textLine of this TextLayout as a String.

### draw

public void **draw**([Graphics2D](http://docs.google.com/java/awt/Graphics2D.html) g2,  
 float x,  
 float y)

Renders this TextLayout at the specified location in the specified [Graphics2D](http://docs.google.com/java/awt/Graphics2D.html) context. The origin of the layout is placed at x, y. Rendering may touch any point within getBounds() of this position. This leaves the g2 unchanged. Text is rendered along the baseline path.

**Parameters:**g2 - the Graphics2D context into which to render the layoutx - the X coordinate of the origin of this TextLayouty - the Y coordinate of the origin of this TextLayout**See Also:**[getBounds()](http://docs.google.com/java/awt/font/TextLayout.html#getBounds())

### getOutline

public [Shape](http://docs.google.com/java/awt/Shape.html) **getOutline**([AffineTransform](http://docs.google.com/java/awt/geom/AffineTransform.html) tx)

Returns a Shape representing the outline of this TextLayout.

**Parameters:**tx - an optional [AffineTransform](http://docs.google.com/java/awt/geom/AffineTransform.html) to apply to the outline of this TextLayout. **Returns:**a Shape that is the outline of this TextLayout. This is in standard coordinates.

### getLayoutPath

public [LayoutPath](http://docs.google.com/java/awt/font/LayoutPath.html) **getLayoutPath**()

Return the LayoutPath, or null if the layout path is the default path (x maps to advance, y maps to offset).

**Returns:**the layout path**Since:** 1.6

### hitToPoint

public void **hitToPoint**([TextHitInfo](http://docs.google.com/java/awt/font/TextHitInfo.html) hit,  
 [Point2D](http://docs.google.com/java/awt/geom/Point2D.html) point)

Convert a hit to a point in standard coordinates. The point is on the baseline of the character at the leading or trailing edge of the character, as appropriate. If the path is broken at the side of the character represented by the hit, the point will be adjacent to the character.

**Parameters:**hit - the hit to check. This must be a valid hit on the TextLayout.point - the returned point. The point is in standard coordinates. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the hit is not valid for the TextLayout. [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if hit or point is null.**Since:** 1.6

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/TextLayout.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/TextHitInfo.html)   [**NEXT CLASS**](http://docs.google.com/java/awt/font/TextLayout.CaretPolicy.html) | [**FRAMES**](http://docs.google.com/index.html?java/awt/font/TextLayout.html)    [**NO FRAMES**](http://docs.google.com/TextLayout.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | [FIELD](#2et92p0) | [CONSTR](#tyjcwt) | [METHOD](#3dy6vkm) | DETAIL: [FIELD](#4d34og8) | [CONSTR](#17dp8vu) | [METHOD](#35nkun2) |

[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).