| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/BasicProgressBarUI.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/plaf/basic/BasicPopupMenuUI.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.ChangeHandler.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/plaf/basic/BasicProgressBarUI.html)    [**NO FRAMES**](http://docs.google.com/BasicProgressBarUI.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | [FIELD](#2et92p0) | [CONSTR](#tyjcwt) | [METHOD](#3dy6vkm) | DETAIL: [FIELD](#2s8eyo1) | [CONSTR](#lnxbz9) | [METHOD](#1ksv4uv) |

## **javax.swing.plaf.basic**

Class BasicProgressBarUI

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 [javax.swing.plaf.ComponentUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html)  
 [javax.swing.plaf.ProgressBarUI](http://docs.google.com/javax/swing/plaf/ProgressBarUI.html)  
 **javax.swing.plaf.basic.BasicProgressBarUI**

**Direct Known Subclasses:** [MetalProgressBarUI](http://docs.google.com/javax/swing/plaf/metal/MetalProgressBarUI.html)

public class **BasicProgressBarUI**extends [ProgressBarUI](http://docs.google.com/javax/swing/plaf/ProgressBarUI.html)

A Basic L&F implementation of ProgressBarUI.

| **Nested Class Summary** | |
| --- | --- |
| class | [**BasicProgressBarUI.ChangeHandler**](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.ChangeHandler.html)            This inner class is marked "public" due to a compiler bug. |

| **Field Summary** | |
| --- | --- |
| protected  [Rectangle](http://docs.google.com/java/awt/Rectangle.html) | [**boxRect**](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.html#boxRect)            Used to hold the location and size of the bouncing box (returned by getBox) to be painted. |
| protected  [ChangeListener](http://docs.google.com/javax/swing/event/ChangeListener.html) | [**changeListener**](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.html#changeListener) |
| protected  [JProgressBar](http://docs.google.com/javax/swing/JProgressBar.html) | [**progressBar**](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.html#progressBar) |

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

| **Method Summary** | |
| --- | --- |
| static [ComponentUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html) | [**createUI**](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.html#createUI(javax.swing.JComponent))([JComponent](http://docs.google.com/javax/swing/JComponent.html) x) |
| protected  int | [**getAmountFull**](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.html#getAmountFull(java.awt.Insets,%20int,%20int))([Insets](http://docs.google.com/java/awt/Insets.html) b, int width, int height)            This determines the amount of the progress bar that should be filled based on the percent done gathered from the model. |
| protected  int | [**getAnimationIndex**](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.html#getAnimationIndex())()            Gets the index of the current animation frame. |
| int | [**getBaseline**](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.html#getBaseline(javax.swing.JComponent,%20int,%20int))([JComponent](http://docs.google.com/javax/swing/JComponent.html) c, int width, int height)            Returns the baseline. |
| [Component.BaselineResizeBehavior](http://docs.google.com/java/awt/Component.BaselineResizeBehavior.html) | [**getBaselineResizeBehavior**](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.html#getBaselineResizeBehavior(javax.swing.JComponent))([JComponent](http://docs.google.com/javax/swing/JComponent.html) c)            Returns an enum indicating how the baseline of the component changes as the size changes. |
| protected  [Rectangle](http://docs.google.com/java/awt/Rectangle.html) | [**getBox**](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.html#getBox(java.awt.Rectangle))([Rectangle](http://docs.google.com/java/awt/Rectangle.html) r)            Stores the position and size of the bouncing box that would be painted for the current animation index in r and returns r. |
| protected  int | [**getBoxLength**](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.html#getBoxLength(int,%20int))(int availableLength, int otherDimension)            Returns the length of the "bouncing box" to be painted. |
| protected  int | [**getCellLength**](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.html#getCellLength())()            Returns the width (if HORIZONTAL) or height (if VERTICAL) of each of the indivdual cells/units to be rendered in the progress bar. |
| protected  int | [**getCellSpacing**](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.html#getCellSpacing())()            Returns the spacing between each of the cells/units in the progress bar. |
| protected  int | [**getFrameCount**](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.html#getFrameCount())()            Returns the number of frames for the complete animation loop used by an indeterminate JProgessBar. |
| [Dimension](http://docs.google.com/java/awt/Dimension.html) | [**getMaximumSize**](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.html#getMaximumSize(javax.swing.JComponent))([JComponent](http://docs.google.com/javax/swing/JComponent.html) c)            Returns the specified component's maximum size appropriate for the look and feel. |
| [Dimension](http://docs.google.com/java/awt/Dimension.html) | [**getMinimumSize**](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.html#getMinimumSize(javax.swing.JComponent))([JComponent](http://docs.google.com/javax/swing/JComponent.html) c)            The Minimum size for this component is 10. |
| protected  [Dimension](http://docs.google.com/java/awt/Dimension.html) | [**getPreferredInnerHorizontal**](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.html#getPreferredInnerHorizontal())() |
| protected  [Dimension](http://docs.google.com/java/awt/Dimension.html) | [**getPreferredInnerVertical**](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.html#getPreferredInnerVertical())() |
| [Dimension](http://docs.google.com/java/awt/Dimension.html) | [**getPreferredSize**](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.html#getPreferredSize(javax.swing.JComponent))([JComponent](http://docs.google.com/javax/swing/JComponent.html) c)            Returns the specified component's preferred size appropriate for the look and feel. |
| protected  [Color](http://docs.google.com/java/awt/Color.html) | [**getSelectionBackground**](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.html#getSelectionBackground())()            The "selectionBackground" is the color of the text when it is painted over an unfilled area of the progress bar. |
| protected  [Color](http://docs.google.com/java/awt/Color.html) | [**getSelectionForeground**](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.html#getSelectionForeground())()            The "selectionForeground" is the color of the text when it is painted over a filled area of the progress bar. |
| protected  [Point](http://docs.google.com/java/awt/Point.html) | [**getStringPlacement**](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.html#getStringPlacement(java.awt.Graphics,%20java.lang.String,%20int,%20int,%20int,%20int))([Graphics](http://docs.google.com/java/awt/Graphics.html) g, [String](http://docs.google.com/java/lang/String.html) progressString, int x, int y, int width, int height)            Designate the place where the progress string will be painted. |
| protected  void | [**incrementAnimationIndex**](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.html#incrementAnimationIndex())()            Sets the index of the current animation frame, to the next valid value, which results in the progress bar being repainted. |
| protected  void | [**installDefaults**](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.html#installDefaults())() |
| protected  void | [**installListeners**](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.html#installListeners())() |
| void | [**installUI**](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.html#installUI(javax.swing.JComponent))([JComponent](http://docs.google.com/javax/swing/JComponent.html) c)            Configures the specified component appropriate for the look and feel. |
| void | [**paint**](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.html#paint(java.awt.Graphics,%20javax.swing.JComponent))([Graphics](http://docs.google.com/java/awt/Graphics.html) g, [JComponent](http://docs.google.com/javax/swing/JComponent.html) c)            Delegates painting to one of two methods: paintDeterminate or paintIndeterminate. |
| protected  void | [**paintDeterminate**](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.html#paintDeterminate(java.awt.Graphics,%20javax.swing.JComponent))([Graphics](http://docs.google.com/java/awt/Graphics.html) g, [JComponent](http://docs.google.com/javax/swing/JComponent.html) c)            All purpose paint method that should do the right thing for almost all linear, determinate progress bars. |
| protected  void | [**paintIndeterminate**](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.html#paintIndeterminate(java.awt.Graphics,%20javax.swing.JComponent))([Graphics](http://docs.google.com/java/awt/Graphics.html) g, [JComponent](http://docs.google.com/javax/swing/JComponent.html) c)            All purpose paint method that should do the right thing for all linear bouncing-box progress bars. |
| protected  void | [**paintString**](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.html#paintString(java.awt.Graphics,%20int,%20int,%20int,%20int,%20int,%20java.awt.Insets))([Graphics](http://docs.google.com/java/awt/Graphics.html) g, int x, int y, int width, int height, int amountFull, [Insets](http://docs.google.com/java/awt/Insets.html) b) |
| protected  void | [**setAnimationIndex**](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.html#setAnimationIndex(int))(int newValue)            Sets the index of the current animation frame to the specified value and requests that the progress bar be repainted. |
| protected  void | [**setCellLength**](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.html#setCellLength(int))(int cellLen) |
| protected  void | [**setCellSpacing**](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.html#setCellSpacing(int))(int cellSpace) |
| protected  void | [**startAnimationTimer**](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.html#startAnimationTimer())()            Starts the animation thread, creating and initializing it if necessary. |
| protected  void | [**stopAnimationTimer**](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.html#stopAnimationTimer())()            Stops the animation thread. |
| protected  void | [**uninstallDefaults**](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.html#uninstallDefaults())() |
| protected  void | [**uninstallListeners**](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.html#uninstallListeners())()            Removes all listeners installed by this object. |
| void | [**uninstallUI**](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.html#uninstallUI(javax.swing.JComponent))([JComponent](http://docs.google.com/javax/swing/JComponent.html) c)            Reverses configuration which was done on the specified component during installUI. |

| **Methods inherited from class javax.swing.plaf.**[**ComponentUI**](http://docs.google.com/javax/swing/plaf/ComponentUI.html) |
| --- |
| [contains](http://docs.google.com/javax/swing/plaf/ComponentUI.html#contains(javax.swing.JComponent,%20int,%20int)), [getAccessibleChild](http://docs.google.com/javax/swing/plaf/ComponentUI.html#getAccessibleChild(javax.swing.JComponent,%20int)), [getAccessibleChildrenCount](http://docs.google.com/javax/swing/plaf/ComponentUI.html#getAccessibleChildrenCount(javax.swing.JComponent)), [update](http://docs.google.com/javax/swing/plaf/ComponentUI.html#update(java.awt.Graphics,%20javax.swing.JComponent)) |

| **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)) |

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

### progressBar

protected [JProgressBar](http://docs.google.com/javax/swing/JProgressBar.html) **progressBar**

### changeListener

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

### boxRect

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

Used to hold the location and size of the bouncing box (returned by getBox) to be painted.

**Since:** 1.5

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

### BasicProgressBarUI

public **BasicProgressBarUI**()

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

### createUI

public static [ComponentUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html) **createUI**([JComponent](http://docs.google.com/javax/swing/JComponent.html) x)

### installUI

public void **installUI**([JComponent](http://docs.google.com/javax/swing/JComponent.html) c)

**Description copied from class:** [**ComponentUI**](http://docs.google.com/javax/swing/plaf/ComponentUI.html#installUI(javax.swing.JComponent)) Configures the specified component appropriate for the look and feel. This method is invoked when the ComponentUI instance is being installed as the UI delegate on the specified component. This method should completely configure the component for the look and feel, including the following:

1. Install any default property values for color, fonts, borders, icons, opacity, etc. on the component. Whenever possible, property values initialized by the client program should *not* be overridden.
2. Install a LayoutManager on the component if necessary.
3. Create/add any required sub-components to the component.
4. Create/install event listeners on the component.
5. Create/install a PropertyChangeListener on the component in order to detect and respond to component property changes appropriately.
6. Install keyboard UI (mnemonics, traversal, etc.) on the component.
7. Initialize any appropriate instance data.

**Overrides:**[installUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html#installUI(javax.swing.JComponent)) in class [ComponentUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html) **Parameters:**c - the component where this UI delegate is being installed**See Also:**[ComponentUI.uninstallUI(javax.swing.JComponent)](http://docs.google.com/javax/swing/plaf/ComponentUI.html#uninstallUI(javax.swing.JComponent)), [JComponent.setUI(javax.swing.plaf.ComponentUI)](http://docs.google.com/javax/swing/JComponent.html#setUI(javax.swing.plaf.ComponentUI)), [JComponent.updateUI()](http://docs.google.com/javax/swing/JComponent.html#updateUI())

### uninstallUI

public void **uninstallUI**([JComponent](http://docs.google.com/javax/swing/JComponent.html) c)

**Description copied from class:** [**ComponentUI**](http://docs.google.com/javax/swing/plaf/ComponentUI.html#uninstallUI(javax.swing.JComponent)) Reverses configuration which was done on the specified component during installUI. This method is invoked when this UIComponent instance is being removed as the UI delegate for the specified component. This method should undo the configuration performed in installUI, being careful to leave the JComponent instance in a clean state (no extraneous listeners, look-and-feel-specific property objects, etc.). This should include the following:

1. Remove any UI-set borders from the component.
2. Remove any UI-set layout managers on the component.
3. Remove any UI-added sub-components from the component.
4. Remove any UI-added event/property listeners from the component.
5. Remove any UI-installed keyboard UI from the component.
6. Nullify any allocated instance data objects to allow for GC.

**Overrides:**[uninstallUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html#uninstallUI(javax.swing.JComponent)) in class [ComponentUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html) **Parameters:**c - the component from which this UI delegate is being removed; this argument is often ignored, but might be used if the UI object is stateless and shared by multiple components**See Also:**[ComponentUI.installUI(javax.swing.JComponent)](http://docs.google.com/javax/swing/plaf/ComponentUI.html#installUI(javax.swing.JComponent)), [JComponent.updateUI()](http://docs.google.com/javax/swing/JComponent.html#updateUI())

### installDefaults

protected void **installDefaults**()

### uninstallDefaults

protected void **uninstallDefaults**()

### installListeners

protected void **installListeners**()

### startAnimationTimer

protected void **startAnimationTimer**()

Starts the animation thread, creating and initializing it if necessary. This method is invoked when an indeterminate progress bar should start animating. Reasons for this may include:

* The progress bar is determinate and becomes displayable
* The progress bar is displayable and becomes determinate
* The progress bar is displayable and determinate and this UI is installed

If you implement your own animation thread, you must override this method.

**Since:** 1.4 **See Also:**[stopAnimationTimer()](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.html#stopAnimationTimer())

### stopAnimationTimer

protected void **stopAnimationTimer**()

Stops the animation thread. This method is invoked when the indeterminate animation should be stopped. Reasons for this may include:

* The progress bar changes to determinate
* The progress bar is no longer part of a displayable hierarchy
* This UI in uninstalled

If you implement your own animation thread, you must override this method.

**Since:** 1.4 **See Also:**[startAnimationTimer()](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.html#startAnimationTimer())

### uninstallListeners

protected void **uninstallListeners**()

Removes all listeners installed by this object.

### getBaseline

public int **getBaseline**([JComponent](http://docs.google.com/javax/swing/JComponent.html) c,  
 int width,  
 int height)

Returns the baseline.

**Overrides:**[getBaseline](http://docs.google.com/javax/swing/plaf/ComponentUI.html#getBaseline(javax.swing.JComponent,%20int,%20int)) in class [ComponentUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html) **Parameters:**c - JComponent baseline is being requested forwidth - the width to get the baseline forheight - the height to get the baseline for **Returns:**baseline or a value < 0 indicating there is no reasonable baseline **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if c is null [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if width or height is < 0**Since:** 1.6 **See Also:**[JComponent.getBaseline(int, int)](http://docs.google.com/javax/swing/JComponent.html#getBaseline(int,%20int))

### getBaselineResizeBehavior

public [Component.BaselineResizeBehavior](http://docs.google.com/java/awt/Component.BaselineResizeBehavior.html) **getBaselineResizeBehavior**([JComponent](http://docs.google.com/javax/swing/JComponent.html) c)

Returns an enum indicating how the baseline of the component changes as the size changes.

**Overrides:**[getBaselineResizeBehavior](http://docs.google.com/javax/swing/plaf/ComponentUI.html#getBaselineResizeBehavior(javax.swing.JComponent)) in class [ComponentUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html) **Parameters:**c - JComponent to return baseline resize behavior for **Returns:**an enum indicating how the baseline changes as the component size changes **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if c is null**Since:** 1.6 **See Also:**[JComponent.getBaseline(int, int)](http://docs.google.com/javax/swing/JComponent.html#getBaseline(int,%20int))

### getPreferredInnerHorizontal

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

### getPreferredInnerVertical

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

### getSelectionForeground

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

The "selectionForeground" is the color of the text when it is painted over a filled area of the progress bar.

### getSelectionBackground

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

The "selectionBackground" is the color of the text when it is painted over an unfilled area of the progress bar.

### getCellLength

protected int **getCellLength**()

Returns the width (if HORIZONTAL) or height (if VERTICAL) of each of the indivdual cells/units to be rendered in the progress bar. However, for text rendering simplification and aesthetic considerations, this function will return 1 when the progress string is being rendered.

**Returns:**the value representing the spacing between cells**See Also:**[setCellLength(int)](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.html#setCellLength(int)), [JProgressBar.isStringPainted()](http://docs.google.com/javax/swing/JProgressBar.html#isStringPainted())

### setCellLength

protected void **setCellLength**(int cellLen)

### getCellSpacing

protected int **getCellSpacing**()

Returns the spacing between each of the cells/units in the progress bar. However, for text rendering simplification and aesthetic considerations, this function will return 0 when the progress string is being rendered.

**Returns:**the value representing the spacing between cells**See Also:**[setCellSpacing(int)](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.html#setCellSpacing(int)), [JProgressBar.isStringPainted()](http://docs.google.com/javax/swing/JProgressBar.html#isStringPainted())

### setCellSpacing

protected void **setCellSpacing**(int cellSpace)

### getAmountFull

protected int **getAmountFull**([Insets](http://docs.google.com/java/awt/Insets.html) b,  
 int width,  
 int height)

This determines the amount of the progress bar that should be filled based on the percent done gathered from the model. This is a common operation so it was abstracted out. It assumes that your progress bar is linear. That is, if you are making a circular progress indicator, you will want to override this method.

### paint

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

Delegates painting to one of two methods: paintDeterminate or paintIndeterminate.

**Overrides:**[paint](http://docs.google.com/javax/swing/plaf/ComponentUI.html#paint(java.awt.Graphics,%20javax.swing.JComponent)) in class [ComponentUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html) **Parameters:**g - the Graphics context in which to paintc - the component being painted; this argument is often ignored, but might be used if the UI object is stateless and shared by multiple components**See Also:**[ComponentUI.update(java.awt.Graphics, javax.swing.JComponent)](http://docs.google.com/javax/swing/plaf/ComponentUI.html#update(java.awt.Graphics,%20javax.swing.JComponent))

### getBox

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

Stores the position and size of the bouncing box that would be painted for the current animation index in r and returns r. Subclasses that add to the painting performed in this class's implementation of paintIndeterminate -- to draw an outline around the bouncing box, for example -- can use this method to get the location of the bouncing box that was just painted. By overriding this method, you have complete control over the size and position of the bouncing box, without having to reimplement paintIndeterminate.

**Parameters:**r - the Rectangle instance to be modified; may be null **Returns:**null if no box should be drawn; otherwise, returns the passed-in rectangle (if non-null) or a new rectangle**Since:** 1.4 **See Also:**[setAnimationIndex(int)](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.html#setAnimationIndex(int))

### getBoxLength

protected int **getBoxLength**(int availableLength,  
 int otherDimension)

Returns the length of the "bouncing box" to be painted. This method is invoked by the default implementation of paintIndeterminate to get the width (if the progress bar is horizontal) or height (if vertical) of the box. For example:

boxRect.width = getBoxLength(componentInnards.width,  
 componentInnards.height);

**Parameters:**availableLength - the amount of space available for the bouncing box to move in; for a horizontal progress bar, for example, this should be the inside width of the progress bar (the component width minus borders)otherDimension - for a horizontal progress bar, this should be the inside height of the progress bar; this value might be used to constrain or determine the return value **Returns:**the size of the box dimension being determined; must be no larger than availableLength**Since:** 1.5 **See Also:**[SwingUtilities.calculateInnerArea(javax.swing.JComponent, java.awt.Rectangle)](http://docs.google.com/javax/swing/SwingUtilities.html#calculateInnerArea(javax.swing.JComponent,%20java.awt.Rectangle))

### paintIndeterminate

protected void **paintIndeterminate**([Graphics](http://docs.google.com/java/awt/Graphics.html) g,  
 [JComponent](http://docs.google.com/javax/swing/JComponent.html) c)

All purpose paint method that should do the right thing for all linear bouncing-box progress bars. Override this if you are making another kind of progress bar.

**Since:** 1.4 **See Also:**[paintDeterminate(java.awt.Graphics, javax.swing.JComponent)](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.html#paintDeterminate(java.awt.Graphics,%20javax.swing.JComponent))

### paintDeterminate

protected void **paintDeterminate**([Graphics](http://docs.google.com/java/awt/Graphics.html) g,  
 [JComponent](http://docs.google.com/javax/swing/JComponent.html) c)

All purpose paint method that should do the right thing for almost all linear, determinate progress bars. By setting a few values in the defaults table, things should work just fine to paint your progress bar. Naturally, override this if you are making a circular or semi-circular progress bar.

**Since:** 1.4 **See Also:**[paintIndeterminate(java.awt.Graphics, javax.swing.JComponent)](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.html#paintIndeterminate(java.awt.Graphics,%20javax.swing.JComponent))

### paintString

protected void **paintString**([Graphics](http://docs.google.com/java/awt/Graphics.html) g,  
 int x,  
 int y,  
 int width,  
 int height,  
 int amountFull,  
 [Insets](http://docs.google.com/java/awt/Insets.html) b)

### getStringPlacement

protected [Point](http://docs.google.com/java/awt/Point.html) **getStringPlacement**([Graphics](http://docs.google.com/java/awt/Graphics.html) g,  
 [String](http://docs.google.com/java/lang/String.html) progressString,  
 int x,  
 int y,  
 int width,  
 int height)

Designate the place where the progress string will be painted. This implementation places it at the center of the progress bar (in both x and y). Override this if you want to right, left, top, or bottom align the progress string or if you need to nudge it around for any reason.

### getPreferredSize

public [Dimension](http://docs.google.com/java/awt/Dimension.html) **getPreferredSize**([JComponent](http://docs.google.com/javax/swing/JComponent.html) c)

**Description copied from class:** [**ComponentUI**](http://docs.google.com/javax/swing/plaf/ComponentUI.html#getPreferredSize(javax.swing.JComponent)) Returns the specified component's preferred size appropriate for the look and feel. If null is returned, the preferred size will be calculated by the component's layout manager instead (this is the preferred approach for any component with a specific layout manager installed). The default implementation of this method returns null.

**Overrides:**[getPreferredSize](http://docs.google.com/javax/swing/plaf/ComponentUI.html#getPreferredSize(javax.swing.JComponent)) in class [ComponentUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html) **Parameters:**c - the component whose preferred size is being queried; this argument is often ignored, but might be used if the UI object is stateless and shared by multiple components**See Also:**[JComponent.getPreferredSize()](http://docs.google.com/javax/swing/JComponent.html#getPreferredSize()), [LayoutManager.preferredLayoutSize(java.awt.Container)](http://docs.google.com/java/awt/LayoutManager.html#preferredLayoutSize(java.awt.Container))

### getMinimumSize

public [Dimension](http://docs.google.com/java/awt/Dimension.html) **getMinimumSize**([JComponent](http://docs.google.com/javax/swing/JComponent.html) c)

The Minimum size for this component is 10. The rationale here is that there should be at least one pixel per 10 percent.

**Overrides:**[getMinimumSize](http://docs.google.com/javax/swing/plaf/ComponentUI.html#getMinimumSize(javax.swing.JComponent)) in class [ComponentUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html) **Parameters:**c - the component whose minimum size is being queried; this argument is often ignored, but might be used if the UI object is stateless and shared by multiple components **Returns:**a Dimension object or null**See Also:**[JComponent.getMinimumSize()](http://docs.google.com/javax/swing/JComponent.html#getMinimumSize()), [LayoutManager.minimumLayoutSize(java.awt.Container)](http://docs.google.com/java/awt/LayoutManager.html#minimumLayoutSize(java.awt.Container)), [ComponentUI.getPreferredSize(javax.swing.JComponent)](http://docs.google.com/javax/swing/plaf/ComponentUI.html#getPreferredSize(javax.swing.JComponent))

### getMaximumSize

public [Dimension](http://docs.google.com/java/awt/Dimension.html) **getMaximumSize**([JComponent](http://docs.google.com/javax/swing/JComponent.html) c)

**Description copied from class:** [**ComponentUI**](http://docs.google.com/javax/swing/plaf/ComponentUI.html#getMaximumSize(javax.swing.JComponent)) Returns the specified component's maximum size appropriate for the look and feel. If null is returned, the maximum size will be calculated by the component's layout manager instead (this is the preferred approach for any component with a specific layout manager installed). The default implementation of this method invokes getPreferredSize and returns that value.

**Overrides:**[getMaximumSize](http://docs.google.com/javax/swing/plaf/ComponentUI.html#getMaximumSize(javax.swing.JComponent)) in class [ComponentUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html) **Parameters:**c - the component whose maximum size is being queried; this argument is often ignored, but might be used if the UI object is stateless and shared by multiple components **Returns:**a Dimension object or null**See Also:**[JComponent.getMaximumSize()](http://docs.google.com/javax/swing/JComponent.html#getMaximumSize()), [LayoutManager2.maximumLayoutSize(java.awt.Container)](http://docs.google.com/java/awt/LayoutManager2.html#maximumLayoutSize(java.awt.Container))

### getAnimationIndex

protected int **getAnimationIndex**()

Gets the index of the current animation frame.

**Since:** 1.4

### getFrameCount

protected final int **getFrameCount**()

Returns the number of frames for the complete animation loop used by an indeterminate JProgessBar. The progress chunk will go from one end to the other and back during the entire loop. This visual behavior may be changed by subclasses in other Look and Feels.

**Returns:**the number of frames**Since:** 1.6

### setAnimationIndex

protected void **setAnimationIndex**(int newValue)

Sets the index of the current animation frame to the specified value and requests that the progress bar be repainted. Subclasses that don't use the default painting code might need to override this method to change the way that the repaint method is invoked.

**Parameters:**newValue - the new animation index; no checking is performed on its value**Since:** 1.4 **See Also:**[incrementAnimationIndex()](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.html#incrementAnimationIndex())

### incrementAnimationIndex

protected void **incrementAnimationIndex**()

Sets the index of the current animation frame, to the next valid value, which results in the progress bar being repainted. The next valid value is, by default, the current animation index plus one. If the new value would be too large, this method sets the index to 0. Subclasses might need to override this method to ensure that the index does not go over the number of frames needed for the particular progress bar instance. This method is invoked by the default animation thread every *X* milliseconds, where *X* is specified by the "ProgressBar.repaintInterval" UI default.

**Since:** 1.4 **See Also:**[setAnimationIndex(int)](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.html#setAnimationIndex(int))

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/BasicProgressBarUI.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/plaf/basic/BasicPopupMenuUI.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/plaf/basic/BasicProgressBarUI.ChangeHandler.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/plaf/basic/BasicProgressBarUI.html)    [**NO FRAMES**](http://docs.google.com/BasicProgressBarUI.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | [FIELD](#2et92p0) | [CONSTR](#tyjcwt) | [METHOD](#3dy6vkm) | DETAIL: [FIELD](#2s8eyo1) | [CONSTR](#lnxbz9) | [METHOD](#1ksv4uv) |

[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).