| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/BasicLookAndFeel.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/BasicListUI.PropertyChangeHandler.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/plaf/basic/BasicMenuBarUI.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/plaf/basic/BasicLookAndFeel.html)    [**NO FRAMES**](http://docs.google.com/BasicLookAndFeel.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#1t3h5sf) | [METHOD](#2s8eyo1) |

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

Class BasicLookAndFeel

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

**All Implemented Interfaces:** [Serializable](http://docs.google.com/java/io/Serializable.html) **Direct Known Subclasses:** [MetalLookAndFeel](http://docs.google.com/javax/swing/plaf/metal/MetalLookAndFeel.html), [SynthLookAndFeel](http://docs.google.com/javax/swing/plaf/synth/SynthLookAndFeel.html)

public abstract class **BasicLookAndFeel**extends [LookAndFeel](http://docs.google.com/javax/swing/LookAndFeel.html)implements [Serializable](http://docs.google.com/java/io/Serializable.html)

A base class to use in creating a look and feel for Swing.

Each of the ComponentUIs provided by BasicLookAndFeel derives its behavior from the defaults table. Unless otherwise noted each of the ComponentUI implementations in this package document the set of defaults they use. Unless otherwise noted the defaults are installed at the time installUI is invoked, and follow the recommendations outlined in LookAndFeel for installing defaults.

**Warning:** Serialized objects of this class will not be compatible with future Swing releases. The current serialization support is appropriate for short term storage or RMI between applications running the same version of Swing. As of 1.4, support for long term storage of all JavaBeansTM has been added to the java.beans package. Please see [XMLEncoder](http://docs.google.com/java/beans/XMLEncoder.html).

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

| **Method Summary** | |
| --- | --- |
| protected  [Action](http://docs.google.com/javax/swing/Action.html) | [**createAudioAction**](http://docs.google.com/javax/swing/plaf/basic/BasicLookAndFeel.html#createAudioAction(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) key)            Creates and returns an Action used to play a sound. |
| protected  [ActionMap](http://docs.google.com/javax/swing/ActionMap.html) | [**getAudioActionMap**](http://docs.google.com/javax/swing/plaf/basic/BasicLookAndFeel.html#getAudioActionMap())()            Returns an ActionMap containing the audio actions for this look and feel. |
| [UIDefaults](http://docs.google.com/javax/swing/UIDefaults.html) | [**getDefaults**](http://docs.google.com/javax/swing/plaf/basic/BasicLookAndFeel.html#getDefaults())()            Returns the look and feel defaults. |
| protected  void | [**initClassDefaults**](http://docs.google.com/javax/swing/plaf/basic/BasicLookAndFeel.html#initClassDefaults(javax.swing.UIDefaults))([UIDefaults](http://docs.google.com/javax/swing/UIDefaults.html) table)            Populates table with mappings from uiClassID to the fully qualified name of the ui class. |
| protected  void | [**initComponentDefaults**](http://docs.google.com/javax/swing/plaf/basic/BasicLookAndFeel.html#initComponentDefaults(javax.swing.UIDefaults))([UIDefaults](http://docs.google.com/javax/swing/UIDefaults.html) table)            Populates table with the defaults for the basic look and feel. |
| void | [**initialize**](http://docs.google.com/javax/swing/plaf/basic/BasicLookAndFeel.html#initialize())()            Initializes the look and feel. |
| protected  void | [**initSystemColorDefaults**](http://docs.google.com/javax/swing/plaf/basic/BasicLookAndFeel.html#initSystemColorDefaults(javax.swing.UIDefaults))([UIDefaults](http://docs.google.com/javax/swing/UIDefaults.html) table)            Populates table with system colors. |
| protected  void | [**loadSystemColors**](http://docs.google.com/javax/swing/plaf/basic/BasicLookAndFeel.html#loadSystemColors(javax.swing.UIDefaults,%20java.lang.String%5B%5D,%20boolean))([UIDefaults](http://docs.google.com/javax/swing/UIDefaults.html) table, [String](http://docs.google.com/java/lang/String.html)[] systemColors, boolean useNative)            Populates table with the name-color pairs in systemColors. |
| protected  void | [**playSound**](http://docs.google.com/javax/swing/plaf/basic/BasicLookAndFeel.html#playSound(javax.swing.Action))([Action](http://docs.google.com/javax/swing/Action.html) audioAction)            If necessary, invokes actionPerformed on audioAction to play a sound. |
| void | [**uninitialize**](http://docs.google.com/javax/swing/plaf/basic/BasicLookAndFeel.html#uninitialize())()            Uninitializes the look and feel. |

| **Methods inherited from class javax.swing.**[**LookAndFeel**](http://docs.google.com/javax/swing/LookAndFeel.html) |
| --- |
| [getDescription](http://docs.google.com/javax/swing/LookAndFeel.html#getDescription()), [getDesktopPropertyValue](http://docs.google.com/javax/swing/LookAndFeel.html#getDesktopPropertyValue(java.lang.String,%20java.lang.Object)), [getDisabledIcon](http://docs.google.com/javax/swing/LookAndFeel.html#getDisabledIcon(javax.swing.JComponent,%20javax.swing.Icon)), [getDisabledSelectedIcon](http://docs.google.com/javax/swing/LookAndFeel.html#getDisabledSelectedIcon(javax.swing.JComponent,%20javax.swing.Icon)), [getID](http://docs.google.com/javax/swing/LookAndFeel.html#getID()), [getLayoutStyle](http://docs.google.com/javax/swing/LookAndFeel.html#getLayoutStyle()), [getName](http://docs.google.com/javax/swing/LookAndFeel.html#getName()), [getSupportsWindowDecorations](http://docs.google.com/javax/swing/LookAndFeel.html#getSupportsWindowDecorations()), [installBorder](http://docs.google.com/javax/swing/LookAndFeel.html#installBorder(javax.swing.JComponent,%20java.lang.String)), [installColors](http://docs.google.com/javax/swing/LookAndFeel.html#installColors(javax.swing.JComponent,%20java.lang.String,%20java.lang.String)), [installColorsAndFont](http://docs.google.com/javax/swing/LookAndFeel.html#installColorsAndFont(javax.swing.JComponent,%20java.lang.String,%20java.lang.String,%20java.lang.String)), [installProperty](http://docs.google.com/javax/swing/LookAndFeel.html#installProperty(javax.swing.JComponent,%20java.lang.String,%20java.lang.Object)), [isNativeLookAndFeel](http://docs.google.com/javax/swing/LookAndFeel.html#isNativeLookAndFeel()), [isSupportedLookAndFeel](http://docs.google.com/javax/swing/LookAndFeel.html#isSupportedLookAndFeel()), [loadKeyBindings](http://docs.google.com/javax/swing/LookAndFeel.html#loadKeyBindings(javax.swing.InputMap,%20java.lang.Object%5B%5D)), [makeComponentInputMap](http://docs.google.com/javax/swing/LookAndFeel.html#makeComponentInputMap(javax.swing.JComponent,%20java.lang.Object%5B%5D)), [makeIcon](http://docs.google.com/javax/swing/LookAndFeel.html#makeIcon(java.lang.Class,%20java.lang.String)), [makeInputMap](http://docs.google.com/javax/swing/LookAndFeel.html#makeInputMap(java.lang.Object%5B%5D)), [makeKeyBindings](http://docs.google.com/javax/swing/LookAndFeel.html#makeKeyBindings(java.lang.Object%5B%5D)), [provideErrorFeedback](http://docs.google.com/javax/swing/LookAndFeel.html#provideErrorFeedback(java.awt.Component)), [toString](http://docs.google.com/javax/swing/LookAndFeel.html#toString()), [uninstallBorder](http://docs.google.com/javax/swing/LookAndFeel.html#uninstallBorder(javax.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()), [wait](http://docs.google.com/java/lang/Object.html#wait()), [wait](http://docs.google.com/java/lang/Object.html#wait(long)), [wait](http://docs.google.com/java/lang/Object.html#wait(long,%20int)) |

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

### BasicLookAndFeel

public **BasicLookAndFeel**()

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

### getDefaults

public [UIDefaults](http://docs.google.com/javax/swing/UIDefaults.html) **getDefaults**()

Returns the look and feel defaults. The returned UIDefaults is populated by invoking, in order, initClassDefaults, initSystemColorDefaults and initComponentDefaults.

While this method is public, it should only be invoked by the UIManager when the look and feel is set as the current look and feel and after initialize has been invoked.

**Overrides:**[getDefaults](http://docs.google.com/javax/swing/LookAndFeel.html#getDefaults()) in class [LookAndFeel](http://docs.google.com/javax/swing/LookAndFeel.html) **Returns:**the look and feel defaults**See Also:**[initClassDefaults(javax.swing.UIDefaults)](http://docs.google.com/javax/swing/plaf/basic/BasicLookAndFeel.html#initClassDefaults(javax.swing.UIDefaults)), [initSystemColorDefaults(javax.swing.UIDefaults)](http://docs.google.com/javax/swing/plaf/basic/BasicLookAndFeel.html#initSystemColorDefaults(javax.swing.UIDefaults)), [initComponentDefaults(javax.swing.UIDefaults)](http://docs.google.com/javax/swing/plaf/basic/BasicLookAndFeel.html#initComponentDefaults(javax.swing.UIDefaults))

### initialize

public void **initialize**()

Initializes the look and feel. While this method is public, it should only be invoked by the UIManager when a look and feel is installed as the current look and feel. This method is invoked before the UIManager invokes getDefaults. This method is intended to perform any initialization for the look and feel. Subclasses should do any one-time setup they need here, rather than in a static initializer, because look and feel class objects may be loaded just to discover that isSupportedLookAndFeel() returns false.

**Overrides:**[initialize](http://docs.google.com/javax/swing/LookAndFeel.html#initialize()) in class [LookAndFeel](http://docs.google.com/javax/swing/LookAndFeel.html) **See Also:**[LookAndFeel.uninitialize()](http://docs.google.com/javax/swing/LookAndFeel.html#uninitialize()), [UIManager.setLookAndFeel(javax.swing.LookAndFeel)](http://docs.google.com/javax/swing/UIManager.html#setLookAndFeel(javax.swing.LookAndFeel))

### uninitialize

public void **uninitialize**()

Uninitializes the look and feel. While this method is public, it should only be invoked by the UIManager when the look and feel is uninstalled. For example, UIManager.setLookAndFeel invokes this when the look and feel is changed.

Subclasses may choose to free up some resources here.

**Overrides:**[uninitialize](http://docs.google.com/javax/swing/LookAndFeel.html#uninitialize()) in class [LookAndFeel](http://docs.google.com/javax/swing/LookAndFeel.html) **See Also:**[LookAndFeel.initialize()](http://docs.google.com/javax/swing/LookAndFeel.html#initialize()), [UIManager.setLookAndFeel(javax.swing.LookAndFeel)](http://docs.google.com/javax/swing/UIManager.html#setLookAndFeel(javax.swing.LookAndFeel))

### initClassDefaults

protected void **initClassDefaults**([UIDefaults](http://docs.google.com/javax/swing/UIDefaults.html) table)

Populates table with mappings from uiClassID to the fully qualified name of the ui class. The value for a particular uiClassID is "javax.swing.plaf.basic.Basic + uiClassID". For example, the value for the uiClassID TreeUI is "javax.swing.plaf.basic.BasicTreeUI".

**Parameters:**table - the UIDefaults instance the entries are added to **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if table is null**See Also:**[LookAndFeel](http://docs.google.com/javax/swing/LookAndFeel.html), [getDefaults()](http://docs.google.com/javax/swing/plaf/basic/BasicLookAndFeel.html#getDefaults())

### initSystemColorDefaults

protected void **initSystemColorDefaults**([UIDefaults](http://docs.google.com/javax/swing/UIDefaults.html) table)

Populates table with system colors. This creates an array of name-color pairs and invokes loadSystemColors.

The name is a String that corresponds to the name of one of the static SystemColor fields in the SystemColor class. A name-color pair is created for every such SystemColor field.

The color corresponds to a hex String as understood by Color.decode. For example, one of the name-color pairs is "desktop"-"#005C5C". This corresponds to the SystemColor field desktop, with a color value of new Color(0x005C5C).

The following shows two of the name-color pairs:

String[] nameColorPairs = new String[] {  
 "desktop", "#005C5C",  
 "activeCaption", "#000080" };  
 loadSystemColors(table, nameColorPairs, isNativeLookAndFeel());

As previously stated, this invokes loadSystemColors with the supplied table and name-color pair array. The last argument to loadSystemColors indicates whether the value of the field in SystemColor should be used. This method passes the value of isNativeLookAndFeel() as the last argument to loadSystemColors.

**Parameters:**table - the UIDefaults object the values are added to **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if table is null**See Also:**[SystemColor](http://docs.google.com/java/awt/SystemColor.html), [getDefaults()](http://docs.google.com/javax/swing/plaf/basic/BasicLookAndFeel.html#getDefaults()), [loadSystemColors(javax.swing.UIDefaults, java.lang.String[], boolean)](http://docs.google.com/javax/swing/plaf/basic/BasicLookAndFeel.html#loadSystemColors(javax.swing.UIDefaults,%20java.lang.String%5B%5D,%20boolean))

### loadSystemColors

protected void **loadSystemColors**([UIDefaults](http://docs.google.com/javax/swing/UIDefaults.html) table,  
 [String](http://docs.google.com/java/lang/String.html)[] systemColors,  
 boolean useNative)

Populates table with the name-color pairs in systemColors. Refer to [initSystemColorDefaults(UIDefaults)](http://docs.google.com/javax/swing/plaf/basic/BasicLookAndFeel.html#initSystemColorDefaults(javax.swing.UIDefaults)) for details on the format of systemColors.

An entry is added to table for each of the name-color pairs in systemColors. The entry key is the name of the name-color pair.

The value of the entry corresponds to the color of the name-color pair. The value of the entry is calculated in one of two ways. With either approach the value is always a ColorUIResource.

If useNative is false, the color is created by using Color.decode to convert the String into a Color. If decode can not convert the String into a Color (NumberFormatException is thrown) then a ColorUIResource of black is used.

If useNative is true, the color is the value of the field in SystemColor with the same name as the name of the name-color pair. If the field is not valid, a ColorUIResource of black is used.

**Parameters:**table - the UIDefaults object the values are added tosystemColors - array of name-color pairs as described in [initSystemColorDefaults(UIDefaults)](http://docs.google.com/javax/swing/plaf/basic/BasicLookAndFeel.html#initSystemColorDefaults(javax.swing.UIDefaults))useNative - whether the color is obtained from SystemColor or Color.decode **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if systemColors is null; or systemColors is not empty, and table is null; or one of the names of the name-color pairs is null; or useNative is false and one of the colors of the name-color pairs is null [ArrayIndexOutOfBoundsException](http://docs.google.com/java/lang/ArrayIndexOutOfBoundsException.html) - if useNative is false and systemColors.length is odd**See Also:**[initSystemColorDefaults(javax.swing.UIDefaults)](http://docs.google.com/javax/swing/plaf/basic/BasicLookAndFeel.html#initSystemColorDefaults(javax.swing.UIDefaults)), [SystemColor](http://docs.google.com/java/awt/SystemColor.html), [Color.decode(String)](http://docs.google.com/java/awt/Color.html#decode(java.lang.String))

### initComponentDefaults

protected void **initComponentDefaults**([UIDefaults](http://docs.google.com/javax/swing/UIDefaults.html) table)

Populates table with the defaults for the basic look and feel.

**Parameters:**table - the UIDefaults to add the values to **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if table is null

### getAudioActionMap

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

Returns an ActionMap containing the audio actions for this look and feel.

The returned ActionMap contains Actions that embody the ability to render an auditory cue. These auditory cues map onto user and system activities that may be useful for an end user to know about (such as a dialog box appearing).

At the appropriate time, the ComponentUI is responsible for obtaining an Action out of the ActionMap and passing it to playSound.

This method first looks up the ActionMap from the defaults using the key "AuditoryCues.actionMap".

If the value is non-null, it is returned. If the value of the default "AuditoryCues.actionMap" is null and the value of the default "AuditoryCues.cueList" is non-null, an ActionMapUIResource is created and populated. Population is done by iterating over each of the elements of the "AuditoryCues.cueList" array, and invoking createAudioAction() to create an Action for each element. The resulting Action is placed in the ActionMapUIResource, using the array element as the key. For example, if the "AuditoryCues.cueList" array contains a single-element, "audioKey", the ActionMapUIResource is created, then populated by way of actionMap.put(cueList[0], createAudioAction(cueList[0])).

If the value of the default "AuditoryCues.actionMap" is null and the value of the default "AuditoryCues.cueList" is null, an empty ActionMapUIResource is created.

**Returns:**an ActionMap containing Actions responsible for playing auditory cues **Throws:** [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the value of the default "AuditoryCues.actionMap" is not an ActionMap, or the value of the default "AuditoryCues.cueList" is not an Object[]**Since:** 1.4 **See Also:**[createAudioAction(java.lang.Object)](http://docs.google.com/javax/swing/plaf/basic/BasicLookAndFeel.html#createAudioAction(java.lang.Object)), [playSound(Action)](http://docs.google.com/javax/swing/plaf/basic/BasicLookAndFeel.html#playSound(javax.swing.Action))

### createAudioAction

protected [Action](http://docs.google.com/javax/swing/Action.html) **createAudioAction**([Object](http://docs.google.com/java/lang/Object.html) key)

Creates and returns an Action used to play a sound.

If key is non-null, an Action is created using the value from the defaults with key key. The value identifies the sound resource to load when actionPerformed is invoked on the Action. The sound resource is loaded into a byte[] by way of getClass().getResourceAsStream().

**Parameters:**key - the key identifying the audio action **Returns:**an Action used to play the source, or null if key is null**Since:** 1.4 **See Also:**[playSound(Action)](http://docs.google.com/javax/swing/plaf/basic/BasicLookAndFeel.html#playSound(javax.swing.Action))

### playSound

protected void **playSound**([Action](http://docs.google.com/javax/swing/Action.html) audioAction)

If necessary, invokes actionPerformed on audioAction to play a sound. The actionPerformed method is invoked if the value of the "AuditoryCues.playList" default is a non-null Object[] containing a String entry equal to the name of the audioAction.

**Parameters:**audioAction - an Action that knows how to render the audio associated with the system or user activity that is occurring; a value of null, is ignored **Throws:** [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if audioAction is non-null and the value of the default "AuditoryCues.playList" is not an Object[]**Since:** 1.4

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/BasicLookAndFeel.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/BasicListUI.PropertyChangeHandler.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/plaf/basic/BasicMenuBarUI.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/plaf/basic/BasicLookAndFeel.html)    [**NO FRAMES**](http://docs.google.com/BasicLookAndFeel.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#1t3h5sf) | [METHOD](#2s8eyo1) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).