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

## **javax.swing**

Class UIManager

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

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

public class **UIManager**extends [Object](http://docs.google.com/java/lang/Object.html)implements [Serializable](http://docs.google.com/java/io/Serializable.html)

UIManager manages the current look and feel, the set of available look and feels, PropertyChangeListeners that are notified when the look and feel changes, look and feel defaults, and convenience methods for obtaining various default values.

### Specifying the look and feel

The look and feel can be specified in two distinct ways: by specifying the fully qualified name of the class for the look and feel, or by creating an instance of LookAndFeel and passing it to setLookAndFeel. The following example illustrates setting the look and feel to the system look and feel:

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());

The following example illustrates setting the look and feel based on class name:

UIManager.setLookAndFeel("javax.swing.plaf.metal.MetalLookAndFeel");

Once the look and feel has been changed it is imperative to invoke updateUI on all JComponents. The method [SwingUtilities.updateComponentTreeUI(java.awt.Component)](http://docs.google.com/javax/swing/SwingUtilities.html#updateComponentTreeUI(java.awt.Component)) makes it easy to apply updateUI to a containment hierarchy. Refer to it for details. The exact behavior of not invoking updateUI after changing the look and feel is unspecified. It is very possible to receive unexpected exceptions, painting problems, or worse.

### Default look and feel

The class used for the default look and feel is chosen in the following manner:

1. If the system property swing.defaultlaf is non-null, use its value as the default look and feel class name.
2. If the [Properties](http://docs.google.com/java/util/Properties.html) file swing.properties exists and contains the key swing.defaultlaf, use its value as the default look and feel class name. The location that is checked for swing.properties may vary depending upon the implementation of the Java platform. In Sun's implementation the location is ${java.home}/lib/swing.properties. Refer to the release notes of the implementation being used for further details.
3. Otherwise use the cross platform look and feel.

### Defaults

UIManager manages three sets of UIDefaults. In order, they are:

1. Developer defaults. With few exceptions Swing does not alter the developer defaults; these are intended to be modified and used by the developer.
2. Look and feel defaults. The look and feel defaults are supplied by the look and feel at the time it is installed as the current look and feel (setLookAndFeel() is invoked). The look and feel defaults can be obtained using the getLookAndFeelDefaults() method.
3. Sytem defaults. The system defaults are provided by Swing.

Invoking any of the various get methods results in checking each of the defaults, in order, returning the first non-null value. For example, invoking UIManager.getString("Table.foreground") results in first checking developer defaults. If the developer defaults contain a value for "Table.foreground" it is returned, otherwise the look and feel defaults are checked, followed by the system defaults.

It's important to note that getDefaults returns a custom instance of UIDefaults with this resolution logic built into it. For example, UIManager.getDefaults().getString("Table.foreground") is equivalent to UIManager.getString("Table.foreground"). Both resolve using the algorithm just described. In many places the documentation uses the word defaults to refer to the custom instance of UIDefaults with the resolution logic as previously described.

When the look and feel is changed, UIManager alters only the look and feel defaults; the developer and system defaults are not altered by the UIManager in any way.

The set of defaults a particular look and feel supports is defined and documented by that look and feel. In addition, each look and feel, or ComponentUI provided by a look and feel, may access the defaults at different times in their life cycle. Some look and feels may agressively look up defaults, so that changing a default may not have an effect after installing the look and feel. Other look and feels may lazily access defaults so that a change to the defaults may effect an existing look and feel. Finally, other look and feels might not configure themselves from the defaults table in any way. None-the-less it is usually the case that a look and feel expects certain defaults, so that in general a ComponentUI provided by one look and feel will not work with another look and feel.

**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).

| **Nested Class Summary** | |
| --- | --- |
| static class | [**UIManager.LookAndFeelInfo**](http://docs.google.com/javax/swing/UIManager.LookAndFeelInfo.html)            Provides a little information about an installed LookAndFeel for the sake of configuring a menu or for initial application set up. |

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

| **Method Summary** | |
| --- | --- |
| static void | [**addAuxiliaryLookAndFeel**](http://docs.google.com/javax/swing/UIManager.html#addAuxiliaryLookAndFeel(javax.swing.LookAndFeel))([LookAndFeel](http://docs.google.com/javax/swing/LookAndFeel.html) laf)            Adds a LookAndFeel to the list of auxiliary look and feels. |
| static void | [**addPropertyChangeListener**](http://docs.google.com/javax/swing/UIManager.html#addPropertyChangeListener(java.beans.PropertyChangeListener))([PropertyChangeListener](http://docs.google.com/java/beans/PropertyChangeListener.html) listener)            Adds a PropertyChangeListener to the listener list. |
| static [Object](http://docs.google.com/java/lang/Object.html) | [**get**](http://docs.google.com/javax/swing/UIManager.html#get(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) key)            Returns an object from the defaults. |
| static [Object](http://docs.google.com/java/lang/Object.html) | [**get**](http://docs.google.com/javax/swing/UIManager.html#get(java.lang.Object,%20java.util.Locale))([Object](http://docs.google.com/java/lang/Object.html) key, [Locale](http://docs.google.com/java/util/Locale.html) l)            Returns an object from the defaults that is appropriate for the given locale. |
| static [LookAndFeel](http://docs.google.com/javax/swing/LookAndFeel.html)[] | [**getAuxiliaryLookAndFeels**](http://docs.google.com/javax/swing/UIManager.html#getAuxiliaryLookAndFeels())()            Returns the list of auxiliary look and feels (can be null). |
| static boolean | [**getBoolean**](http://docs.google.com/javax/swing/UIManager.html#getBoolean(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) key)            Returns a boolean from the defaults which is associated with the key value. |
| static boolean | [**getBoolean**](http://docs.google.com/javax/swing/UIManager.html#getBoolean(java.lang.Object,%20java.util.Locale))([Object](http://docs.google.com/java/lang/Object.html) key, [Locale](http://docs.google.com/java/util/Locale.html) l)            Returns a boolean from the defaults which is associated with the key value and the given Locale. |
| static [Border](http://docs.google.com/javax/swing/border/Border.html) | [**getBorder**](http://docs.google.com/javax/swing/UIManager.html#getBorder(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) key)            Returns a border from the defaults. |
| static [Border](http://docs.google.com/javax/swing/border/Border.html) | [**getBorder**](http://docs.google.com/javax/swing/UIManager.html#getBorder(java.lang.Object,%20java.util.Locale))([Object](http://docs.google.com/java/lang/Object.html) key, [Locale](http://docs.google.com/java/util/Locale.html) l)            Returns a border from the defaults that is appropriate for the given locale. |
| static [Color](http://docs.google.com/java/awt/Color.html) | [**getColor**](http://docs.google.com/javax/swing/UIManager.html#getColor(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) key)            Returns a color from the defaults. |
| static [Color](http://docs.google.com/java/awt/Color.html) | [**getColor**](http://docs.google.com/javax/swing/UIManager.html#getColor(java.lang.Object,%20java.util.Locale))([Object](http://docs.google.com/java/lang/Object.html) key, [Locale](http://docs.google.com/java/util/Locale.html) l)            Returns a color from the defaults that is appropriate for the given locale. |
| static [String](http://docs.google.com/java/lang/String.html) | [**getCrossPlatformLookAndFeelClassName**](http://docs.google.com/javax/swing/UIManager.html#getCrossPlatformLookAndFeelClassName())()            Returns the name of the LookAndFeel class that implements the default cross platform look and feel -- the Java Look and Feel (JLF). |
| static [UIDefaults](http://docs.google.com/javax/swing/UIDefaults.html) | [**getDefaults**](http://docs.google.com/javax/swing/UIManager.html#getDefaults())()            Returns the defaults. |
| static [Dimension](http://docs.google.com/java/awt/Dimension.html) | [**getDimension**](http://docs.google.com/javax/swing/UIManager.html#getDimension(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) key)            Returns a dimension from the defaults. |
| static [Dimension](http://docs.google.com/java/awt/Dimension.html) | [**getDimension**](http://docs.google.com/javax/swing/UIManager.html#getDimension(java.lang.Object,%20java.util.Locale))([Object](http://docs.google.com/java/lang/Object.html) key, [Locale](http://docs.google.com/java/util/Locale.html) l)            Returns a dimension from the defaults that is appropriate for the given locale. |
| static [Font](http://docs.google.com/java/awt/Font.html) | [**getFont**](http://docs.google.com/javax/swing/UIManager.html#getFont(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) key)            Returns a font from the defaults. |
| static [Font](http://docs.google.com/java/awt/Font.html) | [**getFont**](http://docs.google.com/javax/swing/UIManager.html#getFont(java.lang.Object,%20java.util.Locale))([Object](http://docs.google.com/java/lang/Object.html) key, [Locale](http://docs.google.com/java/util/Locale.html) l)            Returns a font from the defaults that is appropriate for the given locale. |
| static [Icon](http://docs.google.com/javax/swing/Icon.html) | [**getIcon**](http://docs.google.com/javax/swing/UIManager.html#getIcon(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) key)            Returns an Icon from the defaults. |
| static [Icon](http://docs.google.com/javax/swing/Icon.html) | [**getIcon**](http://docs.google.com/javax/swing/UIManager.html#getIcon(java.lang.Object,%20java.util.Locale))([Object](http://docs.google.com/java/lang/Object.html) key, [Locale](http://docs.google.com/java/util/Locale.html) l)            Returns an Icon from the defaults that is appropriate for the given locale. |
| static [Insets](http://docs.google.com/java/awt/Insets.html) | [**getInsets**](http://docs.google.com/javax/swing/UIManager.html#getInsets(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) key)            Returns an Insets object from the defaults. |
| static [Insets](http://docs.google.com/java/awt/Insets.html) | [**getInsets**](http://docs.google.com/javax/swing/UIManager.html#getInsets(java.lang.Object,%20java.util.Locale))([Object](http://docs.google.com/java/lang/Object.html) key, [Locale](http://docs.google.com/java/util/Locale.html) l)            Returns an Insets object from the defaults that is appropriate for the given locale. |
| static [UIManager.LookAndFeelInfo](http://docs.google.com/javax/swing/UIManager.LookAndFeelInfo.html)[] | [**getInstalledLookAndFeels**](http://docs.google.com/javax/swing/UIManager.html#getInstalledLookAndFeels())()            Returns an array of LookAndFeelInfos representing the LookAndFeel implementations currently available. |
| static int | [**getInt**](http://docs.google.com/javax/swing/UIManager.html#getInt(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) key)            Returns an integer from the defaults. |
| static int | [**getInt**](http://docs.google.com/javax/swing/UIManager.html#getInt(java.lang.Object,%20java.util.Locale))([Object](http://docs.google.com/java/lang/Object.html) key, [Locale](http://docs.google.com/java/util/Locale.html) l)            Returns an integer from the defaults that is appropriate for the given locale. |
| static [LookAndFeel](http://docs.google.com/javax/swing/LookAndFeel.html) | [**getLookAndFeel**](http://docs.google.com/javax/swing/UIManager.html#getLookAndFeel())()            Returns the current look and feel or null. |
| static [UIDefaults](http://docs.google.com/javax/swing/UIDefaults.html) | [**getLookAndFeelDefaults**](http://docs.google.com/javax/swing/UIManager.html#getLookAndFeelDefaults())()            Returns the UIDefaults from the current look and feel, that were obtained at the time the look and feel was installed. |
| static [PropertyChangeListener](http://docs.google.com/java/beans/PropertyChangeListener.html)[] | [**getPropertyChangeListeners**](http://docs.google.com/javax/swing/UIManager.html#getPropertyChangeListeners())()            Returns an array of all the PropertyChangeListeners added to this UIManager with addPropertyChangeListener(). |
| static [String](http://docs.google.com/java/lang/String.html) | [**getString**](http://docs.google.com/javax/swing/UIManager.html#getString(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) key)            Returns a string from the defaults. |
| static [String](http://docs.google.com/java/lang/String.html) | [**getString**](http://docs.google.com/javax/swing/UIManager.html#getString(java.lang.Object,%20java.util.Locale))([Object](http://docs.google.com/java/lang/Object.html) key, [Locale](http://docs.google.com/java/util/Locale.html) l)            Returns a string from the defaults that is appropriate for the given locale. |
| static [String](http://docs.google.com/java/lang/String.html) | [**getSystemLookAndFeelClassName**](http://docs.google.com/javax/swing/UIManager.html#getSystemLookAndFeelClassName())()            Returns the name of the LookAndFeel class that implements the native system look and feel if there is one, otherwise the name of the default cross platform LookAndFeel class. |
| static [ComponentUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html) | [**getUI**](http://docs.google.com/javax/swing/UIManager.html#getUI(javax.swing.JComponent))([JComponent](http://docs.google.com/javax/swing/JComponent.html) target)            Returns the appropriate ComponentUI implementation for target. |
| static void | [**installLookAndFeel**](http://docs.google.com/javax/swing/UIManager.html#installLookAndFeel(java.lang.String,%20java.lang.String))([String](http://docs.google.com/java/lang/String.html) name, [String](http://docs.google.com/java/lang/String.html) className)            Adds the specified look and feel to the set of available look and feels. |
| static void | [**installLookAndFeel**](http://docs.google.com/javax/swing/UIManager.html#installLookAndFeel(javax.swing.UIManager.LookAndFeelInfo))([UIManager.LookAndFeelInfo](http://docs.google.com/javax/swing/UIManager.LookAndFeelInfo.html) info)            Adds the specified look and feel to the set of available look and feels. |
| static [Object](http://docs.google.com/java/lang/Object.html) | [**put**](http://docs.google.com/javax/swing/UIManager.html#put(java.lang.Object,%20java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) key, [Object](http://docs.google.com/java/lang/Object.html) value)            Stores an object in the developer defaults. |
| static boolean | [**removeAuxiliaryLookAndFeel**](http://docs.google.com/javax/swing/UIManager.html#removeAuxiliaryLookAndFeel(javax.swing.LookAndFeel))([LookAndFeel](http://docs.google.com/javax/swing/LookAndFeel.html) laf)            Removes a LookAndFeel from the list of auxiliary look and feels. |
| static void | [**removePropertyChangeListener**](http://docs.google.com/javax/swing/UIManager.html#removePropertyChangeListener(java.beans.PropertyChangeListener))([PropertyChangeListener](http://docs.google.com/java/beans/PropertyChangeListener.html) listener)            Removes a PropertyChangeListener from the listener list. |
| static void | [**setInstalledLookAndFeels**](http://docs.google.com/javax/swing/UIManager.html#setInstalledLookAndFeels(javax.swing.UIManager.LookAndFeelInfo%5B%5D))([UIManager.LookAndFeelInfo](http://docs.google.com/javax/swing/UIManager.LookAndFeelInfo.html)[] infos)            Sets the set of available look and feels. |
| static void | [**setLookAndFeel**](http://docs.google.com/javax/swing/UIManager.html#setLookAndFeel(javax.swing.LookAndFeel))([LookAndFeel](http://docs.google.com/javax/swing/LookAndFeel.html) newLookAndFeel)            Sets the current look and feel to newLookAndFeel. |
| static void | [**setLookAndFeel**](http://docs.google.com/javax/swing/UIManager.html#setLookAndFeel(java.lang.String))([String](http://docs.google.com/java/lang/String.html) className)            Loads the LookAndFeel specified by the given class name, using the current thread's context class loader, and passes it to setLookAndFeel(LookAndFeel). |

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

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

### UIManager

public **UIManager**()

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

### getInstalledLookAndFeels

public static [UIManager.LookAndFeelInfo](http://docs.google.com/javax/swing/UIManager.LookAndFeelInfo.html)[] **getInstalledLookAndFeels**()

Returns an array of LookAndFeelInfos representing the LookAndFeel implementations currently available. The LookAndFeelInfo objects can be used by an application to construct a menu of look and feel options for the user, or to determine which look and feel to set at startup time. To avoid the penalty of creating numerous LookAndFeel objects, LookAndFeelInfo maintains the class name of the LookAndFeel class, not the actual LookAndFeel instance.

The following example illustrates setting the current look and feel from an instance of LookAndFeelInfo:

UIManager.setLookAndFeel(info.getClassName());

**Returns:**an array of LookAndFeelInfo objects**See Also:**[setLookAndFeel(javax.swing.LookAndFeel)](http://docs.google.com/javax/swing/UIManager.html#setLookAndFeel(javax.swing.LookAndFeel))

### setInstalledLookAndFeels

public static void **setInstalledLookAndFeels**([UIManager.LookAndFeelInfo](http://docs.google.com/javax/swing/UIManager.LookAndFeelInfo.html)[] infos)  
 throws [SecurityException](http://docs.google.com/java/lang/SecurityException.html)

Sets the set of available look and feels. While this method does not check to ensure all of the LookAndFeelInfos are non-null, it is strongly recommended that only non-null values are supplied in the infos array.

**Parameters:**infos - set of LookAndFeelInfo objects specifying the available look and feels **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if infos is null [SecurityException](http://docs.google.com/java/lang/SecurityException.html)**See Also:**[getInstalledLookAndFeels()](http://docs.google.com/javax/swing/UIManager.html#getInstalledLookAndFeels())

### installLookAndFeel

public static void **installLookAndFeel**([UIManager.LookAndFeelInfo](http://docs.google.com/javax/swing/UIManager.LookAndFeelInfo.html) info)

Adds the specified look and feel to the set of available look and feels. While this method allows a null info, it is strongly recommended that a non-null value be used.

**Parameters:**info - a LookAndFeelInfo object that names the look and feel and identifies the class that implements it**See Also:**[setInstalledLookAndFeels(javax.swing.UIManager.LookAndFeelInfo[])](http://docs.google.com/javax/swing/UIManager.html#setInstalledLookAndFeels(javax.swing.UIManager.LookAndFeelInfo%5B%5D))

### installLookAndFeel

public static void **installLookAndFeel**([String](http://docs.google.com/java/lang/String.html) name,  
 [String](http://docs.google.com/java/lang/String.html) className)

Adds the specified look and feel to the set of available look and feels. While this method does not check the arguments in any way, it is strongly recommended that non-null values be supplied.

**Parameters:**name - descriptive name of the look and feelclassName - name of the class that implements the look and feel**See Also:**[setInstalledLookAndFeels(javax.swing.UIManager.LookAndFeelInfo[])](http://docs.google.com/javax/swing/UIManager.html#setInstalledLookAndFeels(javax.swing.UIManager.LookAndFeelInfo%5B%5D))

### getLookAndFeel

public static [LookAndFeel](http://docs.google.com/javax/swing/LookAndFeel.html) **getLookAndFeel**()

Returns the current look and feel or null.

**Returns:**current look and feel, or null**See Also:**[setLookAndFeel(javax.swing.LookAndFeel)](http://docs.google.com/javax/swing/UIManager.html#setLookAndFeel(javax.swing.LookAndFeel))

### setLookAndFeel

public static void **setLookAndFeel**([LookAndFeel](http://docs.google.com/javax/swing/LookAndFeel.html) newLookAndFeel)  
 throws [UnsupportedLookAndFeelException](http://docs.google.com/javax/swing/UnsupportedLookAndFeelException.html)

Sets the current look and feel to newLookAndFeel. If the current look and feel is non-null uninitialize is invoked on it. If newLookAndFeel is non-null, initialize is invoked on it followed by getDefaults. The defaults returned from newLookAndFeel.getDefaults() replace those of the defaults from the previous look and feel. If the newLookAndFeel is null, the look and feel defaults are set to null.

A value of null can be used to set the look and feel to null. As the LookAndFeel is required for most of Swing to function, setting the LookAndFeel to null is strongly discouraged.

This is a JavaBeans bound property.

**Parameters:**newLookAndFeel - LookAndFeel to install **Throws:** [UnsupportedLookAndFeelException](http://docs.google.com/javax/swing/UnsupportedLookAndFeelException.html) - if newLookAndFeel is non-null and newLookAndFeel.isSupportedLookAndFeel() returns false**See Also:**[getLookAndFeel()](http://docs.google.com/javax/swing/UIManager.html#getLookAndFeel())

### setLookAndFeel

public static void **setLookAndFeel**([String](http://docs.google.com/java/lang/String.html) className)  
 throws [ClassNotFoundException](http://docs.google.com/java/lang/ClassNotFoundException.html),  
 [InstantiationException](http://docs.google.com/java/lang/InstantiationException.html),  
 [IllegalAccessException](http://docs.google.com/java/lang/IllegalAccessException.html),  
 [UnsupportedLookAndFeelException](http://docs.google.com/javax/swing/UnsupportedLookAndFeelException.html)

Loads the LookAndFeel specified by the given class name, using the current thread's context class loader, and passes it to setLookAndFeel(LookAndFeel).

**Parameters:**className - a string specifying the name of the class that implements the look and feel **Throws:** [ClassNotFoundException](http://docs.google.com/java/lang/ClassNotFoundException.html) - if the LookAndFeel class could not be found [InstantiationException](http://docs.google.com/java/lang/InstantiationException.html) - if a new instance of the class couldn't be created [IllegalAccessException](http://docs.google.com/java/lang/IllegalAccessException.html) - if the class or initializer isn't accessible [UnsupportedLookAndFeelException](http://docs.google.com/javax/swing/UnsupportedLookAndFeelException.html) - if lnf.isSupportedLookAndFeel() is false [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if className does not identify a class that extends LookAndFeel

### getSystemLookAndFeelClassName

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

Returns the name of the LookAndFeel class that implements the native system look and feel if there is one, otherwise the name of the default cross platform LookAndFeel class. This value can be overriden by setting the swing.systemlaf system property.

**Returns:**the String of the LookAndFeel class**See Also:**[setLookAndFeel(javax.swing.LookAndFeel)](http://docs.google.com/javax/swing/UIManager.html#setLookAndFeel(javax.swing.LookAndFeel)), [getCrossPlatformLookAndFeelClassName()](http://docs.google.com/javax/swing/UIManager.html#getCrossPlatformLookAndFeelClassName())

### getCrossPlatformLookAndFeelClassName

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

Returns the name of the LookAndFeel class that implements the default cross platform look and feel -- the Java Look and Feel (JLF). This value can be overriden by setting the swing.crossplatformlaf system property.

**Returns:**a string with the JLF implementation-class**See Also:**[setLookAndFeel(javax.swing.LookAndFeel)](http://docs.google.com/javax/swing/UIManager.html#setLookAndFeel(javax.swing.LookAndFeel)), [getSystemLookAndFeelClassName()](http://docs.google.com/javax/swing/UIManager.html#getSystemLookAndFeelClassName())

### getDefaults

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

Returns the defaults. The returned defaults resolve using the logic specified in the class documentation.

**Returns:**a UIDefaults object containing the default values

### getFont

public static [Font](http://docs.google.com/java/awt/Font.html) **getFont**([Object](http://docs.google.com/java/lang/Object.html) key)

Returns a font from the defaults. If the value for key is not a Font, null is returned.

**Parameters:**key - an Object specifying the font **Returns:**the Font object **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if key is null

### getFont

public static [Font](http://docs.google.com/java/awt/Font.html) **getFont**([Object](http://docs.google.com/java/lang/Object.html) key,  
 [Locale](http://docs.google.com/java/util/Locale.html) l)

Returns a font from the defaults that is appropriate for the given locale. If the value for key is not a Font, null is returned.

**Parameters:**key - an Object specifying the fontl - the Locale for which the font is desired; refer to UIDefaults for details on how a null Locale is handled **Returns:**the Font object **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if key is null**Since:** 1.4

### getColor

public static [Color](http://docs.google.com/java/awt/Color.html) **getColor**([Object](http://docs.google.com/java/lang/Object.html) key)

Returns a color from the defaults. If the value for key is not a Color, null is returned.

**Parameters:**key - an Object specifying the color **Returns:**the Color object **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if key is null

### getColor

public static [Color](http://docs.google.com/java/awt/Color.html) **getColor**([Object](http://docs.google.com/java/lang/Object.html) key,  
 [Locale](http://docs.google.com/java/util/Locale.html) l)

Returns a color from the defaults that is appropriate for the given locale. If the value for key is not a Color, null is returned.

**Parameters:**key - an Object specifying the colorl - the Locale for which the color is desired; refer to UIDefaults for details on how a null Locale is handled **Returns:**the Color object **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if key is null**Since:** 1.4

### getIcon

public static [Icon](http://docs.google.com/javax/swing/Icon.html) **getIcon**([Object](http://docs.google.com/java/lang/Object.html) key)

Returns an Icon from the defaults. If the value for key is not an Icon, null is returned.

**Parameters:**key - an Object specifying the icon **Returns:**the Icon object **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if key is null

### getIcon

public static [Icon](http://docs.google.com/javax/swing/Icon.html) **getIcon**([Object](http://docs.google.com/java/lang/Object.html) key,  
 [Locale](http://docs.google.com/java/util/Locale.html) l)

Returns an Icon from the defaults that is appropriate for the given locale. If the value for key is not an Icon, null is returned.

**Parameters:**key - an Object specifying the iconl - the Locale for which the icon is desired; refer to UIDefaults for details on how a null Locale is handled **Returns:**the Icon object **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if key is null**Since:** 1.4

### getBorder

public static [Border](http://docs.google.com/javax/swing/border/Border.html) **getBorder**([Object](http://docs.google.com/java/lang/Object.html) key)

Returns a border from the defaults. If the value for key is not a Border, null is returned.

**Parameters:**key - an Object specifying the border **Returns:**the Border object **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if key is null

### getBorder

public static [Border](http://docs.google.com/javax/swing/border/Border.html) **getBorder**([Object](http://docs.google.com/java/lang/Object.html) key,  
 [Locale](http://docs.google.com/java/util/Locale.html) l)

Returns a border from the defaults that is appropriate for the given locale. If the value for key is not a Border, null is returned.

**Parameters:**key - an Object specifying the borderl - the Locale for which the border is desired; refer to UIDefaults for details on how a null Locale is handled **Returns:**the Border object **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if key is null**Since:** 1.4

### getString

public static [String](http://docs.google.com/java/lang/String.html) **getString**([Object](http://docs.google.com/java/lang/Object.html) key)

Returns a string from the defaults. If the value for key is not a String, null is returned.

**Parameters:**key - an Object specifying the string **Returns:**the String **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if key is null

### getString

public static [String](http://docs.google.com/java/lang/String.html) **getString**([Object](http://docs.google.com/java/lang/Object.html) key,  
 [Locale](http://docs.google.com/java/util/Locale.html) l)

Returns a string from the defaults that is appropriate for the given locale. If the value for key is not a String, null is returned.

**Parameters:**key - an Object specifying the stringl - the Locale for which the string is desired; refer to UIDefaults for details on how a null Locale is handled **Returns:**the String **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if key is null**Since:** 1.4

### getInt

public static int **getInt**([Object](http://docs.google.com/java/lang/Object.html) key)

Returns an integer from the defaults. If the value for key is not an Integer, or does not exist, 0 is returned.

**Parameters:**key - an Object specifying the int **Returns:**the int **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if key is null

### getInt

public static int **getInt**([Object](http://docs.google.com/java/lang/Object.html) key,  
 [Locale](http://docs.google.com/java/util/Locale.html) l)

Returns an integer from the defaults that is appropriate for the given locale. If the value for key is not an Integer, or does not exist, 0 is returned.

**Parameters:**key - an Object specifying the intl - the Locale for which the int is desired; refer to UIDefaults for details on how a null Locale is handled **Returns:**the int **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if key is null**Since:** 1.4

### getBoolean

public static boolean **getBoolean**([Object](http://docs.google.com/java/lang/Object.html) key)

Returns a boolean from the defaults which is associated with the key value. If the key is not found or the key doesn't represent a boolean value then false is returned.

**Parameters:**key - an Object specifying the key for the desired boolean value **Returns:**the boolean value corresponding to the key **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if key is null**Since:** 1.4

### getBoolean

public static boolean **getBoolean**([Object](http://docs.google.com/java/lang/Object.html) key,  
 [Locale](http://docs.google.com/java/util/Locale.html) l)

Returns a boolean from the defaults which is associated with the key value and the given Locale. If the key is not found or the key doesn't represent a boolean value then false will be returned.

**Parameters:**key - an Object specifying the key for the desired boolean valuel - the Locale for which the boolean is desired; refer to UIDefaults for details on how a null Locale is handled **Returns:**the boolean value corresponding to the key **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if key is null**Since:** 1.4

### getInsets

public static [Insets](http://docs.google.com/java/awt/Insets.html) **getInsets**([Object](http://docs.google.com/java/lang/Object.html) key)

Returns an Insets object from the defaults. If the value for key is not an Insets, null is returned.

**Parameters:**key - an Object specifying the Insets object **Returns:**the Insets object **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if key is null

### getInsets

public static [Insets](http://docs.google.com/java/awt/Insets.html) **getInsets**([Object](http://docs.google.com/java/lang/Object.html) key,  
 [Locale](http://docs.google.com/java/util/Locale.html) l)

Returns an Insets object from the defaults that is appropriate for the given locale. If the value for key is not an Insets, null is returned.

**Parameters:**key - an Object specifying the Insets objectl - the Locale for which the object is desired; refer to UIDefaults for details on how a null Locale is handled **Returns:**the Insets object **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if key is null**Since:** 1.4

### getDimension

public static [Dimension](http://docs.google.com/java/awt/Dimension.html) **getDimension**([Object](http://docs.google.com/java/lang/Object.html) key)

Returns a dimension from the defaults. If the value for key is not a Dimension, null is returned.

**Parameters:**key - an Object specifying the dimension object **Returns:**the Dimension object **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if key is null

### getDimension

public static [Dimension](http://docs.google.com/java/awt/Dimension.html) **getDimension**([Object](http://docs.google.com/java/lang/Object.html) key,  
 [Locale](http://docs.google.com/java/util/Locale.html) l)

Returns a dimension from the defaults that is appropriate for the given locale. If the value for key is not a Dimension, null is returned.

**Parameters:**key - an Object specifying the dimension objectl - the Locale for which the object is desired; refer to UIDefaults for details on how a null Locale is handled **Returns:**the Dimension object **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if key is null**Since:** 1.4

### get

public static [Object](http://docs.google.com/java/lang/Object.html) **get**([Object](http://docs.google.com/java/lang/Object.html) key)

Returns an object from the defaults.

**Parameters:**key - an Object specifying the desired object **Returns:**the Object **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if key is null

### get

public static [Object](http://docs.google.com/java/lang/Object.html) **get**([Object](http://docs.google.com/java/lang/Object.html) key,  
 [Locale](http://docs.google.com/java/util/Locale.html) l)

Returns an object from the defaults that is appropriate for the given locale.

**Parameters:**key - an Object specifying the desired objectl - the Locale for which the object is desired; refer to UIDefaults for details on how a null Locale is handled **Returns:**the Object **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if key is null**Since:** 1.4

### put

public static [Object](http://docs.google.com/java/lang/Object.html) **put**([Object](http://docs.google.com/java/lang/Object.html) key,  
 [Object](http://docs.google.com/java/lang/Object.html) value)

Stores an object in the developer defaults. This is a cover method for getDefaults().put(key, value). This only effects the developer defaults, not the system or look and feel defaults.

**Parameters:**key - an Object specifying the retrieval keyvalue - the Object to store; refer to UIDefaults for details on how null is handled **Returns:**the Object returned by [UIDefaults.put(java.lang.Object, java.lang.Object)](http://docs.google.com/javax/swing/UIDefaults.html#put(java.lang.Object,%20java.lang.Object)) **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if key is null**See Also:**[UIDefaults.put(java.lang.Object, java.lang.Object)](http://docs.google.com/javax/swing/UIDefaults.html#put(java.lang.Object,%20java.lang.Object))

### getUI

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

Returns the appropriate ComponentUI implementation for target. Typically, this is a cover for getDefaults().getUI(target). However, if an auxiliary look and feel has been installed, this first invokes getUI(target) on the multiplexing look and feel's defaults, and returns that value if it is non-null.

**Parameters:**target - the JComponent to return the ComponentUI for **Returns:**the ComponentUI object for target **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if target is null**See Also:**[UIDefaults.getUI(javax.swing.JComponent)](http://docs.google.com/javax/swing/UIDefaults.html#getUI(javax.swing.JComponent))

### getLookAndFeelDefaults

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

Returns the UIDefaults from the current look and feel, that were obtained at the time the look and feel was installed.

In general, developers should use the UIDefaults returned from getDefaults(). As the current look and feel may expect certain values to exist, altering the UIDefaults returned from this method could have unexpected results.

**Returns:**UIDefaults from the current look and feel**See Also:**[getDefaults()](http://docs.google.com/javax/swing/UIManager.html#getDefaults()), [setLookAndFeel(LookAndFeel)](http://docs.google.com/javax/swing/UIManager.html#setLookAndFeel(javax.swing.LookAndFeel)), [LookAndFeel.getDefaults()](http://docs.google.com/javax/swing/LookAndFeel.html#getDefaults())

### addAuxiliaryLookAndFeel

public static void **addAuxiliaryLookAndFeel**([LookAndFeel](http://docs.google.com/javax/swing/LookAndFeel.html) laf)

Adds a LookAndFeel to the list of auxiliary look and feels. The auxiliary look and feels tell the multiplexing look and feel what other LookAndFeel classes for a component instance are to be used in addition to the default LookAndFeel class when creating a multiplexing UI. The change will only take effect when a new UI class is created or when the default look and feel is changed on a component instance.

Note these are not the same as the installed look and feels.

**Parameters:**laf - the LookAndFeel object**See Also:**[removeAuxiliaryLookAndFeel(javax.swing.LookAndFeel)](http://docs.google.com/javax/swing/UIManager.html#removeAuxiliaryLookAndFeel(javax.swing.LookAndFeel)), [setLookAndFeel(javax.swing.LookAndFeel)](http://docs.google.com/javax/swing/UIManager.html#setLookAndFeel(javax.swing.LookAndFeel)), [getAuxiliaryLookAndFeels()](http://docs.google.com/javax/swing/UIManager.html#getAuxiliaryLookAndFeels()), [getInstalledLookAndFeels()](http://docs.google.com/javax/swing/UIManager.html#getInstalledLookAndFeels())

### removeAuxiliaryLookAndFeel

public static boolean **removeAuxiliaryLookAndFeel**([LookAndFeel](http://docs.google.com/javax/swing/LookAndFeel.html) laf)

Removes a LookAndFeel from the list of auxiliary look and feels. The auxiliary look and feels tell the multiplexing look and feel what other LookAndFeel classes for a component instance are to be used in addition to the default LookAndFeel class when creating a multiplexing UI. The change will only take effect when a new UI class is created or when the default look and feel is changed on a component instance.

Note these are not the same as the installed look and feels.

**Returns:**true if the LookAndFeel was removed from the list**See Also:**[removeAuxiliaryLookAndFeel(javax.swing.LookAndFeel)](http://docs.google.com/javax/swing/UIManager.html#removeAuxiliaryLookAndFeel(javax.swing.LookAndFeel)), [getAuxiliaryLookAndFeels()](http://docs.google.com/javax/swing/UIManager.html#getAuxiliaryLookAndFeels()), [setLookAndFeel(javax.swing.LookAndFeel)](http://docs.google.com/javax/swing/UIManager.html#setLookAndFeel(javax.swing.LookAndFeel)), [getInstalledLookAndFeels()](http://docs.google.com/javax/swing/UIManager.html#getInstalledLookAndFeels())

### getAuxiliaryLookAndFeels

public static [LookAndFeel](http://docs.google.com/javax/swing/LookAndFeel.html)[] **getAuxiliaryLookAndFeels**()

Returns the list of auxiliary look and feels (can be null). The auxiliary look and feels tell the multiplexing look and feel what other LookAndFeel classes for a component instance are to be used in addition to the default LookAndFeel class when creating a multiplexing UI.

Note these are not the same as the installed look and feels.

**Returns:**list of auxiliary LookAndFeels or null**See Also:**[addAuxiliaryLookAndFeel(javax.swing.LookAndFeel)](http://docs.google.com/javax/swing/UIManager.html#addAuxiliaryLookAndFeel(javax.swing.LookAndFeel)), [removeAuxiliaryLookAndFeel(javax.swing.LookAndFeel)](http://docs.google.com/javax/swing/UIManager.html#removeAuxiliaryLookAndFeel(javax.swing.LookAndFeel)), [setLookAndFeel(javax.swing.LookAndFeel)](http://docs.google.com/javax/swing/UIManager.html#setLookAndFeel(javax.swing.LookAndFeel)), [getInstalledLookAndFeels()](http://docs.google.com/javax/swing/UIManager.html#getInstalledLookAndFeels())

### addPropertyChangeListener

public static void **addPropertyChangeListener**([PropertyChangeListener](http://docs.google.com/java/beans/PropertyChangeListener.html) listener)

Adds a PropertyChangeListener to the listener list. The listener is registered for all properties.

**Parameters:**listener - the PropertyChangeListener to be added**See Also:**[PropertyChangeSupport](http://docs.google.com/java/beans/PropertyChangeSupport.html)

### removePropertyChangeListener

public static void **removePropertyChangeListener**([PropertyChangeListener](http://docs.google.com/java/beans/PropertyChangeListener.html) listener)

Removes a PropertyChangeListener from the listener list. This removes a PropertyChangeListener that was registered for all properties.

**Parameters:**listener - the PropertyChangeListener to be removed**See Also:**[PropertyChangeSupport](http://docs.google.com/java/beans/PropertyChangeSupport.html)

### getPropertyChangeListeners

public static [PropertyChangeListener](http://docs.google.com/java/beans/PropertyChangeListener.html)[] **getPropertyChangeListeners**()

Returns an array of all the PropertyChangeListeners added to this UIManager with addPropertyChangeListener().

**Returns:**all of the PropertyChangeListeners added or an empty array if no listeners have been added**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/UIManager.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/UIDefaults.ProxyLazyValue.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/UIManager.LookAndFeelInfo.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/UIManager.html)    [**NO FRAMES**](http://docs.google.com/UIManager.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | FIELD | [CONSTR](#2et92p0) | [METHOD](#tyjcwt) | 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).