| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Window.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/awt/TrayIcon.MessageType.html)   [**NEXT CLASS**](http://docs.google.com/java/awt/Window.AccessibleAWTWindow.html) | [**FRAMES**](http://docs.google.com/index.html?java/awt/Window.html)    [**NO FRAMES**](http://docs.google.com/Window.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | [FIELD](#1t3h5sf) | [CONSTR](#2s8eyo1) | [METHOD](#17dp8vu) | DETAIL: FIELD | [CONSTR](#35nkun2) | [METHOD](#z337ya) |

## **java.awt**

Class Window

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 [java.awt.Component](http://docs.google.com/java/awt/Component.html)  
 [java.awt.Container](http://docs.google.com/java/awt/Container.html)  
 **java.awt.Window**

**All Implemented Interfaces:** [ImageObserver](http://docs.google.com/java/awt/image/ImageObserver.html), [MenuContainer](http://docs.google.com/java/awt/MenuContainer.html), [Serializable](http://docs.google.com/java/io/Serializable.html), [Accessible](http://docs.google.com/javax/accessibility/Accessible.html) **Direct Known Subclasses:** [BasicToolBarUI.DragWindow](http://docs.google.com/javax/swing/plaf/basic/BasicToolBarUI.DragWindow.html), [Dialog](http://docs.google.com/java/awt/Dialog.html), [Frame](http://docs.google.com/java/awt/Frame.html), [JWindow](http://docs.google.com/javax/swing/JWindow.html)

public class **Window**extends [Container](http://docs.google.com/java/awt/Container.html)implements [Accessible](http://docs.google.com/javax/accessibility/Accessible.html)

A Window object is a top-level window with no borders and no menubar. The default layout for a window is BorderLayout.

A window must have either a frame, dialog, or another window defined as its owner when it's constructed.

In a multi-screen environment, you can create a Window on a different screen device by constructing the Window with [Window(Window, GraphicsConfiguration)](http://docs.google.com/java/awt/Window.html#Window(java.awt.Window,%20java.awt.GraphicsConfiguration)). The GraphicsConfiguration object is one of the GraphicsConfiguration objects of the target screen device.

In a virtual device multi-screen environment in which the desktop area could span multiple physical screen devices, the bounds of all configurations are relative to the virtual device coordinate system. The origin of the virtual-coordinate system is at the upper left-hand corner of the primary physical screen. Depending on the location of the primary screen in the virtual device, negative coordinates are possible, as shown in the following figure.



In such an environment, when calling setLocation, you must pass a virtual coordinate to this method. Similarly, calling getLocationOnScreen on a Window returns virtual device coordinates. Call the getBounds method of a GraphicsConfiguration to find its origin in the virtual coordinate system.

The following code sets the location of a Window at (10, 10) relative to the origin of the physical screen of the corresponding GraphicsConfiguration. If the bounds of the GraphicsConfiguration is not taken into account, the Window location would be set at (10, 10) relative to the virtual-coordinate system and would appear on the primary physical screen, which might be different from the physical screen of the specified GraphicsConfiguration.

Window w = new Window(Window owner, GraphicsConfiguration gc);  
 Rectangle bounds = gc.getBounds();  
 w.setLocation(10 + bounds.x, 10 + bounds.y);

Note: the location and size of top-level windows (including Windows, Frames, and Dialogs) are under the control of the desktop's window management system. Calls to setLocation, setSize, and setBounds are requests (not directives) which are forwarded to the window management system. Every effort will be made to honor such requests. However, in some cases the window management system may ignore such requests, or modify the requested geometry in order to place and size the Window in a way that more closely matches the desktop settings.

Due to the asynchronous nature of native event handling, the results returned by getBounds, getLocation, getLocationOnScreen, and getSize might not reflect the actual geometry of the Window on screen until the last request has been processed. During the processing of subsequent requests these values might change accordingly while the window management system fulfills the requests.

An application may set the size and location of an invisible Window arbitrarily, but the window management system may subsequently change its size and/or location when the Window is made visible. One or more ComponentEvents will be generated to indicate the new geometry.

Windows are capable of generating the following WindowEvents: WindowOpened, WindowClosed, WindowGainedFocus, WindowLostFocus.

**Since:** JDK1.0 **See Also:**[WindowEvent](http://docs.google.com/java/awt/event/WindowEvent.html), [addWindowListener(java.awt.event.WindowListener)](http://docs.google.com/java/awt/Window.html#addWindowListener(java.awt.event.WindowListener)), [BorderLayout](http://docs.google.com/java/awt/BorderLayout.html), [Serialized Form](http://docs.google.com/serialized-form.html#java.awt.Window)

| **Nested Class Summary** | |
| --- | --- |
| protected  class | [**Window.AccessibleAWTWindow**](http://docs.google.com/java/awt/Window.AccessibleAWTWindow.html)            This class implements accessibility support for the Window class. |

| **Nested classes/interfaces inherited from class java.awt.**[**Container**](http://docs.google.com/java/awt/Container.html) |
| --- |
| [Container.AccessibleAWTContainer](http://docs.google.com/java/awt/Container.AccessibleAWTContainer.html) |

| **Nested classes/interfaces inherited from class java.awt.**[**Component**](http://docs.google.com/java/awt/Component.html) |
| --- |
| [Component.AccessibleAWTComponent](http://docs.google.com/java/awt/Component.AccessibleAWTComponent.html), [Component.BaselineResizeBehavior](http://docs.google.com/java/awt/Component.BaselineResizeBehavior.html), [Component.BltBufferStrategy](http://docs.google.com/java/awt/Component.BltBufferStrategy.html), [Component.FlipBufferStrategy](http://docs.google.com/java/awt/Component.FlipBufferStrategy.html) |

| **Field Summary** | |
| --- | --- |

| **Fields inherited from class java.awt.**[**Component**](http://docs.google.com/java/awt/Component.html) |
| --- |
| [BOTTOM\_ALIGNMENT](http://docs.google.com/java/awt/Component.html#BOTTOM_ALIGNMENT), [CENTER\_ALIGNMENT](http://docs.google.com/java/awt/Component.html#CENTER_ALIGNMENT), [LEFT\_ALIGNMENT](http://docs.google.com/java/awt/Component.html#LEFT_ALIGNMENT), [RIGHT\_ALIGNMENT](http://docs.google.com/java/awt/Component.html#RIGHT_ALIGNMENT), [TOP\_ALIGNMENT](http://docs.google.com/java/awt/Component.html#TOP_ALIGNMENT) |

| **Fields inherited from interface java.awt.image.**[**ImageObserver**](http://docs.google.com/java/awt/image/ImageObserver.html) |
| --- |
| [ABORT](http://docs.google.com/java/awt/image/ImageObserver.html#ABORT), [ALLBITS](http://docs.google.com/java/awt/image/ImageObserver.html#ALLBITS), [ERROR](http://docs.google.com/java/awt/image/ImageObserver.html#ERROR), [FRAMEBITS](http://docs.google.com/java/awt/image/ImageObserver.html#FRAMEBITS), [HEIGHT](http://docs.google.com/java/awt/image/ImageObserver.html#HEIGHT), [PROPERTIES](http://docs.google.com/java/awt/image/ImageObserver.html#PROPERTIES), [SOMEBITS](http://docs.google.com/java/awt/image/ImageObserver.html#SOMEBITS), [WIDTH](http://docs.google.com/java/awt/image/ImageObserver.html#WIDTH) |

| **Constructor Summary** | |
| --- | --- |
| [**Window**](http://docs.google.com/java/awt/Window.html#Window(java.awt.Frame))([Frame](http://docs.google.com/java/awt/Frame.html) owner)            Constructs a new, initially invisible window with the specified Frame as its owner. |
| [**Window**](http://docs.google.com/java/awt/Window.html#Window(java.awt.Window))([Window](http://docs.google.com/java/awt/Window.html) owner)            Constructs a new, initially invisible window with the specified Window as its owner. |
| [**Window**](http://docs.google.com/java/awt/Window.html#Window(java.awt.Window,%20java.awt.GraphicsConfiguration))([Window](http://docs.google.com/java/awt/Window.html) owner, [GraphicsConfiguration](http://docs.google.com/java/awt/GraphicsConfiguration.html) gc)            Constructs a new, initially invisible window with the specified owner Window and a GraphicsConfiguration of a screen device. |

| **Method Summary** | |
| --- | --- |
| void | [**addNotify**](http://docs.google.com/java/awt/Window.html#addNotify())()            Makes this Window displayable by creating the connection to its native screen resource. |
| void | [**addPropertyChangeListener**](http://docs.google.com/java/awt/Window.html#addPropertyChangeListener(java.beans.PropertyChangeListener))([PropertyChangeListener](http://docs.google.com/java/beans/PropertyChangeListener.html) listener)            Adds a PropertyChangeListener to the listener list. |
| void | [**addPropertyChangeListener**](http://docs.google.com/java/awt/Window.html#addPropertyChangeListener(java.lang.String,%20java.beans.PropertyChangeListener))([String](http://docs.google.com/java/lang/String.html) propertyName, [PropertyChangeListener](http://docs.google.com/java/beans/PropertyChangeListener.html) listener)            Adds a PropertyChangeListener to the listener list for a specific property. |
| void | [**addWindowFocusListener**](http://docs.google.com/java/awt/Window.html#addWindowFocusListener(java.awt.event.WindowFocusListener))([WindowFocusListener](http://docs.google.com/java/awt/event/WindowFocusListener.html) l)            Adds the specified window focus listener to receive window events from this window. |
| void | [**addWindowListener**](http://docs.google.com/java/awt/Window.html#addWindowListener(java.awt.event.WindowListener))([WindowListener](http://docs.google.com/java/awt/event/WindowListener.html) l)            Adds the specified window listener to receive window events from this window. |
| void | [**addWindowStateListener**](http://docs.google.com/java/awt/Window.html#addWindowStateListener(java.awt.event.WindowStateListener))([WindowStateListener](http://docs.google.com/java/awt/event/WindowStateListener.html) l)            Adds the specified window state listener to receive window events from this window. |
| void | [**applyResourceBundle**](http://docs.google.com/java/awt/Window.html#applyResourceBundle(java.util.ResourceBundle))([ResourceBundle](http://docs.google.com/java/util/ResourceBundle.html) rb)  **Deprecated.** *As of J2SE 1.4, replaced by* [*Component.applyComponentOrientation*](http://docs.google.com/java/awt/Component.html#applyComponentOrientation(java.awt.ComponentOrientation))*.* |
| void | [**applyResourceBundle**](http://docs.google.com/java/awt/Window.html#applyResourceBundle(java.lang.String))([String](http://docs.google.com/java/lang/String.html) rbName)  **Deprecated.** *As of J2SE 1.4, replaced by* [*Component.applyComponentOrientation*](http://docs.google.com/java/awt/Component.html#applyComponentOrientation(java.awt.ComponentOrientation))*.* |
| void | [**createBufferStrategy**](http://docs.google.com/java/awt/Window.html#createBufferStrategy(int))(int numBuffers)            Creates a new strategy for multi-buffering on this component. |
| void | [**createBufferStrategy**](http://docs.google.com/java/awt/Window.html#createBufferStrategy(int,%20java.awt.BufferCapabilities))(int numBuffers, [BufferCapabilities](http://docs.google.com/java/awt/BufferCapabilities.html) caps)            Creates a new strategy for multi-buffering on this component with the required buffer capabilities. |
| void | [**dispose**](http://docs.google.com/java/awt/Window.html#dispose())()            Releases all of the native screen resources used by this Window, its subcomponents, and all of its owned children. |
| [AccessibleContext](http://docs.google.com/javax/accessibility/AccessibleContext.html) | [**getAccessibleContext**](http://docs.google.com/java/awt/Window.html#getAccessibleContext())()            Gets the AccessibleContext associated with this Window. |
| [BufferStrategy](http://docs.google.com/java/awt/image/BufferStrategy.html) | [**getBufferStrategy**](http://docs.google.com/java/awt/Window.html#getBufferStrategy())()            Returns the BufferStrategy used by this component. |
| boolean | [**getFocusableWindowState**](http://docs.google.com/java/awt/Window.html#getFocusableWindowState())()            Returns whether this Window can become the focused Window if it meets the other requirements outlined in isFocusableWindow. |
| [Container](http://docs.google.com/java/awt/Container.html) | [**getFocusCycleRootAncestor**](http://docs.google.com/java/awt/Window.html#getFocusCycleRootAncestor())()            Always returns null because Windows have no ancestors; they represent the top of the Component hierarchy. |
| [Component](http://docs.google.com/java/awt/Component.html) | [**getFocusOwner**](http://docs.google.com/java/awt/Window.html#getFocusOwner())()            Returns the child Component of this Window that has focus if this Window is focused; returns null otherwise. |
| [Set](http://docs.google.com/java/util/Set.html)<[AWTKeyStroke](http://docs.google.com/java/awt/AWTKeyStroke.html)> | [**getFocusTraversalKeys**](http://docs.google.com/java/awt/Window.html#getFocusTraversalKeys(int))(int id)            Gets a focus traversal key for this Window. |
| [GraphicsConfiguration](http://docs.google.com/java/awt/GraphicsConfiguration.html) | [**getGraphicsConfiguration**](http://docs.google.com/java/awt/Window.html#getGraphicsConfiguration())()            This method returns the GraphicsConfiguration used by this Window. |
| [List](http://docs.google.com/java/util/List.html)<[Image](http://docs.google.com/java/awt/Image.html)> | [**getIconImages**](http://docs.google.com/java/awt/Window.html#getIconImages())()            Returns the sequence of images to be displayed as the icon for this window. |
| [InputContext](http://docs.google.com/java/awt/im/InputContext.html) | [**getInputContext**](http://docs.google.com/java/awt/Window.html#getInputContext())()            Gets the input context for this window. |
| | <T extends [EventListener](http://docs.google.com/java/util/EventListener.html)>  T[] | | --- | | [**getListeners**](http://docs.google.com/java/awt/Window.html#getListeners(java.lang.Class))([Class](http://docs.google.com/java/lang/Class.html)<T> listenerType)            Returns an array of all the objects currently registered as *Foo*Listeners upon this Window. |
| [Locale](http://docs.google.com/java/util/Locale.html) | [**getLocale**](http://docs.google.com/java/awt/Window.html#getLocale())()            Gets the Locale object that is associated with this window, if the locale has been set. |
| [Dialog.ModalExclusionType](http://docs.google.com/java/awt/Dialog.ModalExclusionType.html) | [**getModalExclusionType**](http://docs.google.com/java/awt/Window.html#getModalExclusionType())()            Returns the modal exclusion type of this window. |
| [Component](http://docs.google.com/java/awt/Component.html) | [**getMostRecentFocusOwner**](http://docs.google.com/java/awt/Window.html#getMostRecentFocusOwner())()            Returns the child Component of this Window that will receive the focus when this Window is focused. |
| [Window](http://docs.google.com/java/awt/Window.html)[] | [**getOwnedWindows**](http://docs.google.com/java/awt/Window.html#getOwnedWindows())()            Return an array containing all the windows this window currently owns. |
| [Window](http://docs.google.com/java/awt/Window.html) | [**getOwner**](http://docs.google.com/java/awt/Window.html#getOwner())()            Returns the owner of this window. |
| static [Window](http://docs.google.com/java/awt/Window.html)[] | [**getOwnerlessWindows**](http://docs.google.com/java/awt/Window.html#getOwnerlessWindows())()            Returns an array of all Windows created by this application that have no owner. |
| [Toolkit](http://docs.google.com/java/awt/Toolkit.html) | [**getToolkit**](http://docs.google.com/java/awt/Window.html#getToolkit())()            Returns the toolkit of this frame. |
| [String](http://docs.google.com/java/lang/String.html) | [**getWarningString**](http://docs.google.com/java/awt/Window.html#getWarningString())()            Gets the warning string that is displayed with this window. |
| [WindowFocusListener](http://docs.google.com/java/awt/event/WindowFocusListener.html)[] | [**getWindowFocusListeners**](http://docs.google.com/java/awt/Window.html#getWindowFocusListeners())()            Returns an array of all the window focus listeners registered on this window. |
| [WindowListener](http://docs.google.com/java/awt/event/WindowListener.html)[] | [**getWindowListeners**](http://docs.google.com/java/awt/Window.html#getWindowListeners())()            Returns an array of all the window listeners registered on this window. |
| static [Window](http://docs.google.com/java/awt/Window.html)[] | [**getWindows**](http://docs.google.com/java/awt/Window.html#getWindows())()            Returns an array of all Windows, both owned and ownerless, created by this application. |
| [WindowStateListener](http://docs.google.com/java/awt/event/WindowStateListener.html)[] | [**getWindowStateListeners**](http://docs.google.com/java/awt/Window.html#getWindowStateListeners())()            Returns an array of all the window state listeners registered on this window. |
| void | [**hide**](http://docs.google.com/java/awt/Window.html#hide())()  **Deprecated.** *As of JDK version 1.5, replaced by* [*setVisible(boolean)*](http://docs.google.com/java/awt/Window.html#setVisible(boolean))*.* |
| boolean | [**isActive**](http://docs.google.com/java/awt/Window.html#isActive())()            Returns whether this Window is active. |
| boolean | [**isAlwaysOnTop**](http://docs.google.com/java/awt/Window.html#isAlwaysOnTop())()            Returns whether this window is an always-on-top window. |
| boolean | [**isAlwaysOnTopSupported**](http://docs.google.com/java/awt/Window.html#isAlwaysOnTopSupported())()            Returns whether the always-on-top mode is supported for this window. |
| boolean | [**isFocusableWindow**](http://docs.google.com/java/awt/Window.html#isFocusableWindow())()            Returns whether this Window can become the focused Window, that is, whether this Window or any of its subcomponents can become the focus owner. |
| boolean | [**isFocusCycleRoot**](http://docs.google.com/java/awt/Window.html#isFocusCycleRoot())()            Always returns true because all Windows must be roots of a focus traversal cycle. |
| boolean | [**isFocused**](http://docs.google.com/java/awt/Window.html#isFocused())()            Returns whether this Window is focused. |
| boolean | [**isLocationByPlatform**](http://docs.google.com/java/awt/Window.html#isLocationByPlatform())()            Returns true if this Window will appear at the default location for the native windowing system the next time this Window is made visible. |
| boolean | [**isShowing**](http://docs.google.com/java/awt/Window.html#isShowing())()            Checks if this Window is showing on screen. |
| void | [**pack**](http://docs.google.com/java/awt/Window.html#pack())()            Causes this Window to be sized to fit the preferred size and layouts of its subcomponents. |
| boolean | [**postEvent**](http://docs.google.com/java/awt/Window.html#postEvent(java.awt.Event))([Event](http://docs.google.com/java/awt/Event.html) e)  **Deprecated.** *As of JDK version 1.1 replaced by dispatchEvent(AWTEvent).* |
| protected  void | [**processEvent**](http://docs.google.com/java/awt/Window.html#processEvent(java.awt.AWTEvent))([AWTEvent](http://docs.google.com/java/awt/AWTEvent.html) e)            Processes events on this window. |
| protected  void | [**processWindowEvent**](http://docs.google.com/java/awt/Window.html#processWindowEvent(java.awt.event.WindowEvent))([WindowEvent](http://docs.google.com/java/awt/event/WindowEvent.html) e)            Processes window events occurring on this window by dispatching them to any registered WindowListener objects. |
| protected  void | [**processWindowFocusEvent**](http://docs.google.com/java/awt/Window.html#processWindowFocusEvent(java.awt.event.WindowEvent))([WindowEvent](http://docs.google.com/java/awt/event/WindowEvent.html) e)            Processes window focus event occuring on this window by dispatching them to any registered WindowFocusListener objects. |
| protected  void | [**processWindowStateEvent**](http://docs.google.com/java/awt/Window.html#processWindowStateEvent(java.awt.event.WindowEvent))([WindowEvent](http://docs.google.com/java/awt/event/WindowEvent.html) e)            Processes window state event occuring on this window by dispatching them to any registered WindowStateListener objects. |
| void | [**removeNotify**](http://docs.google.com/java/awt/Window.html#removeNotify())()            Makes this Container undisplayable by removing its connection to its native screen resource. |
| void | [**removeWindowFocusListener**](http://docs.google.com/java/awt/Window.html#removeWindowFocusListener(java.awt.event.WindowFocusListener))([WindowFocusListener](http://docs.google.com/java/awt/event/WindowFocusListener.html) l)            Removes the specified window focus listener so that it no longer receives window events from this window. |
| void | [**removeWindowListener**](http://docs.google.com/java/awt/Window.html#removeWindowListener(java.awt.event.WindowListener))([WindowListener](http://docs.google.com/java/awt/event/WindowListener.html) l)            Removes the specified window listener so that it no longer receives window events from this window. |
| void | [**removeWindowStateListener**](http://docs.google.com/java/awt/Window.html#removeWindowStateListener(java.awt.event.WindowStateListener))([WindowStateListener](http://docs.google.com/java/awt/event/WindowStateListener.html) l)            Removes the specified window state listener so that it no longer receives window events from this window. |
| void | [**reshape**](http://docs.google.com/java/awt/Window.html#reshape(int,%20int,%20int,%20int))(int x, int y, int width, int height)  **Deprecated.** *As of JDK version 1.1, replaced by setBounds(int, int, int, int).* |
| void | [**setAlwaysOnTop**](http://docs.google.com/java/awt/Window.html#setAlwaysOnTop(boolean))(boolean alwaysOnTop)            Sets whether this window should always be above other windows. |
| void | [**setBounds**](http://docs.google.com/java/awt/Window.html#setBounds(int,%20int,%20int,%20int))(int x, int y, int width, int height)            Moves and resizes this component. |
| void | [**setBounds**](http://docs.google.com/java/awt/Window.html#setBounds(java.awt.Rectangle))([Rectangle](http://docs.google.com/java/awt/Rectangle.html) r)            Moves and resizes this component to conform to the new bounding rectangle r. |
| void | [**setCursor**](http://docs.google.com/java/awt/Window.html#setCursor(java.awt.Cursor))([Cursor](http://docs.google.com/java/awt/Cursor.html) cursor)            Set the cursor image to a specified cursor. |
| void | [**setFocusableWindowState**](http://docs.google.com/java/awt/Window.html#setFocusableWindowState(boolean))(boolean focusableWindowState)            Sets whether this Window can become the focused Window if it meets the other requirements outlined in isFocusableWindow. |
| void | [**setFocusCycleRoot**](http://docs.google.com/java/awt/Window.html#setFocusCycleRoot(boolean))(boolean focusCycleRoot)            Does nothing because Windows must always be roots of a focus traversal cycle. |
| void | [**setIconImage**](http://docs.google.com/java/awt/Window.html#setIconImage(java.awt.Image))([Image](http://docs.google.com/java/awt/Image.html) image)            Sets the image to be displayed as the icon for this window. |
| void | [**setIconImages**](http://docs.google.com/java/awt/Window.html#setIconImages(java.util.List))([List](http://docs.google.com/java/util/List.html)<? extends [Image](http://docs.google.com/java/awt/Image.html)> icons)            Sets the sequence of images to be displayed as the icon for this window. |
| void | [**setLocationByPlatform**](http://docs.google.com/java/awt/Window.html#setLocationByPlatform(boolean))(boolean locationByPlatform)            Sets whether this Window should appear at the default location for the native windowing system or at the current location (returned by getLocation) the next time the Window is made visible. |
| void | [**setLocationRelativeTo**](http://docs.google.com/java/awt/Window.html#setLocationRelativeTo(java.awt.Component))([Component](http://docs.google.com/java/awt/Component.html) c)            Sets the location of the window relative to the specified component. |
| void | [**setMinimumSize**](http://docs.google.com/java/awt/Window.html#setMinimumSize(java.awt.Dimension))([Dimension](http://docs.google.com/java/awt/Dimension.html) minimumSize)            Sets the minimum size of this window to a constant value. |
| void | [**setModalExclusionType**](http://docs.google.com/java/awt/Window.html#setModalExclusionType(java.awt.Dialog.ModalExclusionType))([Dialog.ModalExclusionType](http://docs.google.com/java/awt/Dialog.ModalExclusionType.html) exclusionType)            Specifies the modal exclusion type for this window. |
| void | [**setSize**](http://docs.google.com/java/awt/Window.html#setSize(java.awt.Dimension))([Dimension](http://docs.google.com/java/awt/Dimension.html) d)            Resizes this component so that it has width d.width and height d.height. |
| void | [**setSize**](http://docs.google.com/java/awt/Window.html#setSize(int,%20int))(int width, int height)            Resizes this component so that it has width width and height height. |
| void | [**setVisible**](http://docs.google.com/java/awt/Window.html#setVisible(boolean))(boolean b)            Shows or hides this Window depending on the value of parameter b. |
| void | [**show**](http://docs.google.com/java/awt/Window.html#show())()  **Deprecated.** *As of JDK version 1.5, replaced by* [*setVisible(boolean)*](http://docs.google.com/java/awt/Window.html#setVisible(boolean))*.* |
| void | [**toBack**](http://docs.google.com/java/awt/Window.html#toBack())()            If this Window is visible, sends this Window to the back and may cause it to lose focus or activation if it is the focused or active Window. |
| void | [**toFront**](http://docs.google.com/java/awt/Window.html#toFront())()            If this Window is visible, brings this Window to the front and may make it the focused Window. |

| **Methods inherited from class java.awt.**[**Container**](http://docs.google.com/java/awt/Container.html) |
| --- |
| [add](http://docs.google.com/java/awt/Container.html#add(java.awt.Component)), [add](http://docs.google.com/java/awt/Container.html#add(java.awt.Component,%20int)), [add](http://docs.google.com/java/awt/Container.html#add(java.awt.Component,%20java.lang.Object)), [add](http://docs.google.com/java/awt/Container.html#add(java.awt.Component,%20java.lang.Object,%20int)), [add](http://docs.google.com/java/awt/Container.html#add(java.lang.String,%20java.awt.Component)), [addContainerListener](http://docs.google.com/java/awt/Container.html#addContainerListener(java.awt.event.ContainerListener)), [addImpl](http://docs.google.com/java/awt/Container.html#addImpl(java.awt.Component,%20java.lang.Object,%20int)), [applyComponentOrientation](http://docs.google.com/java/awt/Container.html#applyComponentOrientation(java.awt.ComponentOrientation)), [areFocusTraversalKeysSet](http://docs.google.com/java/awt/Container.html#areFocusTraversalKeysSet(int)), [countComponents](http://docs.google.com/java/awt/Container.html#countComponents()), [deliverEvent](http://docs.google.com/java/awt/Container.html#deliverEvent(java.awt.Event)), [doLayout](http://docs.google.com/java/awt/Container.html#doLayout()), [findComponentAt](http://docs.google.com/java/awt/Container.html#findComponentAt(int,%20int)), [findComponentAt](http://docs.google.com/java/awt/Container.html#findComponentAt(java.awt.Point)), [getAlignmentX](http://docs.google.com/java/awt/Container.html#getAlignmentX()), [getAlignmentY](http://docs.google.com/java/awt/Container.html#getAlignmentY()), [getComponent](http://docs.google.com/java/awt/Container.html#getComponent(int)), [getComponentAt](http://docs.google.com/java/awt/Container.html#getComponentAt(int,%20int)), [getComponentAt](http://docs.google.com/java/awt/Container.html#getComponentAt(java.awt.Point)), [getComponentCount](http://docs.google.com/java/awt/Container.html#getComponentCount()), [getComponents](http://docs.google.com/java/awt/Container.html#getComponents()), [getComponentZOrder](http://docs.google.com/java/awt/Container.html#getComponentZOrder(java.awt.Component)), [getContainerListeners](http://docs.google.com/java/awt/Container.html#getContainerListeners()), [getFocusTraversalPolicy](http://docs.google.com/java/awt/Container.html#getFocusTraversalPolicy()), [getInsets](http://docs.google.com/java/awt/Container.html#getInsets()), [getLayout](http://docs.google.com/java/awt/Container.html#getLayout()), [getMaximumSize](http://docs.google.com/java/awt/Container.html#getMaximumSize()), [getMinimumSize](http://docs.google.com/java/awt/Container.html#getMinimumSize()), [getMousePosition](http://docs.google.com/java/awt/Container.html#getMousePosition(boolean)), [getPreferredSize](http://docs.google.com/java/awt/Container.html#getPreferredSize()), [insets](http://docs.google.com/java/awt/Container.html#insets()), [invalidate](http://docs.google.com/java/awt/Container.html#invalidate()), [isAncestorOf](http://docs.google.com/java/awt/Container.html#isAncestorOf(java.awt.Component)), [isFocusCycleRoot](http://docs.google.com/java/awt/Container.html#isFocusCycleRoot(java.awt.Container)), [isFocusTraversalPolicyProvider](http://docs.google.com/java/awt/Container.html#isFocusTraversalPolicyProvider()), [isFocusTraversalPolicySet](http://docs.google.com/java/awt/Container.html#isFocusTraversalPolicySet()), [layout](http://docs.google.com/java/awt/Container.html#layout()), [list](http://docs.google.com/java/awt/Container.html#list(java.io.PrintStream,%20int)), [list](http://docs.google.com/java/awt/Container.html#list(java.io.PrintWriter,%20int)), [locate](http://docs.google.com/java/awt/Container.html#locate(int,%20int)), [minimumSize](http://docs.google.com/java/awt/Container.html#minimumSize()), [paint](http://docs.google.com/java/awt/Container.html#paint(java.awt.Graphics)), [paintComponents](http://docs.google.com/java/awt/Container.html#paintComponents(java.awt.Graphics)), [paramString](http://docs.google.com/java/awt/Container.html#paramString()), [preferredSize](http://docs.google.com/java/awt/Container.html#preferredSize()), [print](http://docs.google.com/java/awt/Container.html#print(java.awt.Graphics)), [printComponents](http://docs.google.com/java/awt/Container.html#printComponents(java.awt.Graphics)), [processContainerEvent](http://docs.google.com/java/awt/Container.html#processContainerEvent(java.awt.event.ContainerEvent)), [remove](http://docs.google.com/java/awt/Container.html#remove(java.awt.Component)), [remove](http://docs.google.com/java/awt/Container.html#remove(int)), [removeAll](http://docs.google.com/java/awt/Container.html#removeAll()), [removeContainerListener](http://docs.google.com/java/awt/Container.html#removeContainerListener(java.awt.event.ContainerListener)), [setComponentZOrder](http://docs.google.com/java/awt/Container.html#setComponentZOrder(java.awt.Component,%20int)), [setFocusTraversalKeys](http://docs.google.com/java/awt/Container.html#setFocusTraversalKeys(int,%20java.util.Set)), [setFocusTraversalPolicy](http://docs.google.com/java/awt/Container.html#setFocusTraversalPolicy(java.awt.FocusTraversalPolicy)), [setFocusTraversalPolicyProvider](http://docs.google.com/java/awt/Container.html#setFocusTraversalPolicyProvider(boolean)), [setFont](http://docs.google.com/java/awt/Container.html#setFont(java.awt.Font)), [setLayout](http://docs.google.com/java/awt/Container.html#setLayout(java.awt.LayoutManager)), [transferFocusBackward](http://docs.google.com/java/awt/Container.html#transferFocusBackward()), [transferFocusDownCycle](http://docs.google.com/java/awt/Container.html#transferFocusDownCycle()), [update](http://docs.google.com/java/awt/Container.html#update(java.awt.Graphics)), [validate](http://docs.google.com/java/awt/Container.html#validate()), [validateTree](http://docs.google.com/java/awt/Container.html#validateTree()) |

| **Methods inherited from class java.awt.**[**Component**](http://docs.google.com/java/awt/Component.html) |
| --- |
| [action](http://docs.google.com/java/awt/Component.html#action(java.awt.Event,%20java.lang.Object)), [add](http://docs.google.com/java/awt/Component.html#add(java.awt.PopupMenu)), [addComponentListener](http://docs.google.com/java/awt/Component.html#addComponentListener(java.awt.event.ComponentListener)), [addFocusListener](http://docs.google.com/java/awt/Component.html#addFocusListener(java.awt.event.FocusListener)), [addHierarchyBoundsListener](http://docs.google.com/java/awt/Component.html#addHierarchyBoundsListener(java.awt.event.HierarchyBoundsListener)), [addHierarchyListener](http://docs.google.com/java/awt/Component.html#addHierarchyListener(java.awt.event.HierarchyListener)), [addInputMethodListener](http://docs.google.com/java/awt/Component.html#addInputMethodListener(java.awt.event.InputMethodListener)), [addKeyListener](http://docs.google.com/java/awt/Component.html#addKeyListener(java.awt.event.KeyListener)), [addMouseListener](http://docs.google.com/java/awt/Component.html#addMouseListener(java.awt.event.MouseListener)), [addMouseMotionListener](http://docs.google.com/java/awt/Component.html#addMouseMotionListener(java.awt.event.MouseMotionListener)), [addMouseWheelListener](http://docs.google.com/java/awt/Component.html#addMouseWheelListener(java.awt.event.MouseWheelListener)), [bounds](http://docs.google.com/java/awt/Component.html#bounds()), [checkImage](http://docs.google.com/java/awt/Component.html#checkImage(java.awt.Image,%20java.awt.image.ImageObserver)), [checkImage](http://docs.google.com/java/awt/Component.html#checkImage(java.awt.Image,%20int,%20int,%20java.awt.image.ImageObserver)), [coalesceEvents](http://docs.google.com/java/awt/Component.html#coalesceEvents(java.awt.AWTEvent,%20java.awt.AWTEvent)), [contains](http://docs.google.com/java/awt/Component.html#contains(int,%20int)), [contains](http://docs.google.com/java/awt/Component.html#contains(java.awt.Point)), [createImage](http://docs.google.com/java/awt/Component.html#createImage(java.awt.image.ImageProducer)), [createImage](http://docs.google.com/java/awt/Component.html#createImage(int,%20int)), [createVolatileImage](http://docs.google.com/java/awt/Component.html#createVolatileImage(int,%20int)), [createVolatileImage](http://docs.google.com/java/awt/Component.html#createVolatileImage(int,%20int,%20java.awt.ImageCapabilities)), [disable](http://docs.google.com/java/awt/Component.html#disable()), [disableEvents](http://docs.google.com/java/awt/Component.html#disableEvents(long)), [dispatchEvent](http://docs.google.com/java/awt/Component.html#dispatchEvent(java.awt.AWTEvent)), [enable](http://docs.google.com/java/awt/Component.html#enable()), [enable](http://docs.google.com/java/awt/Component.html#enable(boolean)), [enableEvents](http://docs.google.com/java/awt/Component.html#enableEvents(long)), [enableInputMethods](http://docs.google.com/java/awt/Component.html#enableInputMethods(boolean)), [firePropertyChange](http://docs.google.com/java/awt/Component.html#firePropertyChange(java.lang.String,%20boolean,%20boolean)), [firePropertyChange](http://docs.google.com/java/awt/Component.html#firePropertyChange(java.lang.String,%20byte,%20byte)), [firePropertyChange](http://docs.google.com/java/awt/Component.html#firePropertyChange(java.lang.String,%20char,%20char)), [firePropertyChange](http://docs.google.com/java/awt/Component.html#firePropertyChange(java.lang.String,%20double,%20double)), [firePropertyChange](http://docs.google.com/java/awt/Component.html#firePropertyChange(java.lang.String,%20float,%20float)), [firePropertyChange](http://docs.google.com/java/awt/Component.html#firePropertyChange(java.lang.String,%20int,%20int)), [firePropertyChange](http://docs.google.com/java/awt/Component.html#firePropertyChange(java.lang.String,%20long,%20long)), [firePropertyChange](http://docs.google.com/java/awt/Component.html#firePropertyChange(java.lang.String,%20java.lang.Object,%20java.lang.Object)), [firePropertyChange](http://docs.google.com/java/awt/Component.html#firePropertyChange(java.lang.String,%20short,%20short)), [getBackground](http://docs.google.com/java/awt/Component.html#getBackground()), [getBaseline](http://docs.google.com/java/awt/Component.html#getBaseline(int,%20int)), [getBaselineResizeBehavior](http://docs.google.com/java/awt/Component.html#getBaselineResizeBehavior()), [getBounds](http://docs.google.com/java/awt/Component.html#getBounds()), [getBounds](http://docs.google.com/java/awt/Component.html#getBounds(java.awt.Rectangle)), [getColorModel](http://docs.google.com/java/awt/Component.html#getColorModel()), [getComponentListeners](http://docs.google.com/java/awt/Component.html#getComponentListeners()), [getComponentOrientation](http://docs.google.com/java/awt/Component.html#getComponentOrientation()), [getCursor](http://docs.google.com/java/awt/Component.html#getCursor()), [getDropTarget](http://docs.google.com/java/awt/Component.html#getDropTarget()), [getFocusListeners](http://docs.google.com/java/awt/Component.html#getFocusListeners()), [getFocusTraversalKeysEnabled](http://docs.google.com/java/awt/Component.html#getFocusTraversalKeysEnabled()), [getFont](http://docs.google.com/java/awt/Component.html#getFont()), [getFontMetrics](http://docs.google.com/java/awt/Component.html#getFontMetrics(java.awt.Font)), [getForeground](http://docs.google.com/java/awt/Component.html#getForeground()), [getGraphics](http://docs.google.com/java/awt/Component.html#getGraphics()), [getHeight](http://docs.google.com/java/awt/Component.html#getHeight()), [getHierarchyBoundsListeners](http://docs.google.com/java/awt/Component.html#getHierarchyBoundsListeners()), [getHierarchyListeners](http://docs.google.com/java/awt/Component.html#getHierarchyListeners()), [getIgnoreRepaint](http://docs.google.com/java/awt/Component.html#getIgnoreRepaint()), [getInputMethodListeners](http://docs.google.com/java/awt/Component.html#getInputMethodListeners()), [getInputMethodRequests](http://docs.google.com/java/awt/Component.html#getInputMethodRequests()), [getKeyListeners](http://docs.google.com/java/awt/Component.html#getKeyListeners()), [getLocation](http://docs.google.com/java/awt/Component.html#getLocation()), [getLocation](http://docs.google.com/java/awt/Component.html#getLocation(java.awt.Point)), [getLocationOnScreen](http://docs.google.com/java/awt/Component.html#getLocationOnScreen()), [getMouseListeners](http://docs.google.com/java/awt/Component.html#getMouseListeners()), [getMouseMotionListeners](http://docs.google.com/java/awt/Component.html#getMouseMotionListeners()), [getMousePosition](http://docs.google.com/java/awt/Component.html#getMousePosition()), [getMouseWheelListeners](http://docs.google.com/java/awt/Component.html#getMouseWheelListeners()), [getName](http://docs.google.com/java/awt/Component.html#getName()), [getParent](http://docs.google.com/java/awt/Component.html#getParent()), [getPeer](http://docs.google.com/java/awt/Component.html#getPeer()), [getPropertyChangeListeners](http://docs.google.com/java/awt/Component.html#getPropertyChangeListeners()), [getPropertyChangeListeners](http://docs.google.com/java/awt/Component.html#getPropertyChangeListeners(java.lang.String)), [getSize](http://docs.google.com/java/awt/Component.html#getSize()), [getSize](http://docs.google.com/java/awt/Component.html#getSize(java.awt.Dimension)), [getTreeLock](http://docs.google.com/java/awt/Component.html#getTreeLock()), [getWidth](http://docs.google.com/java/awt/Component.html#getWidth()), [getX](http://docs.google.com/java/awt/Component.html#getX()), [getY](http://docs.google.com/java/awt/Component.html#getY()), [gotFocus](http://docs.google.com/java/awt/Component.html#gotFocus(java.awt.Event,%20java.lang.Object)), [handleEvent](http://docs.google.com/java/awt/Component.html#handleEvent(java.awt.Event)), [hasFocus](http://docs.google.com/java/awt/Component.html#hasFocus()), [imageUpdate](http://docs.google.com/java/awt/Component.html#imageUpdate(java.awt.Image,%20int,%20int,%20int,%20int,%20int)), [inside](http://docs.google.com/java/awt/Component.html#inside(int,%20int)), [isBackgroundSet](http://docs.google.com/java/awt/Component.html#isBackgroundSet()), [isCursorSet](http://docs.google.com/java/awt/Component.html#isCursorSet()), [isDisplayable](http://docs.google.com/java/awt/Component.html#isDisplayable()), [isDoubleBuffered](http://docs.google.com/java/awt/Component.html#isDoubleBuffered()), [isEnabled](http://docs.google.com/java/awt/Component.html#isEnabled()), [isFocusable](http://docs.google.com/java/awt/Component.html#isFocusable()), [isFocusOwner](http://docs.google.com/java/awt/Component.html#isFocusOwner()), [isFocusTraversable](http://docs.google.com/java/awt/Component.html#isFocusTraversable()), [isFontSet](http://docs.google.com/java/awt/Component.html#isFontSet()), [isForegroundSet](http://docs.google.com/java/awt/Component.html#isForegroundSet()), [isLightweight](http://docs.google.com/java/awt/Component.html#isLightweight()), [isMaximumSizeSet](http://docs.google.com/java/awt/Component.html#isMaximumSizeSet()), [isMinimumSizeSet](http://docs.google.com/java/awt/Component.html#isMinimumSizeSet()), [isOpaque](http://docs.google.com/java/awt/Component.html#isOpaque()), [isPreferredSizeSet](http://docs.google.com/java/awt/Component.html#isPreferredSizeSet()), [isValid](http://docs.google.com/java/awt/Component.html#isValid()), [isVisible](http://docs.google.com/java/awt/Component.html#isVisible()), [keyDown](http://docs.google.com/java/awt/Component.html#keyDown(java.awt.Event,%20int)), [keyUp](http://docs.google.com/java/awt/Component.html#keyUp(java.awt.Event,%20int)), [list](http://docs.google.com/java/awt/Component.html#list()), [list](http://docs.google.com/java/awt/Component.html#list(java.io.PrintStream)), [list](http://docs.google.com/java/awt/Component.html#list(java.io.PrintWriter)), [location](http://docs.google.com/java/awt/Component.html#location()), [lostFocus](http://docs.google.com/java/awt/Component.html#lostFocus(java.awt.Event,%20java.lang.Object)), [mouseDown](http://docs.google.com/java/awt/Component.html#mouseDown(java.awt.Event,%20int,%20int)), [mouseDrag](http://docs.google.com/java/awt/Component.html#mouseDrag(java.awt.Event,%20int,%20int)), [mouseEnter](http://docs.google.com/java/awt/Component.html#mouseEnter(java.awt.Event,%20int,%20int)), [mouseExit](http://docs.google.com/java/awt/Component.html#mouseExit(java.awt.Event,%20int,%20int)), [mouseMove](http://docs.google.com/java/awt/Component.html#mouseMove(java.awt.Event,%20int,%20int)), [mouseUp](http://docs.google.com/java/awt/Component.html#mouseUp(java.awt.Event,%20int,%20int)), [move](http://docs.google.com/java/awt/Component.html#move(int,%20int)), [nextFocus](http://docs.google.com/java/awt/Component.html#nextFocus()), [paintAll](http://docs.google.com/java/awt/Component.html#paintAll(java.awt.Graphics)), [prepareImage](http://docs.google.com/java/awt/Component.html#prepareImage(java.awt.Image,%20java.awt.image.ImageObserver)), [prepareImage](http://docs.google.com/java/awt/Component.html#prepareImage(java.awt.Image,%20int,%20int,%20java.awt.image.ImageObserver)), [printAll](http://docs.google.com/java/awt/Component.html#printAll(java.awt.Graphics)), [processComponentEvent](http://docs.google.com/java/awt/Component.html#processComponentEvent(java.awt.event.ComponentEvent)), [processFocusEvent](http://docs.google.com/java/awt/Component.html#processFocusEvent(java.awt.event.FocusEvent)), [processHierarchyBoundsEvent](http://docs.google.com/java/awt/Component.html#processHierarchyBoundsEvent(java.awt.event.HierarchyEvent)), [processHierarchyEvent](http://docs.google.com/java/awt/Component.html#processHierarchyEvent(java.awt.event.HierarchyEvent)), [processInputMethodEvent](http://docs.google.com/java/awt/Component.html#processInputMethodEvent(java.awt.event.InputMethodEvent)), [processKeyEvent](http://docs.google.com/java/awt/Component.html#processKeyEvent(java.awt.event.KeyEvent)), [processMouseEvent](http://docs.google.com/java/awt/Component.html#processMouseEvent(java.awt.event.MouseEvent)), [processMouseMotionEvent](http://docs.google.com/java/awt/Component.html#processMouseMotionEvent(java.awt.event.MouseEvent)), [processMouseWheelEvent](http://docs.google.com/java/awt/Component.html#processMouseWheelEvent(java.awt.event.MouseWheelEvent)), [remove](http://docs.google.com/java/awt/Component.html#remove(java.awt.MenuComponent)), [removeComponentListener](http://docs.google.com/java/awt/Component.html#removeComponentListener(java.awt.event.ComponentListener)), [removeFocusListener](http://docs.google.com/java/awt/Component.html#removeFocusListener(java.awt.event.FocusListener)), [removeHierarchyBoundsListener](http://docs.google.com/java/awt/Component.html#removeHierarchyBoundsListener(java.awt.event.HierarchyBoundsListener)), [removeHierarchyListener](http://docs.google.com/java/awt/Component.html#removeHierarchyListener(java.awt.event.HierarchyListener)), [removeInputMethodListener](http://docs.google.com/java/awt/Component.html#removeInputMethodListener(java.awt.event.InputMethodListener)), [removeKeyListener](http://docs.google.com/java/awt/Component.html#removeKeyListener(java.awt.event.KeyListener)), [removeMouseListener](http://docs.google.com/java/awt/Component.html#removeMouseListener(java.awt.event.MouseListener)), [removeMouseMotionListener](http://docs.google.com/java/awt/Component.html#removeMouseMotionListener(java.awt.event.MouseMotionListener)), [removeMouseWheelListener](http://docs.google.com/java/awt/Component.html#removeMouseWheelListener(java.awt.event.MouseWheelListener)), [removePropertyChangeListener](http://docs.google.com/java/awt/Component.html#removePropertyChangeListener(java.beans.PropertyChangeListener)), [removePropertyChangeListener](http://docs.google.com/java/awt/Component.html#removePropertyChangeListener(java.lang.String,%20java.beans.PropertyChangeListener)), [repaint](http://docs.google.com/java/awt/Component.html#repaint()), [repaint](http://docs.google.com/java/awt/Component.html#repaint(int,%20int,%20int,%20int)), [repaint](http://docs.google.com/java/awt/Component.html#repaint(long)), [repaint](http://docs.google.com/java/awt/Component.html#repaint(long,%20int,%20int,%20int,%20int)), [requestFocus](http://docs.google.com/java/awt/Component.html#requestFocus()), [requestFocus](http://docs.google.com/java/awt/Component.html#requestFocus(boolean)), [requestFocusInWindow](http://docs.google.com/java/awt/Component.html#requestFocusInWindow()), [requestFocusInWindow](http://docs.google.com/java/awt/Component.html#requestFocusInWindow(boolean)), [resize](http://docs.google.com/java/awt/Component.html#resize(java.awt.Dimension)), [resize](http://docs.google.com/java/awt/Component.html#resize(int,%20int)), [setBackground](http://docs.google.com/java/awt/Component.html#setBackground(java.awt.Color)), [setComponentOrientation](http://docs.google.com/java/awt/Component.html#setComponentOrientation(java.awt.ComponentOrientation)), [setDropTarget](http://docs.google.com/java/awt/Component.html#setDropTarget(java.awt.dnd.DropTarget)), [setEnabled](http://docs.google.com/java/awt/Component.html#setEnabled(boolean)), [setFocusable](http://docs.google.com/java/awt/Component.html#setFocusable(boolean)), [setFocusTraversalKeysEnabled](http://docs.google.com/java/awt/Component.html#setFocusTraversalKeysEnabled(boolean)), [setForeground](http://docs.google.com/java/awt/Component.html#setForeground(java.awt.Color)), [setIgnoreRepaint](http://docs.google.com/java/awt/Component.html#setIgnoreRepaint(boolean)), [setLocale](http://docs.google.com/java/awt/Component.html#setLocale(java.util.Locale)), [setLocation](http://docs.google.com/java/awt/Component.html#setLocation(int,%20int)), [setLocation](http://docs.google.com/java/awt/Component.html#setLocation(java.awt.Point)), [setMaximumSize](http://docs.google.com/java/awt/Component.html#setMaximumSize(java.awt.Dimension)), [setName](http://docs.google.com/java/awt/Component.html#setName(java.lang.String)), [setPreferredSize](http://docs.google.com/java/awt/Component.html#setPreferredSize(java.awt.Dimension)), [show](http://docs.google.com/java/awt/Component.html#show(boolean)), [size](http://docs.google.com/java/awt/Component.html#size()), [toString](http://docs.google.com/java/awt/Component.html#toString()), [transferFocus](http://docs.google.com/java/awt/Component.html#transferFocus()), [transferFocusUpCycle](http://docs.google.com/java/awt/Component.html#transferFocusUpCycle()) |

| **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** |
| --- |

### Window

public **Window**([Frame](http://docs.google.com/java/awt/Frame.html) owner)

Constructs a new, initially invisible window with the specified Frame as its owner. The window will not be focusable unless its owner is showing on the screen.

If there is a security manager, this method first calls the security manager's checkTopLevelWindow method with this as its argument to determine whether or not the window must be displayed with a warning banner.

**Parameters:**owner - the Frame to act as owner or null if this window has no owner **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the owner's GraphicsConfiguration is not from a screen device [HeadlessException](http://docs.google.com/java/awt/HeadlessException.html) - when GraphicsEnvironment.isHeadless returns true**See Also:**[GraphicsEnvironment.isHeadless()](http://docs.google.com/java/awt/GraphicsEnvironment.html#isHeadless()), [SecurityManager.checkTopLevelWindow(java.lang.Object)](http://docs.google.com/java/lang/SecurityManager.html#checkTopLevelWindow(java.lang.Object)), [isShowing()](http://docs.google.com/java/awt/Window.html#isShowing())

### Window

public **Window**([Window](http://docs.google.com/java/awt/Window.html) owner)

Constructs a new, initially invisible window with the specified Window as its owner. This window will not be focusable unless its nearest owning Frame or Dialog is showing on the screen.

If there is a security manager, this method first calls the security manager's checkTopLevelWindow method with this as its argument to determine whether or not the window must be displayed with a warning banner.

**Parameters:**owner - the Window to act as owner or null if this window has no owner **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the owner's GraphicsConfiguration is not from a screen device [HeadlessException](http://docs.google.com/java/awt/HeadlessException.html) - when GraphicsEnvironment.isHeadless() returns true**Since:** 1.2 **See Also:**[GraphicsEnvironment.isHeadless()](http://docs.google.com/java/awt/GraphicsEnvironment.html#isHeadless()), [SecurityManager.checkTopLevelWindow(java.lang.Object)](http://docs.google.com/java/lang/SecurityManager.html#checkTopLevelWindow(java.lang.Object)), [isShowing()](http://docs.google.com/java/awt/Window.html#isShowing())

### Window

public **Window**([Window](http://docs.google.com/java/awt/Window.html) owner,  
 [GraphicsConfiguration](http://docs.google.com/java/awt/GraphicsConfiguration.html) gc)

Constructs a new, initially invisible window with the specified owner Window and a GraphicsConfiguration of a screen device. The Window will not be focusable unless its nearest owning Frame or Dialog is showing on the screen.

If there is a security manager, this method first calls the security manager's checkTopLevelWindow method with this as its argument to determine whether or not the window must be displayed with a warning banner.

**Parameters:**owner - the window to act as owner or null if this window has no ownergc - the GraphicsConfiguration of the target screen device; if gc is null, the system default GraphicsConfiguration is assumed **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if gc is not from a screen device [HeadlessException](http://docs.google.com/java/awt/HeadlessException.html) - when GraphicsEnvironment.isHeadless() returns true**Since:** 1.3 **See Also:**[GraphicsEnvironment.isHeadless()](http://docs.google.com/java/awt/GraphicsEnvironment.html#isHeadless()), [SecurityManager.checkTopLevelWindow(java.lang.Object)](http://docs.google.com/java/lang/SecurityManager.html#checkTopLevelWindow(java.lang.Object)), [GraphicsConfiguration.getBounds()](http://docs.google.com/java/awt/GraphicsConfiguration.html#getBounds()), [isShowing()](http://docs.google.com/java/awt/Window.html#isShowing())

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

### getIconImages

public [List](http://docs.google.com/java/util/List.html)<[Image](http://docs.google.com/java/awt/Image.html)> **getIconImages**()

Returns the sequence of images to be displayed as the icon for this window.

This method returns a copy of the internally stored list, so all operations on the returned object will not affect the window's behavior.

**Returns:**the copy of icon images' list for this window, or empty list if this window doesn't have icon images.**Since:** 1.6 **See Also:**[setIconImages(java.util.List)](http://docs.google.com/java/awt/Window.html#setIconImages(java.util.List)), [setIconImage(Image)](http://docs.google.com/java/awt/Window.html#setIconImage(java.awt.Image))

### setIconImages

public void **setIconImages**([List](http://docs.google.com/java/util/List.html)<? extends [Image](http://docs.google.com/java/awt/Image.html)> icons)

Sets the sequence of images to be displayed as the icon for this window. Subsequent calls to getIconImages will always return a copy of the icons list.

Depending on the platform capabilities one or several images of different dimensions will be used as the window's icon.

The icons list is scanned for the images of most appropriate dimensions from the beginning. If the list contains several images of the same size, the first will be used.

Ownerless windows with no icon specified use platfrom-default icon. The icon of an owned window may be inherited from the owner unless explicitly overridden. Setting the icon to null or empty list restores the default behavior.

Note : Native windowing systems may use different images of differing dimensions to represent a window, depending on the context (e.g. window decoration, window list, taskbar, etc.). They could also use just a single image for all contexts or no image at all.

**Parameters:**icons - the list of icon images to be displayed.**Since:** 1.6 **See Also:**[getIconImages()](http://docs.google.com/java/awt/Window.html#getIconImages()), [setIconImage(Image)](http://docs.google.com/java/awt/Window.html#setIconImage(java.awt.Image))

### setIconImage

public void **setIconImage**([Image](http://docs.google.com/java/awt/Image.html) image)

Sets the image to be displayed as the icon for this window.

This method can be used instead of [setIconImages()](http://docs.google.com/java/awt/Window.html#setIconImages(java.util.List)) to specify a single image as a window's icon.

The following statement:

setIconImage(image);

is equivalent to:

ArrayList imageList = new ArrayList();  
 imageList.add(image);  
 setIconImages(imageList);

Note : Native windowing systems may use different images of differing dimensions to represent a window, depending on the context (e.g. window decoration, window list, taskbar, etc.). They could also use just a single image for all contexts or no image at all.

**Parameters:**image - the icon image to be displayed.**Since:** 1.6 **See Also:**[setIconImages(java.util.List)](http://docs.google.com/java/awt/Window.html#setIconImages(java.util.List)), [getIconImages()](http://docs.google.com/java/awt/Window.html#getIconImages())

### addNotify

public void **addNotify**()

Makes this Window displayable by creating the connection to its native screen resource. This method is called internally by the toolkit and should not be called directly by programs.

**Overrides:**[addNotify](http://docs.google.com/java/awt/Container.html#addNotify()) in class [Container](http://docs.google.com/java/awt/Container.html) **Since:** JDK1.0 **See Also:**[Component.isDisplayable()](http://docs.google.com/java/awt/Component.html#isDisplayable()), [Container.removeNotify()](http://docs.google.com/java/awt/Container.html#removeNotify())

### removeNotify

public void **removeNotify**()

Makes this Container undisplayable by removing its connection to its native screen resource. Making a container undisplayable will cause all of its children to be made undisplayable. This method is called by the toolkit internally and should not be called directly by programs.

**Overrides:**[removeNotify](http://docs.google.com/java/awt/Container.html#removeNotify()) in class [Container](http://docs.google.com/java/awt/Container.html) **See Also:**[Component.isDisplayable()](http://docs.google.com/java/awt/Component.html#isDisplayable()), [Container.addNotify()](http://docs.google.com/java/awt/Container.html#addNotify())

### pack

public void **pack**()

Causes this Window to be sized to fit the preferred size and layouts of its subcomponents. If the window and/or its owner are not yet displayable, both are made displayable before calculating the preferred size. The Window will be validated after the preferredSize is calculated.

**See Also:**[Component.isDisplayable()](http://docs.google.com/java/awt/Component.html#isDisplayable())

### setMinimumSize

public void **setMinimumSize**([Dimension](http://docs.google.com/java/awt/Dimension.html) minimumSize)

Sets the minimum size of this window to a constant value. Subsequent calls to getMinimumSize will always return this value. If current window's size is less than minimumSize the size of the window is automatically enlarged to honor the minimum size.

If the setSize or setBounds methods are called afterwards with a width or height less than that specified by setMinimumSize the window is automatically enlarged to honor the minimumSize value. Setting the minimum size to null restores the default behavior.

Resizing operation may be restricted if the user tries to resize window below the minimumSize value. This behaviour is platform-dependent.

**Overrides:**[setMinimumSize](http://docs.google.com/java/awt/Component.html#setMinimumSize(java.awt.Dimension)) in class [Component](http://docs.google.com/java/awt/Component.html) **Parameters:**minimumSize - the new minimum size of this window**Since:** 1.6 **See Also:**[Component.setMinimumSize(java.awt.Dimension)](http://docs.google.com/java/awt/Component.html#setMinimumSize(java.awt.Dimension)), [Container.getMinimumSize()](http://docs.google.com/java/awt/Container.html#getMinimumSize()), [Component.isMinimumSizeSet()](http://docs.google.com/java/awt/Component.html#isMinimumSizeSet()), [setSize(Dimension)](http://docs.google.com/java/awt/Window.html#setSize(java.awt.Dimension))

### setSize

public void **setSize**([Dimension](http://docs.google.com/java/awt/Dimension.html) d)

Resizes this component so that it has width d.width and height d.height.

The d.width and d.height values are automatically enlarged if either is less than the minimum size as specified by previous call to setMinimumSize.

**Overrides:**[setSize](http://docs.google.com/java/awt/Component.html#setSize(java.awt.Dimension)) in class [Component](http://docs.google.com/java/awt/Component.html) **Parameters:**d - the dimension specifying the new size of this component**Since:** 1.6 **See Also:**[Component.getSize()](http://docs.google.com/java/awt/Component.html#getSize()), [setBounds(int, int, int, int)](http://docs.google.com/java/awt/Window.html#setBounds(int,%20int,%20int,%20int)), [setMinimumSize(java.awt.Dimension)](http://docs.google.com/java/awt/Window.html#setMinimumSize(java.awt.Dimension))

### setSize

public void **setSize**(int width,  
 int height)

Resizes this component so that it has width width and height height.

The width and height values are automatically enlarged if either is less than the minimum size as specified by previous call to setMinimumSize.

**Overrides:**[setSize](http://docs.google.com/java/awt/Component.html#setSize(int,%20int)) in class [Component](http://docs.google.com/java/awt/Component.html) **Parameters:**width - the new width of this component in pixelsheight - the new height of this component in pixels**Since:** 1.6 **See Also:**[Component.getSize()](http://docs.google.com/java/awt/Component.html#getSize()), [setBounds(int, int, int, int)](http://docs.google.com/java/awt/Window.html#setBounds(int,%20int,%20int,%20int)), [setMinimumSize(java.awt.Dimension)](http://docs.google.com/java/awt/Window.html#setMinimumSize(java.awt.Dimension))

### reshape

[@Deprecated](http://docs.google.com/java/lang/Deprecated.html)  
public void **reshape**(int x,  
 int y,  
 int width,  
 int height)

**Deprecated.** *As of JDK version 1.1, replaced by setBounds(int, int, int, int).*

**Overrides:**[reshape](http://docs.google.com/java/awt/Component.html#reshape(int,%20int,%20int,%20int)) in class [Component](http://docs.google.com/java/awt/Component.html)

### setVisible

public void **setVisible**(boolean b)

Shows or hides this Window depending on the value of parameter b.

**Overrides:**[setVisible](http://docs.google.com/java/awt/Component.html#setVisible(boolean)) in class [Component](http://docs.google.com/java/awt/Component.html) **Parameters:**b - if true, makes the Window visible, otherwise hides the Window. If the Window and/or its owner are not yet displayable, both are made displayable. The Window will be validated prior to being made visible. If the Window is already visible, this will bring the Window to the front.

If false, hides this Window, its subcomponents, and all of its owned children. The Window and its subcomponents can be made visible again with a call to #setVisible(true).

**See Also:**[Component.isDisplayable()](http://docs.google.com/java/awt/Component.html#isDisplayable()), [Component.setVisible(boolean)](http://docs.google.com/java/awt/Component.html#setVisible(boolean)), [toFront()](http://docs.google.com/java/awt/Window.html#toFront()), [dispose()](http://docs.google.com/java/awt/Window.html#dispose())

### show

[@Deprecated](http://docs.google.com/java/lang/Deprecated.html)  
public void **show**()

**Deprecated.** *As of JDK version 1.5, replaced by* [*setVisible(boolean)*](http://docs.google.com/java/awt/Window.html#setVisible(boolean))*.*

Makes the Window visible. If the Window and/or its owner are not yet displayable, both are made displayable. The Window will be validated prior to being made visible. If the Window is already visible, this will bring the Window to the front.

**Overrides:**[show](http://docs.google.com/java/awt/Component.html#show()) in class [Component](http://docs.google.com/java/awt/Component.html) **See Also:**[Component.isDisplayable()](http://docs.google.com/java/awt/Component.html#isDisplayable()), [toFront()](http://docs.google.com/java/awt/Window.html#toFront())

### hide

[@Deprecated](http://docs.google.com/java/lang/Deprecated.html)  
public void **hide**()

**Deprecated.** *As of JDK version 1.5, replaced by* [*setVisible(boolean)*](http://docs.google.com/java/awt/Window.html#setVisible(boolean))*.*

Hide this Window, its subcomponents, and all of its owned children. The Window and its subcomponents can be made visible again with a call to show.

**Overrides:**[hide](http://docs.google.com/java/awt/Component.html#hide()) in class [Component](http://docs.google.com/java/awt/Component.html) **See Also:**[show()](http://docs.google.com/java/awt/Window.html#show()), [dispose()](http://docs.google.com/java/awt/Window.html#dispose())

### dispose

public void **dispose**()

Releases all of the native screen resources used by this Window, its subcomponents, and all of its owned children. That is, the resources for these Components will be destroyed, any memory they consume will be returned to the OS, and they will be marked as undisplayable.

The Window and its subcomponents can be made displayable again by rebuilding the native resources with a subsequent call to pack or show. The states of the recreated Window and its subcomponents will be identical to the states of these objects at the point where the Window was disposed (not accounting for additional modifications between those actions).

**Note**: When the last displayable window within the Java virtual machine (VM) is disposed of, the VM may terminate. See  [AWT Threading Issues](http://docs.google.com/doc-files/AWTThreadIssues.html#Autoshutdown) for more information.

**See Also:**[Component.isDisplayable()](http://docs.google.com/java/awt/Component.html#isDisplayable()), [pack()](http://docs.google.com/java/awt/Window.html#pack()), [show()](http://docs.google.com/java/awt/Window.html#show())

### toFront

public void **toFront**()

If this Window is visible, brings this Window to the front and may make it the focused Window.

Places this Window at the top of the stacking order and shows it in front of any other Windows in this VM. No action will take place if this Window is not visible. Some platforms do not allow Windows which own other Windows to appear on top of those owned Windows. Some platforms may not permit this VM to place its Windows above windows of native applications, or Windows of other VMs. This permission may depend on whether a Window in this VM is already focused. Every attempt will be made to move this Window as high as possible in the stacking order; however, developers should not assume that this method will move this Window above all other windows in every situation.

Because of variations in native windowing systems, no guarantees about changes to the focused and active Windows can be made. Developers must never assume that this Window is the focused or active Window until this Window receives a WINDOW\_GAINED\_FOCUS or WINDOW\_ACTIVATED event. On platforms where the top-most window is the focused window, this method will **probably** focus this Window, if it is not already focused. On platforms where the stacking order does not typically affect the focused window, this method will **probably** leave the focused and active Windows unchanged.

If this method causes this Window to be focused, and this Window is a Frame or a Dialog, it will also become activated. If this Window is focused, but it is not a Frame or a Dialog, then the first Frame or Dialog that is an owner of this Window will be activated.

If this window is blocked by modal dialog, then the blocking dialog is brought to the front and remains above the blocked window.

**See Also:**[toBack()](http://docs.google.com/java/awt/Window.html#toBack())

### toBack

public void **toBack**()

If this Window is visible, sends this Window to the back and may cause it to lose focus or activation if it is the focused or active Window.

Places this Window at the bottom of the stacking order and shows it behind any other Windows in this VM. No action will take place is this Window is not visible. Some platforms do not allow Windows which are owned by other Windows to appear below their owners. Every attempt will be made to move this Window as low as possible in the stacking order; however, developers should not assume that this method will move this Window below all other windows in every situation.

Because of variations in native windowing systems, no guarantees about changes to the focused and active Windows can be made. Developers must never assume that this Window is no longer the focused or active Window until this Window receives a WINDOW\_LOST\_FOCUS or WINDOW\_DEACTIVATED event. On platforms where the top-most window is the focused window, this method will **probably** cause this Window to lose focus. In that case, the next highest, focusable Window in this VM will receive focus. On platforms where the stacking order does not typically affect the focused window, this method will **probably** leave the focused and active Windows unchanged.

**See Also:**[toFront()](http://docs.google.com/java/awt/Window.html#toFront())

### getToolkit

public [Toolkit](http://docs.google.com/java/awt/Toolkit.html) **getToolkit**()

Returns the toolkit of this frame.

**Overrides:**[getToolkit](http://docs.google.com/java/awt/Component.html#getToolkit()) in class [Component](http://docs.google.com/java/awt/Component.html) **Returns:**the toolkit of this window.**See Also:**[Toolkit](http://docs.google.com/java/awt/Toolkit.html), [Toolkit.getDefaultToolkit()](http://docs.google.com/java/awt/Toolkit.html#getDefaultToolkit()), [Component.getToolkit()](http://docs.google.com/java/awt/Component.html#getToolkit())

### getWarningString

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

Gets the warning string that is displayed with this window. If this window is insecure, the warning string is displayed somewhere in the visible area of the window. A window is insecure if there is a security manager, and the security manager's checkTopLevelWindow method returns false when this window is passed to it as an argument.

If the window is secure, then getWarningString returns null. If the window is insecure, this method checks for the system property awt.appletWarning and returns the string value of that property.

**Returns:**the warning string for this window.**See Also:**[SecurityManager.checkTopLevelWindow(java.lang.Object)](http://docs.google.com/java/lang/SecurityManager.html#checkTopLevelWindow(java.lang.Object))

### getLocale

public [Locale](http://docs.google.com/java/util/Locale.html) **getLocale**()

Gets the Locale object that is associated with this window, if the locale has been set. If no locale has been set, then the default locale is returned.

**Overrides:**[getLocale](http://docs.google.com/java/awt/Component.html#getLocale()) in class [Component](http://docs.google.com/java/awt/Component.html) **Returns:**the locale that is set for this window.**Since:** JDK1.1 **See Also:**[Locale](http://docs.google.com/java/util/Locale.html)

### getInputContext

public [InputContext](http://docs.google.com/java/awt/im/InputContext.html) **getInputContext**()

Gets the input context for this window. A window always has an input context, which is shared by subcomponents unless they create and set their own.

**Overrides:**[getInputContext](http://docs.google.com/java/awt/Component.html#getInputContext()) in class [Component](http://docs.google.com/java/awt/Component.html) **Returns:**the input context used by this component; null if no context can be determined**Since:** 1.2 **See Also:**[Component.getInputContext()](http://docs.google.com/java/awt/Component.html#getInputContext())

### setCursor

public void **setCursor**([Cursor](http://docs.google.com/java/awt/Cursor.html) cursor)

Set the cursor image to a specified cursor.

The method may have no visual effect if the Java platform implementation and/or the native system do not support changing the mouse cursor shape.

**Overrides:**[setCursor](http://docs.google.com/java/awt/Component.html#setCursor(java.awt.Cursor)) in class [Component](http://docs.google.com/java/awt/Component.html) **Parameters:**cursor - One of the constants defined by the Cursor class. If this parameter is null then the cursor for this window will be set to the type Cursor.DEFAULT\_CURSOR.**Since:** JDK1.1 **See Also:**[Component.getCursor()](http://docs.google.com/java/awt/Component.html#getCursor()), [Cursor](http://docs.google.com/java/awt/Cursor.html)

### getOwner

public [Window](http://docs.google.com/java/awt/Window.html) **getOwner**()

Returns the owner of this window.

**Since:** 1.2

### getOwnedWindows

public [Window](http://docs.google.com/java/awt/Window.html)[] **getOwnedWindows**()

Return an array containing all the windows this window currently owns.

**Since:** 1.2

### getWindows

public static [Window](http://docs.google.com/java/awt/Window.html)[] **getWindows**()

Returns an array of all Windows, both owned and ownerless, created by this application. If called from an applet, the array includes only the Windows accessible by that applet.

**Warning:** this method may return system created windows, such as a print dialog. Applications should not assume the existence of these dialogs, nor should an application assume anything about these dialogs such as component positions, LayoutManagers or serialization.

**Since:** 1.6 **See Also:**[Frame.getFrames()](http://docs.google.com/java/awt/Frame.html#getFrames()), [getOwnerlessWindows()](http://docs.google.com/java/awt/Window.html#getOwnerlessWindows())

### getOwnerlessWindows

public static [Window](http://docs.google.com/java/awt/Window.html)[] **getOwnerlessWindows**()

Returns an array of all Windows created by this application that have no owner. They include Frames and ownerless Dialogs and Windows. If called from an applet, the array includes only the Windows accessible by that applet.

**Warning:** this method may return system created windows, such as a print dialog. Applications should not assume the existence of these dialogs, nor should an application assume anything about these dialogs such as component positions, LayoutManagers or serialization.

**Since:** 1.6 **See Also:**[Frame.getFrames()](http://docs.google.com/java/awt/Frame.html#getFrames()), [getWindows(sun.awt.AppContext)](http://docs.google.com/java/awt/Window.html#getWindows(sun.awt.AppContext))

### setModalExclusionType

public void **setModalExclusionType**([Dialog.ModalExclusionType](http://docs.google.com/java/awt/Dialog.ModalExclusionType.html) exclusionType)

Specifies the modal exclusion type for this window. If a window is modal excluded, it is not blocked by some modal dialogs. See [Dialog.ModalExclusionType](http://docs.google.com/java/awt/Dialog.ModalExclusionType.html) for possible modal exclusion types.

If the given type is not supported, NO\_EXCLUDE is used.

Note: changing the modal exclusion type for a visible window may have no effect until it is hidden and then shown again.

**Parameters:**exclusionType - the modal exclusion type for this window; a null value is equivivalent to [NO\_EXCLUDE](http://docs.google.com/java/awt/Dialog.ModalExclusionType.html#NO_EXCLUDE) **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if the calling thread does not have permission to set the modal exclusion property to the window with the given exclusionType**Since:** 1.6 **See Also:**[Dialog.ModalExclusionType](http://docs.google.com/java/awt/Dialog.ModalExclusionType.html), [getModalExclusionType()](http://docs.google.com/java/awt/Window.html#getModalExclusionType()), [Toolkit.isModalExclusionTypeSupported(java.awt.Dialog.ModalExclusionType)](http://docs.google.com/java/awt/Toolkit.html#isModalExclusionTypeSupported(java.awt.Dialog.ModalExclusionType))

### getModalExclusionType

public [Dialog.ModalExclusionType](http://docs.google.com/java/awt/Dialog.ModalExclusionType.html) **getModalExclusionType**()

Returns the modal exclusion type of this window.

**Returns:**the modal exclusion type of this window**Since:** 1.6 **See Also:**[Dialog.ModalExclusionType](http://docs.google.com/java/awt/Dialog.ModalExclusionType.html), [setModalExclusionType(java.awt.Dialog.ModalExclusionType)](http://docs.google.com/java/awt/Window.html#setModalExclusionType(java.awt.Dialog.ModalExclusionType))

### addWindowListener

public void **addWindowListener**([WindowListener](http://docs.google.com/java/awt/event/WindowListener.html) l)

Adds the specified window listener to receive window events from this window. If l is null, no exception is thrown and no action is performed.

Refer to [AWT Threading Issues](http://docs.google.com/doc-files/AWTThreadIssues.html#ListenersThreads) for details on AWT's threading model.

**Parameters:**l - the window listener**See Also:**[removeWindowListener(java.awt.event.WindowListener)](http://docs.google.com/java/awt/Window.html#removeWindowListener(java.awt.event.WindowListener)), [getWindowListeners()](http://docs.google.com/java/awt/Window.html#getWindowListeners())

### addWindowStateListener

public void **addWindowStateListener**([WindowStateListener](http://docs.google.com/java/awt/event/WindowStateListener.html) l)

Adds the specified window state listener to receive window events from this window. If l is null, no exception is thrown and no action is performed.

Refer to [AWT Threading Issues](http://docs.google.com/doc-files/AWTThreadIssues.html#ListenersThreads) for details on AWT's threading model.

**Parameters:**l - the window state listener**Since:** 1.4 **See Also:**[removeWindowStateListener(java.awt.event.WindowStateListener)](http://docs.google.com/java/awt/Window.html#removeWindowStateListener(java.awt.event.WindowStateListener)), [getWindowStateListeners()](http://docs.google.com/java/awt/Window.html#getWindowStateListeners())

### addWindowFocusListener

public void **addWindowFocusListener**([WindowFocusListener](http://docs.google.com/java/awt/event/WindowFocusListener.html) l)

Adds the specified window focus listener to receive window events from this window. If l is null, no exception is thrown and no action is performed.

Refer to [AWT Threading Issues](http://docs.google.com/doc-files/AWTThreadIssues.html#ListenersThreads) for details on AWT's threading model.

**Parameters:**l - the window focus listener**Since:** 1.4 **See Also:**[removeWindowFocusListener(java.awt.event.WindowFocusListener)](http://docs.google.com/java/awt/Window.html#removeWindowFocusListener(java.awt.event.WindowFocusListener)), [getWindowFocusListeners()](http://docs.google.com/java/awt/Window.html#getWindowFocusListeners())

### removeWindowListener

public void **removeWindowListener**([WindowListener](http://docs.google.com/java/awt/event/WindowListener.html) l)

Removes the specified window listener so that it no longer receives window events from this window. If l is null, no exception is thrown and no action is performed.

Refer to [AWT Threading Issues](http://docs.google.com/doc-files/AWTThreadIssues.html#ListenersThreads) for details on AWT's threading model.

**Parameters:**l - the window listener**See Also:**[addWindowListener(java.awt.event.WindowListener)](http://docs.google.com/java/awt/Window.html#addWindowListener(java.awt.event.WindowListener)), [getWindowListeners()](http://docs.google.com/java/awt/Window.html#getWindowListeners())

### removeWindowStateListener

public void **removeWindowStateListener**([WindowStateListener](http://docs.google.com/java/awt/event/WindowStateListener.html) l)

Removes the specified window state listener so that it no longer receives window events from this window. If l is null, no exception is thrown and no action is performed.

Refer to [AWT Threading Issues](http://docs.google.com/doc-files/AWTThreadIssues.html#ListenersThreads) for details on AWT's threading model.

**Parameters:**l - the window state listener**Since:** 1.4 **See Also:**[addWindowStateListener(java.awt.event.WindowStateListener)](http://docs.google.com/java/awt/Window.html#addWindowStateListener(java.awt.event.WindowStateListener)), [getWindowStateListeners()](http://docs.google.com/java/awt/Window.html#getWindowStateListeners())

### removeWindowFocusListener

public void **removeWindowFocusListener**([WindowFocusListener](http://docs.google.com/java/awt/event/WindowFocusListener.html) l)

Removes the specified window focus listener so that it no longer receives window events from this window. If l is null, no exception is thrown and no action is performed.

Refer to [AWT Threading Issues](http://docs.google.com/doc-files/AWTThreadIssues.html#ListenersThreads) for details on AWT's threading model.

**Parameters:**l - the window focus listener**Since:** 1.4 **See Also:**[addWindowFocusListener(java.awt.event.WindowFocusListener)](http://docs.google.com/java/awt/Window.html#addWindowFocusListener(java.awt.event.WindowFocusListener)), [getWindowFocusListeners()](http://docs.google.com/java/awt/Window.html#getWindowFocusListeners())

### getWindowListeners

public [WindowListener](http://docs.google.com/java/awt/event/WindowListener.html)[] **getWindowListeners**()

Returns an array of all the window listeners registered on this window.

**Returns:**all of this window's WindowListeners or an empty array if no window listeners are currently registered**Since:** 1.4 **See Also:**[addWindowListener(java.awt.event.WindowListener)](http://docs.google.com/java/awt/Window.html#addWindowListener(java.awt.event.WindowListener)), [removeWindowListener(java.awt.event.WindowListener)](http://docs.google.com/java/awt/Window.html#removeWindowListener(java.awt.event.WindowListener))

### getWindowFocusListeners

public [WindowFocusListener](http://docs.google.com/java/awt/event/WindowFocusListener.html)[] **getWindowFocusListeners**()

Returns an array of all the window focus listeners registered on this window.

**Returns:**all of this window's WindowFocusListeners or an empty array if no window focus listeners are currently registered**Since:** 1.4 **See Also:**[addWindowFocusListener(java.awt.event.WindowFocusListener)](http://docs.google.com/java/awt/Window.html#addWindowFocusListener(java.awt.event.WindowFocusListener)), [removeWindowFocusListener(java.awt.event.WindowFocusListener)](http://docs.google.com/java/awt/Window.html#removeWindowFocusListener(java.awt.event.WindowFocusListener))

### getWindowStateListeners

public [WindowStateListener](http://docs.google.com/java/awt/event/WindowStateListener.html)[] **getWindowStateListeners**()

Returns an array of all the window state listeners registered on this window.

**Returns:**all of this window's WindowStateListeners or an empty array if no window state listeners are currently registered**Since:** 1.4 **See Also:**[addWindowStateListener(java.awt.event.WindowStateListener)](http://docs.google.com/java/awt/Window.html#addWindowStateListener(java.awt.event.WindowStateListener)), [removeWindowStateListener(java.awt.event.WindowStateListener)](http://docs.google.com/java/awt/Window.html#removeWindowStateListener(java.awt.event.WindowStateListener))

### getListeners

public <T extends [EventListener](http://docs.google.com/java/util/EventListener.html)> T[] **getListeners**([Class](http://docs.google.com/java/lang/Class.html)<T> listenerType)

Returns an array of all the objects currently registered as *Foo*Listeners upon this Window. *Foo*Listeners are registered using the add*Foo*Listener method.

You can specify the listenerType argument with a class literal, such as *Foo*Listener.class. For example, you can query a Window w for its window listeners with the following code:

WindowListener[] wls = (WindowListener[])(w.getListeners(WindowListener.class));

If no such listeners exist, this method returns an empty array.

**Overrides:**[getListeners](http://docs.google.com/java/awt/Container.html#getListeners(java.lang.Class)) in class [Container](http://docs.google.com/java/awt/Container.html) **Parameters:**listenerType - the type of listeners requested; this parameter should specify an interface that descends from java.util.EventListener **Returns:**an array of all objects registered as *Foo*Listeners on this window, or an empty array if no such listeners have been added **Throws:** [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if listenerType doesn't specify a class or interface that implements java.util.EventListener**Since:** 1.3 **See Also:**[getWindowListeners()](http://docs.google.com/java/awt/Window.html#getWindowListeners())

### processEvent

protected void **processEvent**([AWTEvent](http://docs.google.com/java/awt/AWTEvent.html) e)

Processes events on this window. If the event is an WindowEvent, it invokes the processWindowEvent method, else it invokes its superclass's processEvent.

Note that if the event parameter is null the behavior is unspecified and may result in an exception.

**Overrides:**[processEvent](http://docs.google.com/java/awt/Container.html#processEvent(java.awt.AWTEvent)) in class [Container](http://docs.google.com/java/awt/Container.html) **Parameters:**e - the event**See Also:**[Component.processComponentEvent(java.awt.event.ComponentEvent)](http://docs.google.com/java/awt/Component.html#processComponentEvent(java.awt.event.ComponentEvent)), [Component.processFocusEvent(java.awt.event.FocusEvent)](http://docs.google.com/java/awt/Component.html#processFocusEvent(java.awt.event.FocusEvent)), [Component.processKeyEvent(java.awt.event.KeyEvent)](http://docs.google.com/java/awt/Component.html#processKeyEvent(java.awt.event.KeyEvent)), [Component.processMouseEvent(java.awt.event.MouseEvent)](http://docs.google.com/java/awt/Component.html#processMouseEvent(java.awt.event.MouseEvent)), [Component.processMouseMotionEvent(java.awt.event.MouseEvent)](http://docs.google.com/java/awt/Component.html#processMouseMotionEvent(java.awt.event.MouseEvent)), [Component.processInputMethodEvent(java.awt.event.InputMethodEvent)](http://docs.google.com/java/awt/Component.html#processInputMethodEvent(java.awt.event.InputMethodEvent)), [Component.processHierarchyEvent(java.awt.event.HierarchyEvent)](http://docs.google.com/java/awt/Component.html#processHierarchyEvent(java.awt.event.HierarchyEvent)), [Component.processMouseWheelEvent(java.awt.event.MouseWheelEvent)](http://docs.google.com/java/awt/Component.html#processMouseWheelEvent(java.awt.event.MouseWheelEvent))

### processWindowEvent

protected void **processWindowEvent**([WindowEvent](http://docs.google.com/java/awt/event/WindowEvent.html) e)

Processes window events occurring on this window by dispatching them to any registered WindowListener objects. NOTE: This method will not be called unless window events are enabled for this component; this happens when one of the following occurs:

* A WindowListener object is registered via addWindowListener
* Window events are enabled via enableEvents

Note that if the event parameter is null the behavior is unspecified and may result in an exception.

**Parameters:**e - the window event**See Also:**[Component.enableEvents(long)](http://docs.google.com/java/awt/Component.html#enableEvents(long))

### processWindowFocusEvent

protected void **processWindowFocusEvent**([WindowEvent](http://docs.google.com/java/awt/event/WindowEvent.html) e)

Processes window focus event occuring on this window by dispatching them to any registered WindowFocusListener objects. NOTE: this method will not be called unless window focus events are enabled for this window. This happens when one of the following occurs:

* a WindowFocusListener is registered via addWindowFocusListener
* Window focus events are enabled via enableEvents

Note that if the event parameter is null the behavior is unspecified and may result in an exception.

**Parameters:**e - the window focus event**Since:** 1.4 **See Also:**[Component.enableEvents(long)](http://docs.google.com/java/awt/Component.html#enableEvents(long))

### processWindowStateEvent

protected void **processWindowStateEvent**([WindowEvent](http://docs.google.com/java/awt/event/WindowEvent.html) e)

Processes window state event occuring on this window by dispatching them to any registered WindowStateListener objects. NOTE: this method will not be called unless window state events are enabled for this window. This happens when one of the following occurs:

* a WindowStateListener is registered via addWindowStateListener
* window state events are enabled via enableEvents

Note that if the event parameter is null the behavior is unspecified and may result in an exception.

**Parameters:**e - the window state event**Since:** 1.4 **See Also:**[Component.enableEvents(long)](http://docs.google.com/java/awt/Component.html#enableEvents(long))

### setAlwaysOnTop

public final void **setAlwaysOnTop**(boolean alwaysOnTop)  
 throws [SecurityException](http://docs.google.com/java/lang/SecurityException.html)

Sets whether this window should always be above other windows. If there are multiple always-on-top windows, their relative order is unspecified and platform dependent.

If some other window is already always-on-top then the relative order between these windows is unspecified (depends on platform). No window can be brought to be over the always-on-top window except maybe another always-on-top window.

All windows owned by an always-on-top window inherit this state and automatically become always-on-top. If a window ceases to be always-on-top, the windows that it owns will no longer be always-on-top. When an always-on-top window is sent [toBack](http://docs.google.com/java/awt/Window.html#toBack()), its always-on-top state is set to false.

When this method is called on a window with a value of true, and the window is visible and the platform supports always-on-top for this window, the window is immediately brought forward, "sticking" it in the top-most position. If the window isn`t currently visible, this method sets the always-on-top state to true but does not bring the window forward. When the window is later shown, it will be always-on-top.

When this method is called on a window with a value of false the always-on-top state is set to normal. The window remains in the top-most position but it`s z-order can be changed as for any other window. Calling this method with a value of false on a window that has a normal state has no effect. Setting the always-on-top state to false has no effect on the relative z-order of the windows if there are no other always-on-top windows.

**Note**: some platforms might not support always-on-top windows. To detect if always-on-top windows are supported by the current platform, use [Toolkit.isAlwaysOnTopSupported()](http://docs.google.com/java/awt/Toolkit.html#isAlwaysOnTopSupported()) and [isAlwaysOnTopSupported()](http://docs.google.com/java/awt/Window.html#isAlwaysOnTopSupported()). If always-on-top mode isn't supported by the toolkit or for this window, calling this method has no effect.

If a SecurityManager is installed, the calling thread must be granted the AWTPermission "setWindowAlwaysOnTop" in order to set the value of this property. If this permission is not granted, this method will throw a SecurityException, and the current value of the property will be left unchanged.

**Parameters:**alwaysOnTop - true if the window should always be above other windows **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if the calling thread does not have permission to set the value of always-on-top property**Since:** 1.5 **See Also:**[isAlwaysOnTop()](http://docs.google.com/java/awt/Window.html#isAlwaysOnTop()), [toFront()](http://docs.google.com/java/awt/Window.html#toFront()), [toBack()](http://docs.google.com/java/awt/Window.html#toBack()), [AWTPermission](http://docs.google.com/java/awt/AWTPermission.html), [isAlwaysOnTopSupported()](http://docs.google.com/java/awt/Window.html#isAlwaysOnTopSupported()), [Toolkit.isAlwaysOnTopSupported()](http://docs.google.com/java/awt/Toolkit.html#isAlwaysOnTopSupported())

### isAlwaysOnTopSupported

public boolean **isAlwaysOnTopSupported**()

Returns whether the always-on-top mode is supported for this window. Some platforms may not support always-on-top windows, some may support only some kinds of top-level windows; for example, a platform may not support always-on-top modal dialogs.

**Returns:**true, if the always-on-top mode is supported by the toolkit and for this window, false, if always-on-top mode is not supported for this window or toolkit doesn't support always-on-top windows.**Since:** 1.6 **See Also:**[setAlwaysOnTop(boolean)](http://docs.google.com/java/awt/Window.html#setAlwaysOnTop(boolean)), [Toolkit.isAlwaysOnTopSupported()](http://docs.google.com/java/awt/Toolkit.html#isAlwaysOnTopSupported())

### isAlwaysOnTop

public final boolean **isAlwaysOnTop**()

Returns whether this window is an always-on-top window.

**Returns:**true, if the window is in always-on-top state, false otherwise**Since:** 1.5 **See Also:**[setAlwaysOnTop(boolean)](http://docs.google.com/java/awt/Window.html#setAlwaysOnTop(boolean))

### getFocusOwner

public [Component](http://docs.google.com/java/awt/Component.html) **getFocusOwner**()

Returns the child Component of this Window that has focus if this Window is focused; returns null otherwise.

**Returns:**the child Component with focus, or null if this Window is not focused**See Also:**[getMostRecentFocusOwner()](http://docs.google.com/java/awt/Window.html#getMostRecentFocusOwner()), [isFocused()](http://docs.google.com/java/awt/Window.html#isFocused())

### getMostRecentFocusOwner

public [Component](http://docs.google.com/java/awt/Component.html) **getMostRecentFocusOwner**()

Returns the child Component of this Window that will receive the focus when this Window is focused. If this Window is currently focused, this method returns the same Component as getFocusOwner(). If this Window is not focused, then the child Component that most recently requested focus will be returned. If no child Component has ever requested focus, and this is a focusable Window, then this Window's initial focusable Component is returned. If no child Component has ever requested focus, and this is a non-focusable Window, null is returned.

**Returns:**the child Component that will receive focus when this Window is focused**Since:** 1.4 **See Also:**[getFocusOwner()](http://docs.google.com/java/awt/Window.html#getFocusOwner()), [isFocused()](http://docs.google.com/java/awt/Window.html#isFocused()), [isFocusableWindow()](http://docs.google.com/java/awt/Window.html#isFocusableWindow())

### isActive

public boolean **isActive**()

Returns whether this Window is active. Only a Frame or a Dialog may be active. The native windowing system may denote the active Window or its children with special decorations, such as a highlighted title bar. The active Window is always either the focused Window, or the first Frame or Dialog that is an owner of the focused Window.

**Returns:**whether this is the active Window.**Since:** 1.4 **See Also:**[isFocused()](http://docs.google.com/java/awt/Window.html#isFocused())

### isFocused

public boolean **isFocused**()

Returns whether this Window is focused. If there exists a focus owner, the focused Window is the Window that is, or contains, that focus owner. If there is no focus owner, then no Window is focused.

If the focused Window is a Frame or a Dialog it is also the active Window. Otherwise, the active Window is the first Frame or Dialog that is an owner of the focused Window.

**Returns:**whether this is the focused Window.**Since:** 1.4 **See Also:**[isActive()](http://docs.google.com/java/awt/Window.html#isActive())

### getFocusTraversalKeys

public [Set](http://docs.google.com/java/util/Set.html)<[AWTKeyStroke](http://docs.google.com/java/awt/AWTKeyStroke.html)> **getFocusTraversalKeys**(int id)

Gets a focus traversal key for this Window. (See setFocusTraversalKeys for a full description of each key.)

If the traversal key has not been explicitly set for this Window, then this Window's parent's traversal key is returned. If the traversal key has not been explicitly set for any of this Window's ancestors, then the current KeyboardFocusManager's default traversal key is returned.

**Overrides:**[getFocusTraversalKeys](http://docs.google.com/java/awt/Container.html#getFocusTraversalKeys(int)) in class [Container](http://docs.google.com/java/awt/Container.html) **Parameters:**id - one of KeyboardFocusManager.FORWARD\_TRAVERSAL\_KEYS, KeyboardFocusManager.BACKWARD\_TRAVERSAL\_KEYS, KeyboardFocusManager.UP\_CYCLE\_TRAVERSAL\_KEYS, or KeyboardFocusManager.DOWN\_CYCLE\_TRAVERSAL\_KEYS **Returns:**the AWTKeyStroke for the specified key **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if id is not one of KeyboardFocusManager.FORWARD\_TRAVERSAL\_KEYS, KeyboardFocusManager.BACKWARD\_TRAVERSAL\_KEYS, KeyboardFocusManager.UP\_CYCLE\_TRAVERSAL\_KEYS, or KeyboardFocusManager.DOWN\_CYCLE\_TRAVERSAL\_KEYS**Since:** 1.4 **See Also:**[Container.setFocusTraversalKeys(int, java.util.Set)](http://docs.google.com/java/awt/Container.html#setFocusTraversalKeys(int,%20java.util.Set)), [KeyboardFocusManager.FORWARD\_TRAVERSAL\_KEYS](http://docs.google.com/java/awt/KeyboardFocusManager.html#FORWARD_TRAVERSAL_KEYS), [KeyboardFocusManager.BACKWARD\_TRAVERSAL\_KEYS](http://docs.google.com/java/awt/KeyboardFocusManager.html#BACKWARD_TRAVERSAL_KEYS), [KeyboardFocusManager.UP\_CYCLE\_TRAVERSAL\_KEYS](http://docs.google.com/java/awt/KeyboardFocusManager.html#UP_CYCLE_TRAVERSAL_KEYS), [KeyboardFocusManager.DOWN\_CYCLE\_TRAVERSAL\_KEYS](http://docs.google.com/java/awt/KeyboardFocusManager.html#DOWN_CYCLE_TRAVERSAL_KEYS)

### setFocusCycleRoot

public final void **setFocusCycleRoot**(boolean focusCycleRoot)

Does nothing because Windows must always be roots of a focus traversal cycle. The passed-in value is ignored.

**Overrides:**[setFocusCycleRoot](http://docs.google.com/java/awt/Container.html#setFocusCycleRoot(boolean)) in class [Container](http://docs.google.com/java/awt/Container.html) **Parameters:**focusCycleRoot - this value is ignored**Since:** 1.4 **See Also:**[isFocusCycleRoot()](http://docs.google.com/java/awt/Window.html#isFocusCycleRoot()), [Container.setFocusTraversalPolicy(java.awt.FocusTraversalPolicy)](http://docs.google.com/java/awt/Container.html#setFocusTraversalPolicy(java.awt.FocusTraversalPolicy)), [Container.getFocusTraversalPolicy()](http://docs.google.com/java/awt/Container.html#getFocusTraversalPolicy())

### isFocusCycleRoot

public final boolean **isFocusCycleRoot**()

Always returns true because all Windows must be roots of a focus traversal cycle.

**Overrides:**[isFocusCycleRoot](http://docs.google.com/java/awt/Container.html#isFocusCycleRoot()) in class [Container](http://docs.google.com/java/awt/Container.html) **Returns:**true**Since:** 1.4 **See Also:**[setFocusCycleRoot(boolean)](http://docs.google.com/java/awt/Window.html#setFocusCycleRoot(boolean)), [Container.setFocusTraversalPolicy(java.awt.FocusTraversalPolicy)](http://docs.google.com/java/awt/Container.html#setFocusTraversalPolicy(java.awt.FocusTraversalPolicy)), [Container.getFocusTraversalPolicy()](http://docs.google.com/java/awt/Container.html#getFocusTraversalPolicy())

### getFocusCycleRootAncestor

public final [Container](http://docs.google.com/java/awt/Container.html) **getFocusCycleRootAncestor**()

Always returns null because Windows have no ancestors; they represent the top of the Component hierarchy.

**Overrides:**[getFocusCycleRootAncestor](http://docs.google.com/java/awt/Component.html#getFocusCycleRootAncestor()) in class [Component](http://docs.google.com/java/awt/Component.html) **Returns:**null**Since:** 1.4 **See Also:**[Container.isFocusCycleRoot()](http://docs.google.com/java/awt/Container.html#isFocusCycleRoot())

### isFocusableWindow

public final boolean **isFocusableWindow**()

Returns whether this Window can become the focused Window, that is, whether this Window or any of its subcomponents can become the focus owner. For a Frame or Dialog to be focusable, its focusable Window state must be set to true. For a Window which is not a Frame or Dialog to be focusable, its focusable Window state must be set to true, its nearest owning Frame or Dialog must be showing on the screen, and it must contain at least one Component in its focus traversal cycle. If any of these conditions is not met, then neither this Window nor any of its subcomponents can become the focus owner.

**Returns:**true if this Window can be the focused Window; false otherwise**Since:** 1.4 **See Also:**[getFocusableWindowState()](http://docs.google.com/java/awt/Window.html#getFocusableWindowState()), [setFocusableWindowState(boolean)](http://docs.google.com/java/awt/Window.html#setFocusableWindowState(boolean)), [isShowing()](http://docs.google.com/java/awt/Window.html#isShowing()), [Component.isFocusable()](http://docs.google.com/java/awt/Component.html#isFocusable())

### getFocusableWindowState

public boolean **getFocusableWindowState**()

Returns whether this Window can become the focused Window if it meets the other requirements outlined in isFocusableWindow. If this method returns false, then isFocusableWindow will return false as well. If this method returns true, then isFocusableWindow may return true or false depending upon the other requirements which must be met in order for a Window to be focusable.

By default, all Windows have a focusable Window state of true.

**Returns:**whether this Window can be the focused Window**Since:** 1.4 **See Also:**[isFocusableWindow()](http://docs.google.com/java/awt/Window.html#isFocusableWindow()), [setFocusableWindowState(boolean)](http://docs.google.com/java/awt/Window.html#setFocusableWindowState(boolean)), [isShowing()](http://docs.google.com/java/awt/Window.html#isShowing()), [Component.setFocusable(boolean)](http://docs.google.com/java/awt/Component.html#setFocusable(boolean))

### setFocusableWindowState

public void **setFocusableWindowState**(boolean focusableWindowState)

Sets whether this Window can become the focused Window if it meets the other requirements outlined in isFocusableWindow. If this Window's focusable Window state is set to false, then isFocusableWindow will return false. If this Window's focusable Window state is set to true, then isFocusableWindow may return true or false depending upon the other requirements which must be met in order for a Window to be focusable.

Setting a Window's focusability state to false is the standard mechanism for an application to identify to the AWT a Window which will be used as a floating palette or toolbar, and thus should be a non-focusable Window. Setting the focusability state on a visible Window can have a delayed effect on some platforms — the actual change may happen only when the Window becomes hidden and then visible again. To ensure consistent behavior across platforms, set the Window's focusable state when the Window is invisible and then show it.

**Parameters:**focusableWindowState - whether this Window can be the focused Window**Since:** 1.4 **See Also:**[isFocusableWindow()](http://docs.google.com/java/awt/Window.html#isFocusableWindow()), [getFocusableWindowState()](http://docs.google.com/java/awt/Window.html#getFocusableWindowState()), [isShowing()](http://docs.google.com/java/awt/Window.html#isShowing()), [Component.setFocusable(boolean)](http://docs.google.com/java/awt/Component.html#setFocusable(boolean))

### addPropertyChangeListener

public 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 bound properties of this class, including the following:

* this Window's font ("font")
* this Window's background color ("background")
* this Window's foreground color ("foreground")
* this Window's focusability ("focusable")
* this Window's focus traversal keys enabled state ("focusTraversalKeysEnabled")
* this Window's Set of FORWARD\_TRAVERSAL\_KEYS ("forwardFocusTraversalKeys")
* this Window's Set of BACKWARD\_TRAVERSAL\_KEYS ("backwardFocusTraversalKeys")
* this Window's Set of UP\_CYCLE\_TRAVERSAL\_KEYS ("upCycleFocusTraversalKeys")
* this Window's Set of DOWN\_CYCLE\_TRAVERSAL\_KEYS ("downCycleFocusTraversalKeys")
* this Window's focus traversal policy ("focusTraversalPolicy")
* this Window's focusable Window state ("focusableWindowState")
* this Window's always-on-top state("alwaysOnTop")

Note that if this Window is inheriting a bound property, then no event will be fired in response to a change in the inherited property.

If listener is null, no exception is thrown and no action is performed.

**Overrides:**[addPropertyChangeListener](http://docs.google.com/java/awt/Container.html#addPropertyChangeListener(java.beans.PropertyChangeListener)) in class [Container](http://docs.google.com/java/awt/Container.html) **Parameters:**listener - the PropertyChangeListener to be added**See Also:**[Component.removePropertyChangeListener(java.beans.PropertyChangeListener)](http://docs.google.com/java/awt/Component.html#removePropertyChangeListener(java.beans.PropertyChangeListener)), [addPropertyChangeListener(java.lang.String,java.beans.PropertyChangeListener)](http://docs.google.com/java/awt/Window.html#addPropertyChangeListener(java.lang.String,%20java.beans.PropertyChangeListener))

### addPropertyChangeListener

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

Adds a PropertyChangeListener to the listener list for a specific property. The specified property may be user-defined, or one of the following:

* this Window's font ("font")
* this Window's background color ("background")
* this Window's foreground color ("foreground")
* this Window's focusability ("focusable")
* this Window's focus traversal keys enabled state ("focusTraversalKeysEnabled")
* this Window's Set of FORWARD\_TRAVERSAL\_KEYS ("forwardFocusTraversalKeys")
* this Window's Set of BACKWARD\_TRAVERSAL\_KEYS ("backwardFocusTraversalKeys")
* this Window's Set of UP\_CYCLE\_TRAVERSAL\_KEYS ("upCycleFocusTraversalKeys")
* this Window's Set of DOWN\_CYCLE\_TRAVERSAL\_KEYS ("downCycleFocusTraversalKeys")
* this Window's focus traversal policy ("focusTraversalPolicy")
* this Window's focusable Window state ("focusableWindowState")
* this Window's always-on-top state("alwaysOnTop")

Note that if this Window is inheriting a bound property, then no event will be fired in response to a change in the inherited property.

If listener is null, no exception is thrown and no action is performed.

**Overrides:**[addPropertyChangeListener](http://docs.google.com/java/awt/Container.html#addPropertyChangeListener(java.lang.String,%20java.beans.PropertyChangeListener)) in class [Container](http://docs.google.com/java/awt/Container.html) **Parameters:**propertyName - one of the property names listed abovelistener - the PropertyChangeListener to be added**See Also:**[addPropertyChangeListener(java.beans.PropertyChangeListener)](http://docs.google.com/java/awt/Window.html#addPropertyChangeListener(java.beans.PropertyChangeListener)), [Component.removePropertyChangeListener(java.beans.PropertyChangeListener)](http://docs.google.com/java/awt/Component.html#removePropertyChangeListener(java.beans.PropertyChangeListener))

### postEvent

[@Deprecated](http://docs.google.com/java/lang/Deprecated.html)  
public boolean **postEvent**([Event](http://docs.google.com/java/awt/Event.html) e)

**Deprecated.** *As of JDK version 1.1 replaced by dispatchEvent(AWTEvent).*

**Specified by:**[postEvent](http://docs.google.com/java/awt/MenuContainer.html#postEvent(java.awt.Event)) in interface [MenuContainer](http://docs.google.com/java/awt/MenuContainer.html)**Overrides:**[postEvent](http://docs.google.com/java/awt/Component.html#postEvent(java.awt.Event)) in class [Component](http://docs.google.com/java/awt/Component.html)

### isShowing

public boolean **isShowing**()

Checks if this Window is showing on screen.

**Overrides:**[isShowing](http://docs.google.com/java/awt/Component.html#isShowing()) in class [Component](http://docs.google.com/java/awt/Component.html) **Returns:**true if the component is showing, false otherwise**See Also:**[Component.setVisible(boolean)](http://docs.google.com/java/awt/Component.html#setVisible(boolean))

### applyResourceBundle

[@Deprecated](http://docs.google.com/java/lang/Deprecated.html)  
public void **applyResourceBundle**([ResourceBundle](http://docs.google.com/java/util/ResourceBundle.html) rb)

**Deprecated.** *As of J2SE 1.4, replaced by* [*Component.applyComponentOrientation*](http://docs.google.com/java/awt/Component.html#applyComponentOrientation(java.awt.ComponentOrientation))*.*

### applyResourceBundle

[@Deprecated](http://docs.google.com/java/lang/Deprecated.html)  
public void **applyResourceBundle**([String](http://docs.google.com/java/lang/String.html) rbName)

**Deprecated.** *As of J2SE 1.4, replaced by* [*Component.applyComponentOrientation*](http://docs.google.com/java/awt/Component.html#applyComponentOrientation(java.awt.ComponentOrientation))*.*

### getAccessibleContext

public [AccessibleContext](http://docs.google.com/javax/accessibility/AccessibleContext.html) **getAccessibleContext**()

Gets the AccessibleContext associated with this Window. For windows, the AccessibleContext takes the form of an AccessibleAWTWindow. A new AccessibleAWTWindow instance is created if necessary.

**Specified by:**[getAccessibleContext](http://docs.google.com/javax/accessibility/Accessible.html#getAccessibleContext()) in interface [Accessible](http://docs.google.com/javax/accessibility/Accessible.html)**Overrides:**[getAccessibleContext](http://docs.google.com/java/awt/Component.html#getAccessibleContext()) in class [Component](http://docs.google.com/java/awt/Component.html) **Returns:**an AccessibleAWTWindow that serves as the AccessibleContext of this Window**Since:** 1.3

### getGraphicsConfiguration

public [GraphicsConfiguration](http://docs.google.com/java/awt/GraphicsConfiguration.html) **getGraphicsConfiguration**()

This method returns the GraphicsConfiguration used by this Window.

**Overrides:**[getGraphicsConfiguration](http://docs.google.com/java/awt/Component.html#getGraphicsConfiguration()) in class [Component](http://docs.google.com/java/awt/Component.html) **Returns:**the GraphicsConfiguration used by this Component or null**Since:** 1.3

### setLocationRelativeTo

public void **setLocationRelativeTo**([Component](http://docs.google.com/java/awt/Component.html) c)

Sets the location of the window relative to the specified component.

If the component is not currently showing, or c is null, the window is placed at the center of the screen. The center point can be determined with [GraphicsEnvironment.getCenterPoint](http://docs.google.com/java/awt/GraphicsEnvironment.html#getCenterPoint())

If the bottom of the component is offscreen, the window is placed to the side of the Component that is closest to the center of the screen. So if the Component is on the right part of the screen, the Window is placed to its left, and visa versa.

**Parameters:**c - the component in relation to which the window's location is determined**Since:** 1.4 **See Also:**[GraphicsEnvironment.getCenterPoint()](http://docs.google.com/java/awt/GraphicsEnvironment.html#getCenterPoint())

### createBufferStrategy

public void **createBufferStrategy**(int numBuffers)

Creates a new strategy for multi-buffering on this component. Multi-buffering is useful for rendering performance. This method attempts to create the best strategy available with the number of buffers supplied. It will always create a BufferStrategy with that number of buffers. A page-flipping strategy is attempted first, then a blitting strategy using accelerated buffers. Finally, an unaccelerated blitting strategy is used.

Each time this method is called, the existing buffer strategy for this component is discarded.

**Parameters:**numBuffers - number of buffers to create **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if numBuffers is less than 1. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the component is not displayable**Since:** 1.4 **See Also:**[Component.isDisplayable()](http://docs.google.com/java/awt/Component.html#isDisplayable()), [getBufferStrategy()](http://docs.google.com/java/awt/Window.html#getBufferStrategy())

### createBufferStrategy

public void **createBufferStrategy**(int numBuffers,  
 [BufferCapabilities](http://docs.google.com/java/awt/BufferCapabilities.html) caps)  
 throws [AWTException](http://docs.google.com/java/awt/AWTException.html)

Creates a new strategy for multi-buffering on this component with the required buffer capabilities. This is useful, for example, if only accelerated memory or page flipping is desired (as specified by the buffer capabilities).

Each time this method is called, the existing buffer strategy for this component is discarded.

**Parameters:**numBuffers - number of buffers to create, including the front buffercaps - the required capabilities for creating the buffer strategy; cannot be null **Throws:** [AWTException](http://docs.google.com/java/awt/AWTException.html) - if the capabilities supplied could not be supported or met; this may happen, for example, if there is not enough accelerated memory currently available, or if page flipping is specified but not possible. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if numBuffers is less than 1, or if caps is null**Since:** 1.4 **See Also:**[getBufferStrategy()](http://docs.google.com/java/awt/Window.html#getBufferStrategy())

### getBufferStrategy

public [BufferStrategy](http://docs.google.com/java/awt/image/BufferStrategy.html) **getBufferStrategy**()

Returns the BufferStrategy used by this component. This method will return null if a BufferStrategy has not yet been created or has been disposed.

**Returns:**the buffer strategy used by this component**Since:** 1.4 **See Also:**[createBufferStrategy(int)](http://docs.google.com/java/awt/Window.html#createBufferStrategy(int))

### setLocationByPlatform

public void **setLocationByPlatform**(boolean locationByPlatform)

Sets whether this Window should appear at the default location for the native windowing system or at the current location (returned by getLocation) the next time the Window is made visible. This behavior resembles a native window shown without programmatically setting its location. Most windowing systems cascade windows if their locations are not explicitly set. The actual location is determined once the window is shown on the screen.

This behavior can also be enabled by setting the System Property "java.awt.Window.locationByPlatform" to "true", though calls to this method take precedence.

Calls to setVisible, setLocation and setBounds after calling setLocationByPlatform clear this property of the Window.

For example, after the following code is executed:

setLocationByPlatform(true);  
 setVisible(true);  
 boolean flag = isLocationByPlatform();

The window will be shown at platform's default location and flag will be false.

In the following sample:

setLocationByPlatform(true);  
 setLocation(10, 10);  
 boolean flag = isLocationByPlatform();  
 setVisible(true);

The window will be shown at (10, 10) and flag will be false.

**Parameters:**locationByPlatform - true if this Window should appear at the default location, false if at the current location **Throws:** IllegalComponentStateException - if the window is showing on screen and locationByPlatform is true.**Since:** 1.5 **See Also:**[Component.setLocation(int, int)](http://docs.google.com/java/awt/Component.html#setLocation(int,%20int)), [isShowing()](http://docs.google.com/java/awt/Window.html#isShowing()), [setVisible(boolean)](http://docs.google.com/java/awt/Window.html#setVisible(boolean)), [isLocationByPlatform()](http://docs.google.com/java/awt/Window.html#isLocationByPlatform()), [System.getProperty(String)](http://docs.google.com/java/lang/System.html#getProperty(java.lang.String))

### isLocationByPlatform

public boolean **isLocationByPlatform**()

Returns true if this Window will appear at the default location for the native windowing system the next time this Window is made visible. This method always returns false if the Window is showing on the screen.

**Returns:**whether this Window will appear at the default location**Since:** 1.5 **See Also:**[setLocationByPlatform(boolean)](http://docs.google.com/java/awt/Window.html#setLocationByPlatform(boolean)), [isShowing()](http://docs.google.com/java/awt/Window.html#isShowing())

### setBounds

public void **setBounds**(int x,  
 int y,  
 int width,  
 int height)

Moves and resizes this component. The new location of the top-left corner is specified by x and y, and the new size is specified by width and height.

The width or height values are automatically enlarged if either is less than the minimum size as specified by previous call to setMinimumSize.

**Overrides:**[setBounds](http://docs.google.com/java/awt/Component.html#setBounds(int,%20int,%20int,%20int)) in class [Component](http://docs.google.com/java/awt/Component.html) **Parameters:**x - the new *x*-coordinate of this componenty - the new *y*-coordinate of this componentwidth - the new width of this componentheight - the new height of this component**Since:** 1.6 **See Also:**[Component.getBounds()](http://docs.google.com/java/awt/Component.html#getBounds()), [Component.setLocation(int, int)](http://docs.google.com/java/awt/Component.html#setLocation(int,%20int)), [Component.setLocation(Point)](http://docs.google.com/java/awt/Component.html#setLocation(java.awt.Point)), [setSize(int, int)](http://docs.google.com/java/awt/Window.html#setSize(int,%20int)), [setSize(Dimension)](http://docs.google.com/java/awt/Window.html#setSize(java.awt.Dimension)), [setMinimumSize(java.awt.Dimension)](http://docs.google.com/java/awt/Window.html#setMinimumSize(java.awt.Dimension)), [setLocationByPlatform(boolean)](http://docs.google.com/java/awt/Window.html#setLocationByPlatform(boolean)), [isLocationByPlatform()](http://docs.google.com/java/awt/Window.html#isLocationByPlatform())

### setBounds

public void **setBounds**([Rectangle](http://docs.google.com/java/awt/Rectangle.html) r)

Moves and resizes this component to conform to the new bounding rectangle r. This component's new position is specified by r.x and r.y, and its new size is specified by r.width and r.height

The r.width or r.height values will be automatically enlarged if either is less than the minimum size as specified by previous call to setMinimumSize.

**Overrides:**[setBounds](http://docs.google.com/java/awt/Component.html#setBounds(java.awt.Rectangle)) in class [Component](http://docs.google.com/java/awt/Component.html) **Parameters:**r - the new bounding rectangle for this component**Since:** 1.6 **See Also:**[Component.getBounds()](http://docs.google.com/java/awt/Component.html#getBounds()), [Component.setLocation(int, int)](http://docs.google.com/java/awt/Component.html#setLocation(int,%20int)), [Component.setLocation(Point)](http://docs.google.com/java/awt/Component.html#setLocation(java.awt.Point)), [setSize(int, int)](http://docs.google.com/java/awt/Window.html#setSize(int,%20int)), [setSize(Dimension)](http://docs.google.com/java/awt/Window.html#setSize(java.awt.Dimension)), [setMinimumSize(java.awt.Dimension)](http://docs.google.com/java/awt/Window.html#setMinimumSize(java.awt.Dimension)), [setLocationByPlatform(boolean)](http://docs.google.com/java/awt/Window.html#setLocationByPlatform(boolean)), [isLocationByPlatform()](http://docs.google.com/java/awt/Window.html#isLocationByPlatform())

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Window.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/awt/TrayIcon.MessageType.html)   [**NEXT CLASS**](http://docs.google.com/java/awt/Window.AccessibleAWTWindow.html) | [**FRAMES**](http://docs.google.com/index.html?java/awt/Window.html)    [**NO FRAMES**](http://docs.google.com/Window.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | [FIELD](#1t3h5sf) | [CONSTR](#2s8eyo1) | [METHOD](#17dp8vu) | DETAIL: FIELD | [CONSTR](#35nkun2) | [METHOD](#z337ya) |

[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).