AWT WINDOW CLASS

Introduction

The class **Window** is a top level window with no border and no menubar. It uses BorderLayout as default layout manager.

Class declaration

Following is the declaration for **java.awt.Window** class:

```
public class Window
   extends Container
      implements Accessible
```

Class constructors

S.N.	Constructor & Description
1	**Window***Frameowner* Constructs a new, initially invisible window with the specified Frame as its owner.
2	**Window***Windowowner* Constructs a new, initially invisible window with the specified Window as its owner.
3	**Window***Windowowner, GraphicsConfigurationgc* Constructs a new, initially invisible window with the specified owner Window and a GraphicsConfiguration of a screen device.

Class methods

S.N.	Method & Description
1	**void addNotify** Makes this Window displayable by creating the connection to its native screen resource.
2	**void addPropertyChangeListener***PropertyChangeListenerlistener* Adds a PropertyChangeListener to the listener list.
3	**void add Property ChangeListener***StringpropertyName, PropertyChangeListenerlistener* Adds a PropertyChangeListener to the listener list for a specific property.

4

void addWindowFocusListener*WindowFocusListenerl*

Adds the specified window focus listener to receive window events from this window.

5

void addWindowListener*WindowListenerl*

Adds the specified window listener to receive window events from this window.

6

void addWindowStateListener*WindowStateListenerl*

Adds the specified window state listener to receive window events from this window.

7

void applyResourceBundle*ResourceBundlerb*

Deprecated. As of J2SE 1.4, replaced by Component.applyComponentOrientation.

8

void applyResourceBundle*StringrbName*

Deprecated. As of J2SE 1.4, replaced by Component.applyComponentOrientation.

9

void createBufferStrategy*intnumBuffers*

Creates a new strategy for multi-buffering on this component.

10

void createBufferStrategy*intnumBuffers, BufferCapabilitiescaps*

Creates a new strategy for multi-buffering on this component with the required buffer capabilities.

11

void dispose

Releases all of the native screen resources used by this Window, its subcomponents, and all of its owned children.

12

AccessibleContext getAccessibleContext

Gets the AccessibleContext associated with this Window.

13

BufferStrategy getBufferStrategy

Returns the BufferStrategy used by this component.

14

boolean getFocusableWindowState

Returns whether this Window can become the focused Window if it meets the other requirements outlined in isFocusableWindow.

15

Container getFocusCycleRootAncestor

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

16

Component getFocusOwner

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

17

Set<AWTKeyStroke> getFocusTraversalKeys*intid*

Gets a focus traversal key for this Window.

18

GraphicsConfiguration getGraphicsConfiguration

This method returns the GraphicsConfiguration used by this Window.

19

List<Image> getIconImages

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

20

InputContext getInputContext

Gets the input context for this window.

21

<T extends EventListener> T[] getListeners*Class < T > listenerType*

Returns an array of all the objects currently registered as FooListeners upon this Window.

22

Locale getLocale

Gets the Locale object that is associated with this window, if the locale has been set.

23

Dialog.ModalExclusionType getModalExclusionType

Returns the modal exclusion type of this window.

24

Component getMostRecentFocusOwner

Returns the child Component of this Window that will receive the focus when this Window is focused.

25

Window[] getOwnedWindows

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

26

Window getOwner

Returns the owner of this window.

27
static Window[] getOwnerlessWindows

Returns an array of all Windows created by this application that have no owner.

28
Toolkit getToolkit

Returns the toolkit of this frame.

29
String getWarningString

Gets the warning string that is displayed with this window.

30
WindowFocusListener[] getWindowFocusListeners

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

31
WindowListener[] getWindowListeners

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

32
static Window[] getWindows

Returns an array of all Windows, both owned and ownerless, created by this application.

33
WindowStateListener[] getWindowStateListeners

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

34
void hide

Deprecated. As of JDK version 1.5, replaced by setVisible*boolean*.

35
boolean isActive

Returns whether this Window is active.

36
boolean isAlwaysOnTop

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

37
boolean isAlwaysOnTopSupported

Returns whether the always-on-top mode is supported for this window.

38
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.

39

boolean isFocusCycleRoot

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

40

boolean isFocused

Returns whether this Window is focused.

41

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.

42

boolean isShowing

Checks if this Window is showing on screen.

43

void pack

Causes this Window to be sized to fit the preferred size and layouts of its subcomponents.

44

void paint*Graphicsg*

Paints the container.

45

boolean postEvent*Evente*

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

46

protected void processEvent*AWTEvente*

Processes events on this window.

47

protected void processWindowEvent*WindowEvente*

Processes window events occurring on this window by dispatching them to any registered WindowListener objects.

48

protected void processWindowFocusEvent*WindowEvente*

Processes window focus event occuring on this window by dispatching them to any registered WindowFocusListener objects.

49

protected void processWindowStateEvent*WindowEvente*

Processes window state event occuring on this window by dispatching them to any registered WindowStateListener objects.

50

void removeNotify

Makes this Container undisplayable by removing its connection to its native screen resource.

51

void removeWindowFocusListener*WindowFocusListenerl*

Removes the specified window focus listener so that it no longer receives window events from this window.

52

void removeWindowListener*WindowListenerl*

Removes the specified window listener so that it no longer receives window events from this window.

53

void removeWindowStateListener*WindowStateListenerl*

Removes the specified window state listener so that it no longer receives window events from this window.

54

void reshape*intx, inty, intwidth, intheight*

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

55

void setAlwaysOnTop*booleanalwaysOnTop*

Sets whether this window should always be above other windows.

56

void setBounds*intx, inty, intwidth, intheight*

Moves and resizes this component.

57

void setBounds*Rectangler*

Moves and resizes this component to conform to the new bounding rectangle r.

58

void setCursor*Cursorcursor*

Set the cursor image to a specified cursor.

59

void setFocusableWindowState*booleanfocusableWindowState*

Sets whether this Window can become the focused Window if it meets the other requirements outlined in isFocusableWindow.

60

void setFocusCycleRoot*booleanfocusCycleRoot*

Does nothing because Windows must always be roots of a focus traversal cycle.

61
void setIconImage*Imageimage*

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

62
void setIconImages*List < ?extendsImage > icons*

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

63
void setLocationByPlatform*booleanlocationByPlatform*

Sets whether this Window should appear at the default location for the native windowing system or at the current location *returnedbygetLocation* the next time the Window is made visible.

64
void setLocationRelativeTo*Componentc*

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

65
void setMinimumSize*DimensionminimumSize*

Sets the minimum size of this window to a constant value.

66
void setModalExclusionType*Dialog. ModalExclusionTypeexclusionType*

Specifies the modal exclusion type for this window.

67
void setSize*Dimensiond*

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

68
void setSize*intwidth, intheight*

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

69
void setVisible*booleanb*

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

70
void show

Deprecated. As of JDK version 1.5, replaced by setVisible*boolean*.

71
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.

72 **void toFront**

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

Methods inherited

This class inherits methods from the following classes:

- java.awt.Window
- java.awt.Container
- java.awt.Component
- java.lang.Object

Window Example

Create the following java program using any editor of your choice in say **D:/ > AWT > com > tutorialspoint > gui >**

AwtContainerDemo.java

```
package com.tutorialspoint.gui;

import java.awt.*;
import java.awt.event.*;

public class AwtContainerDemo {
   private Frame mainFrame;
   private Label headerLabel;
   private Label statusLabel;
   private Panel controlPanel;
   private Label msglabel;

   public AwtContainerDemo(){
      prepareGUI();
   }

   public static void main(String[] args){
      AwtContainerDemo  awtContainerDemo = new AwtContainerDemo();
      awtContainerDemo.showFrameDemo();
   }

   private void prepareGUI(){
      mainFrame = new Frame("Java AWT Examples");
      mainFrame.setSize(400,400);
      mainFrame.setLayout(new GridLayout(3, 1));
      mainFrame.addWindowListener(new WindowAdapter() {
         public void windowClosing(WindowEvent windowEvent){
            System.exit(0);
         }
      });
      headerLabel = new Label();
      headerLabel.setAlignment(Label.CENTER);
      statusLabel = new Label();
      statusLabel.setAlignment(Label.CENTER);
      statusLabel.setSize(350,100);

      msglabel = new Label();
      msglabel.setAlignment(Label.CENTER);
      msglabel.setText("Welcome to TutorialsPoint AWT Tutorial.");
      controlPanel = new Panel();
      controlPanel.setLayout(new FlowLayout());

      mainFrame.add(headerLabel);
      mainFrame.add(controlPanel);
      mainFrame.add(statusLabel);
      mainFrame.setVisible(true);
   }

   private void showWindowDemo(){
      headerLabel.setText("Container in action: Window");
      final MessageWindow window =
         new MessageWindow(mainFrame,
         "Welcome to TutorialsPoint AWT Tutorial.");

      Button okButton = new Button("Open a Window");
      okButton.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent e) {
            window.setVisible(true);
            statusLabel.setText("A Window shown to the user.");
         }
      });
      controlPanel.add(okButton);
      mainFrame.setVisible(true);
   }

   class MessageWindow extends Window{
      private String message;

      public MessageWindow(Frame parent, String message) {
         super(parent);
         this.message = message;
         setSize(300, 300);
         setLocationRelativeTo(parent);
         setBackground(Color.gray);
      }

      public void paint(Graphics g) {
         super.paint(g);
         g.drawRect(0,0,getSize().width - 1,getSize().height - 1);
         g.drawString(message,50,150);
      }
   }
}
```

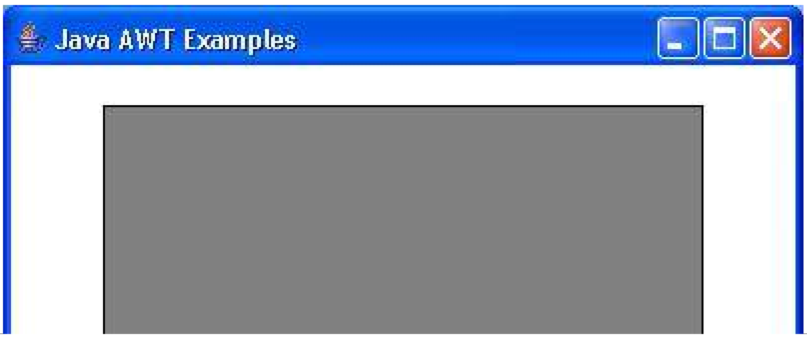
Compile the program using command prompt. Go to **D:/ > AWT** and type the following command.

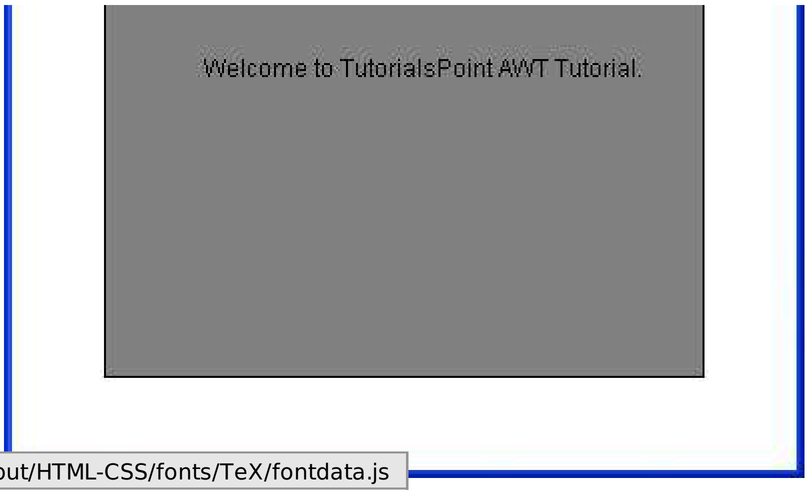
```
D:\AWT>javac com\tutorialspoint\gui\AwtContainerDemo.java
```

If no error comes that means compilation is successful. Run the program using following command.

```
D:\AWT>java com.tutorialspoint.gui.AwtContainerDemo
```

Verify the following output




Welcome to TutorialsPoint AWT Tutorial.