



AWT

abstract window toolkit

tutorialspoint

SIMPLY EASY LEARNING

www.tutorialspoint.com

 <https://www.facebook.com/tutorialspointindia>

 <https://twitter.com/tutorialspoint>

About the Tutorial

JAVA provides a rich set of libraries to create Graphical User Interface (GUI) objects in an platform independent way. Abstract Window Toolkit (AWT) is a set of APIs used by Java programmers to create GUI objects. In this tutorial, we will learn how to use AWT to create GUI objects such as buttons, scroll bars, layout, menus, and more.

Audience

This tutorial is designed for all those software professionals who would like to learn JAVA GUI Programming in simple and easy steps.

Prerequisites

Before proceeding with this tutorial, you should have a basic understanding of Java programming language and how to use it in practice.

Copyright & Disclaimer

© Copyright 2015 by Tutorials Point (I) Pvt. Ltd.

All the content and graphics published in this e-book are the property of Tutorials Point (I) Pvt. Ltd. The user of this e-book is prohibited to reuse, retain, copy, distribute or republish any contents or a part of the contents of this e-book in any manner without written consent of the publisher.

We strive to update the contents of our website and tutorials as timely and as precisely as possible, however, the contents may contain inaccuracies or errors. Tutorials Point (I) Pvt. Ltd. provides no guarantee regarding the accuracy, timeliness, or completeness of our website or its contents including this tutorial. If you discover any errors on our website or in this tutorial, please notify us at contact@tutorialspoint.com

Table of Contents

About the Tutorial.....	i
Audience	i
Prerequisites	i
Copyright & Disclaimer.....	i
Table of Contents	ii
 1. AWT – OVERVIEW	 1
Graphical User Interface.....	1
Basic Terminologies.....	1
Examples of GUI Based Applications	2
Advantages of GUI over CUI	2
 2. AWT – ENVIRONMENT SETUP	 4
Setting up the Path for Windows 2000/XP	4
Setting up the Path for Windows 95/98/ME	4
Setting up the Path for Linux, UNIX, Solaris, FreeBSD	4
Popular Java Editors	4
 3. AWT – CONTROLS.....	 6
AWT Component Class	7
AWT UI Elements	20
AWT Label Class	21
AWT Button Class.....	25
AWT CheckBox Class	30
AWT CheckBoxGroup Class	35
AWT List Class	40
AWT TextField Class	46

AWT TextArea Class	51
AWT Choice Class	57
AWT Canvas Class	62
AWT Image Class	66
AWT Scrollbar Class.....	71
AWT Dialog Class.....	77
AWT FileDialog Class	83
4. AWT – EVENT HANDLING	89
What is an Event?	89
Types of Event.....	89
What is Event Handling?	89
Callback Methods.....	90
Event Handling Example.....	90
5. AWT – EVENT CLASSES	95
EventObject Class.....	95
Class Declaration	95
Field	95
Class Constructors	95
Class Methods	95
Methods Inherited	96
AWT Event Classes	96
AWT AWTEvent Class	97
AWT ActionEvent Class	99
AWT InputEvent Class	100
AWT KeyEvent Class	102
AWT MouseEvent Class	111

AWT TextEvent Class	114
AWT WindowEvent Class	115
AWT AdjustmentEvent Class	117
AWT ComponentEvent Class	118
AWT ContainerEvent Class	119
AWT MouseMotionEvent Class	121
AWT PaintEvent Class.....	121
6. AWT – EVENT LISTENERS.....	128
EventListener Interface	128
Class Declaration	128
AWT Event Listener Interfaces	128
AWT ActionListener Interface	129
AWT ComponentListener Interface	132
AWT ItemListener Interface	137
AWT KeyListener Interface	140
AWT MouseListener Interface	144
AWT TextListener Interface	148
AWT WindowListener Interface	152
AWT AdjustmentListener Interface	157
AWT ContainerListener Interface	160
AWT MouseMotionListener Interface	164
AWT FocusListener Interface	168
7. AWT – EVENT ADAPTERS.....	173
AWT Adapters	173
AWT FocusAdapter Class	173
AWT KeyAdapter Class	177

AWT MouseAdapter Class	181
AWT MouseMotionAdapter Class	185
AWT WindowAdapter Class	189
8. AWT – LAYOUTS	194
Introduction	194
Layout Manager	194
AWT Layout Manager Interface.....	195
AWT LayoutManager2 Interface.....	195
AWT Layout Manager Classes.....	196
AWT BorderLayout Class	197
AWT CardLayout Class.....	202
AWT FlowLayout Class	207
AWT GridLayout Class	212
AWT GridBagLayout Class	217
9. AWT – CONTAINERS	224
AWT Container Class	224
AWT UI Elements	228
AWT Panel Class.....	229
Class Constructors	229
AWT Frame Class.....	232
AWT Window Class	239
10. AWT – MENU CLASSES	248
Menu Hierarchy.....	248
Menu Controls	248
AWT MenuComponent Class.....	249
AWT MenuBar Class	250

AWT MenuItem Class	256
AWT Menu Class	263
AWT CheckboxMenuItem Class	270
AWT PopupMenu Class	276
11. AWT – GRAPHICS CLASSES.....	281
Graphics Controls	281
AWT Graphics Class	282
AWT Graphics2D Class.....	287
AWT Arc2D Class	292
AWT CubicCurve2D Class.....	297
AWT Ellipse2D Class	302
AWT Rectangle2D Class	305
AWT QuadCurve2D Class.....	310
AWT Line2D Class	315
AWT Font Class	319
AWT Color Class	327
AWT BasicStroke Class	332

1. AWT – OVERVIEW

Graphical User Interface

Graphical User Interface (GUI) offers user interaction via some graphical components. For example, our underlying Operating System also offers GUI via window, frame, Panel, Button, Textfield, TextArea, Listbox, Combobox, Label, Checkbox etc. These all are known as components. Using these components, we can create an interactive user interface for an application.

GUI provides result to end-users in response to its raised events. It is entirely based on events. For example, clicking on a button, closing a window, opening a window, typing something in a text area etc. These activities are known as events. GUI makes it easier for the end user to use an application. It also makes them interesting.

Basic Terminologies

Term	Description
Component	Component is an object having a graphical representation that can be displayed on the screen and that can interact with the user. For example, buttons, checkboxes, list and scrollbars of a graphical user interface.
Container	Container object is a component that can contain other components. Components added to a container are tracked in a list. The order of the list will define the components' front-to-back stacking order within the container. If no index is specified when adding a component to a container, it will be added to the end of the list.
Panel	Panel provides space in which an application can attach any other components, including other panels.
Window	Window is a rectangular area, which is displayed on the screen. In a different window, we can execute different program and display different data. Window provide us with multitasking environment. A window must have either a frame, dialog, or another window defined as its owner when it's constructed.
Frame	A Frame is a top-level window with a title and a border. The size of the frame includes any area designated for the border. Frame encapsulates window . It and has a title bar, menu bar, borders, and resizing corners.

Canvas	Canvas component represents a blank rectangular area of the screen onto which the application can draw. Application can also trap input events from the use of the blank area of Canvas component.
--------	--

Examples of GUI Based Applications

Following are some of the examples for GUI based applications:

- Automated Teller Machine (ATM)
- Airline Ticketing System
- Information Kiosks at railway stations
- Mobile Applications
- Navigation Systems

Advantages of GUI over CUI

- GUI provides graphical icons to interact while the CUI (Character User Interface) offers the simple text-based interfaces.
- GUI makes the application more entertaining and interesting on the other hand CUI does not.
- GUI offers click and execute environment while in CUI every time, we have to enter the command for a task.
- New user can easily interact with graphical user interface by the visual indicators, but it is difficult in Character user interface.
- GUI offers a lot of controls of file system and the operating system while in CUI, you have to use commands, which is difficult to remember.
- Windows concept in GUI allows the user to view, manipulate, and control the multiple applications at once while in CUI, user can control one task at a time.
- GUI provides multitasking environment so as the CUI also does, but CUI does not provide same ease as the GUI do.
- Using GUI, it is easier to control and navigate the operating system, which becomes very slow in command user interface.
- GUI can be easily customized but CUI cannot be.

2. AWT – ENVIRONMENT SETUP

This chapter guides you on how to download and set up Java on your computer. Please follow the steps given below to set up the environment.

Java SE is freely available on the link [Download Java](#). So you download a version based on your operating system.

Follow the instructions to download java and run the **.exe** to install Java on your computer. Once you installed Java on your computer, you would need to set up environment variables to point to correct installation directories.

Setting up the Path for Windows 2000/XP

Assuming you have installed Java in [c:\Program Files\java\jdk directory](#):

- Right-click on 'My Computer' and select 'Properties'.
- Click on the 'Environment variables' button under the 'Advanced' tab.
- Now alter the 'Path' variable so that it also contains the path to the Java executable. For example, if the path is currently set to 'C:\WINDOWS\SYSTEM32', then change your path to read 'C:\WINDOWS\SYSTEM32;c:\Program Files\java\jdk\bin'.

Setting up the Path for Windows 95/98/ME

Assuming you have installed Java in [c:\Program Files\java\jdk directory](#):

- Edit the 'C:\autoexec.bat' file and add the following line at the end:
'SET PATH=%PATH%;[C:\Program Files\java\jdk\bin](#)'

Setting up the Path for Linux, UNIX, Solaris, FreeBSD

Environment variable PATH should be set to point — where the java binaries have been installed. Refer to your shell documentation, if you have trouble doing this.

Example, if you use *bash* as your shell, then you would add the following line to the end of your '.bashrc: export PATH=/path/to/java:\$PATH'

Popular Java Editors

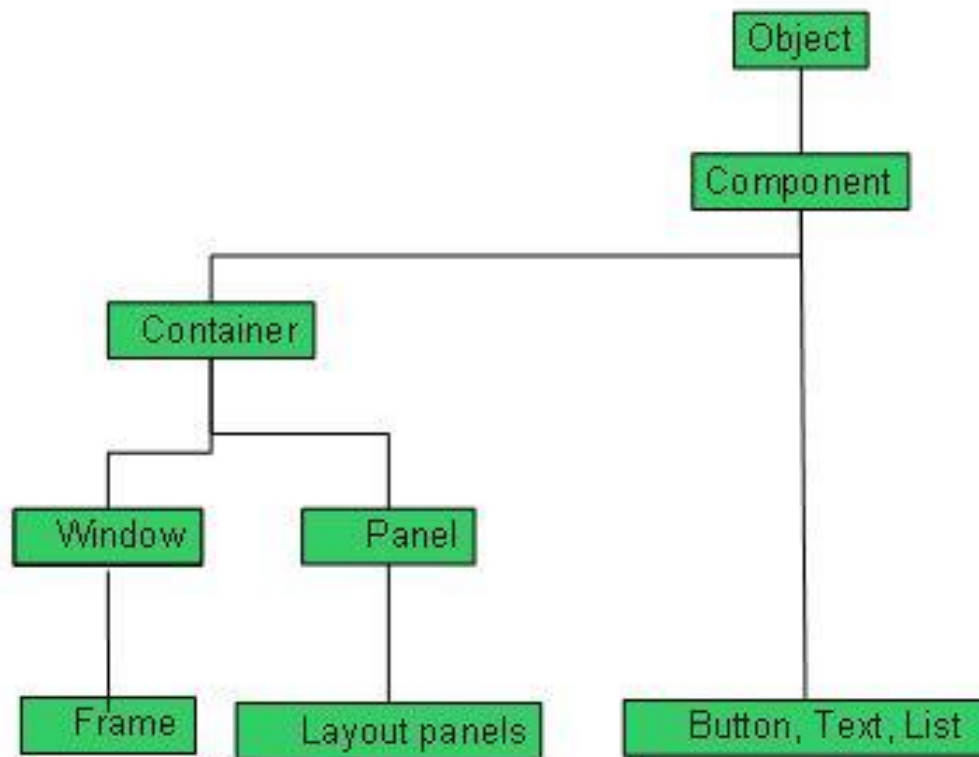
To write your java programs, you will need a text editor. There are even more sophisticated IDE available in the market. But for now, you can consider one of the following:

- **Notepad:** On Windows system, you can use any simple text editor like Notepad (Recommended for this tutorial), TextPad.
- **Netbeans:** It is a Java IDE that is open source and free, it can be downloaded from <http://www.netbeans.org/index.html>.
- **Eclipse:** It is also a java IDE developed by the eclipse open source community and can be downloaded from <http://www.eclipse.org/>.

3. AWT – CONTROLS

Every user interface considers the following three main aspects:

- **UI elements:** These are the core visual elements, the user eventually sees and interacts with. GWT provides a huge list of widely used and common elements varying from basic to complex. We will discuss all these in this tutorial.
- **Layouts:** They define how UI elements should be organized on the screen and provide a final look and feel to the GUI (Graphical User Interface). This part will be covered in Layout chapter.
- **Behavior:** These are events that occur when the user interacts with UI elements. This part will be covered in Event Handling chapter.



Every AWT controls inherit properties from Component class.

Sr. No.	Control & Description
1	Component A Component is an abstract super class for GUI controls and it represents an object with graphical representation.

AWT Component Class

The class **Component** is the abstract base class for the non-menu user-interface controls of AWT. Component represents an object with graphical representation.

Class Declaration

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

```
public abstract class Component
    extends Object
        implements ImageObserver, MenuContainer, Serializable
```

Field

Following are the fields for **java.awt.Component** class:

- **static float BOTTOM_ALIGNMENT** -- Ease-of-use constant for getAlignmentY.
- **static float CENTER_ALIGNMENT** -- Ease-of-use constant for getAlignmentY and getAlignmentX.
- **static float LEFT_ALIGNMENT** -- Ease-of-use constant for getAlignmentX.
- **static float RIGHT_ALIGNMENT** -- Ease-of-use constant for getAlignmentX.
- **static float TOP_ALIGNMENT** -- Ease-of-use constant for getAlignmentY().

Class Constructors

S.N.	Constructor & Description
1	protected Component() This creates a new Component.

Class Methods

S.N.	Method & Description
1	boolean action(Event evt, Object what)

	Deprecated. As of JDK version 1.1, should register this component as ActionListener on component which fires action events.
2	void add(PopupMenu popup) Adds the specified popup menu to the component.
3	void addComponentListener(ComponentListener l) Adds the specified component listener to receive component events from this component.
4	void addFocusListener(FocusListener l) Adds the specified focus listener to receive focus events from this component when this component gains input focus.
5	void addHierarchyBoundsListener(HierarchyBoundsListener l) Adds the specified hierarchy bounds listener to receive hierarchy bounds events from this component when the hierarchy to which this container belongs changes.
6	void addHierarchyListener(HierarchyListener l) Adds the specified hierarchy listener to receive hierarchy changed events from this component when the hierarchy to which this container belongs changes.
7	void addInputMethodListener(InputMethodListener l) Adds the specified input method listener to receive input method events from this component.
8	void addKeyListener(KeyListener l) Adds the specified key listener to receive key events from this component.
9	void addMouseListener(MouseListener l) Adds the specified mouse listener to receive mouse events from this component.
10	void addMouseMotionListener(MouseMotionListener l) Adds the specified mouse motion listener to receive mouse motion events from this component.
11	void addMouseWheelListener(MouseWheelListener l) Adds the specified mouse wheel listener to receive mouse wheel events from this component.
12	void addNotify() Makes this Component displayable by connecting it to a native screen resource.
13	void addPropertyChangeListener(PropertyChangeListener listener) Adds a PropertyChangeListener to the listener list.
14	void addPropertyChangeListener(String propertyName, PropertyChangeListener listener) Adds a PropertyChangeListener to the listener list for a specific property.
15	void applyComponentOrientation(ComponentOrientation orientation) Sets the ComponentOrientation property of this component and all components contained within it.
16	boolean areFocusTraversalKeysSet(int id) Returns whether the Set of focus traversal keys for the given focus traversal operation has been explicitly defined for this Component.
17	int checkImage(Image image, ImageObserver observer) Returns the status of the construction of a screen representation of the specified image.
18	int checkImage(Image image, int width, int height, ImageObserver observer)

	Returns the status of the construction of a screen representation of the specified image.
19	boolean contains(int x,int y) Checks whether this component "contains" the specified point, where x and y are defined to be relative to the coordinate system of this component.
20	boolean contains(Point p) Checks whether this component "contains" the specified point, where the points x and y coordinates are defined to be relative to the coordinate system of this component.
21	Image createImage(ImageProducer producer) Creates an image from the specified image producer.
22	Image createImage(int width,int height) Creates an off-screen drawable image to be used for double buffering.
23	VolatileImage createVolatileImage(int width,int height) Creates a volatile off-screen drawable image to be used for double buffering.
24	VolatileImage createVolatileImage(int width,int height, ImageCapabilities caps) Creates a volatile off-screen drawable image, with the given capabilities.
25	void deliverEvent(Event e) Deprecated. As of JDK version 1.1, replaced by dispatchEvent(AWTEvent e).
26	void disable() Deprecated. As of JDK version 1.1, replaced by setEnabled(boolean).
27	protected void disableEvents(long eventsToDisable) Disables the events defined by the specified event mask parameter from being delivered to this component.
28	void dispatchEvent(AWTEvent e) Dispatches an event to this component or one of its sub components.
29	void doLayout() Prompts the layout manager to lay out this component.
30	void enable() Deprecated. As of JDK version 1.1, replaced by setEnabled(boolean).
31	void enable(boolean b) Deprecated. As of JDK version 1.1, replaced by setEnabled(boolean).
32	protected void enableEvents(long eventsToEnable) Enables the events defined by the specified event mask parameter to be delivered to this component.
33	void enableInputMethods(boolean enable) Enables or disables input method support for this component.
34	protected void firePropertyChange(String propertyName, boolean oldValue, boolean newValue) Support for reporting bound property changes for boolean properties.
35	void firePropertyChange(String propertyName, byte oldValue, byte newValue) Reports a bound property change.
36	void firePropertyChange(String propertyName, char oldValue, char newValue) Reports a bound property change.

37	void firePropertyChange(String propertyName, double oldValue, double newValue) Reports a bound property change.
38	void firePropertyChange(String propertyName, float oldValue, float newValue) Reports a bound property change.
39	void firePropertyChange(String propertyName, long oldValue, long newValue) Reports a bound property change.
40	protected void firePropertyChange(String propertyName, Object oldValue, Object newValue) Support for reporting bound property changes for Object properties.
41	void firePropertyChange(String propertyName, short oldValue, short newValue) Reports a bound property change.
42	AccessibleContext getAccessibleContext() Gets the AccessibleContext associated with this Component.
43	float getAlignmentX() Returns the alignment along the x axis.
44	float getAlignmentY() Returns the alignment along the y axis.
45	Color getBackground() Gets the background color of this component.
46	int getBaseline(int width,int height) Returns the baseline.
47	Component.BaselineResizeBehavior getBaselineResizeBehavior() Returns an enum indicating how the baseline of the component changes as the size changes.
48	Rectangle getBounds() Gets the bounds of this component in the form of a Rectangle object.
49	Rectangle getBounds(Rectangle rv) Stores the bounds of this component into return value rv and return rv.
50	ColorModel getColorModel() Gets the instance of ColorModel used to display the component on the output device.
51	Component getComponentAt(int x,int y) Determines if this component or one of its immediate subcomponents contains the (x, y) location, and if so, returns the containing component.
52	Component getComponentAt(Point p) Returns the component or subcomponent that contains the specified point.
53	ComponentListener[] getComponentListeners() Returns an array of all the component listeners registered on this component.
54	ComponentOrientation getComponentOrientation() Retrieves the language-sensitive orientation that is to be used to order the elements or text within this component.
55	Cursor getCursor() Gets the cursor set in the component.
56	DropTarget getDropTarget() Gets the DropTarget associated with this Component.
57	Container getFocusCycleRootAncestor()

	Returns the Container which is the focus cycle root of this Component's focus traversal cycle.
58	FocusListener[] getFocusListeners() Returns an array of all the focus listeners registered on this component.
59	Set<AWTKeyStroke> getFocusTraversalKeys(int id) Returns the Set of focus traversal keys for a given traversal operation for this Component.
60	boolean getFocusTraversalKeysEnabled() Returns whether focus traversal keys are enabled for this Component.
61	Font getFont() Gets the font of this component.
62	FontMetrics getFontMetrics(Font font) Gets the font metrics for the specified font.
63	Color getForeground() Gets the foreground color of this component.
64	Graphics getGraphics() Creates a graphics context for this component.
65	GraphicsConfiguration getGraphicsConfiguration() Gets the GraphicsConfiguration associated with this Component.
66	int getHeight() Returns the current height of this component.
67	HierarchyBoundsListener[] getHierarchyBoundsListeners() Returns an array of all the hierarchy bounds listeners registered on this component.
68	HierarchyListener[] getHierarchyListeners() Returns an array of all the hierarchy listeners registered on this component.
69	boolean getIgnoreRepaint()
70	InputContext getInputContext() Gets the input context used by this component for handling the communication with input methods when text is entered in this component.
71	InputMethodListener[] getInputMethodListeners() Returns an array of all the input method listeners registered on this component.
72	InputMethodRequests getInputMethodRequests() Gets the input method request handler which supports requests from input methods for this component.
73	KeyListener[] getKeyListeners() Returns an array of all the key listeners registered on this component.
74	<T extends EventListener> T[] getListeners(Class<T> listenerType) Returns an array of all the objects currently registered as FooListeners upon this Component.
75	Locale getLocale() Gets the locale of this component.
76	Point getLocation() Gets the location of this component in the form of a point specifying the component's top-left corner.
77	Point getLocation(Point rv) Stores the x,y origin of this component into return value rv and return rv.
78	Point getLocationOnScreen()

	Gets the location of this component in the form of a point specifying the component's top-left corner in the screen's coordinate space.
79	Dimension getMaximumSize() Gets the maximum size of this component.
80	Dimension getMinimumSize() Gets the minimum size of this component.
81	MouseListener[] getMouseListeners() Returns an array of all the mouse listeners registered on this component.
82	MouseMotionListener[] getMouseMotionListeners() Returns an array of all the mouse motion listeners registered on this component.
83	Point getMousePosition() Returns the position of the mouse pointer in this Component's coordinate space if the Component is directly under the mouse pointer, otherwise returns null.
84	MouseWheelListener[] getMouseWheelListeners() Returns an array of all the mouse wheel listeners registered on this component.
85	String getName() Gets the name of the component.
86	Container getParent() Gets the parent of this component.
87	java.awt.peer.ComponentPeer getPeer() Deprecated. As of JDK version 1.1, programs should not directly manipulate peers; replaced by boolean isDisplayable().
88	Dimension getPreferredSize() Gets the preferred size of this component.
89	PropertyChangeListener[] getPropertyChangeListeners() Returns an array of all the property change listeners registered on this component.
90	PropertyChangeListener[] getPropertyChangeListeners (String propertyName) Returns an array of all the listeners which have been associated with the named property.
91	Dimension getSize() Returns the size of this component in the form of a Dimension object.
92	Dimension getSize(Dimension rv)Stores the width/height of this component into return value rv and return rv.
93	Toolkit getToolkit() Gets the toolkit of this component.
94	Object getTreeLock() Gets this component's locking object (the object that owns the thread synchronization monitor) for AWT component-tree and layout operations.
95	int getWidth() Returns the current width of this component.
96	int getX() Returns the current x coordinate of the components origin.
97	int getY() Returns the current y coordinate of the components origin.

98	boolean gotFocus(Event evt, Object what) Deprecated. As of JDK version 1.1, replaced by processFocusEvent(FocusEvent)
99	boolean handleEvent(Event evt) Deprecated. As of JDK version 1.1 replaced by processEvent(AWTEvent).
100	boolean hasFocus() Returns true if this Component is the focus owner.
101	void hide() Deprecated. As of JDK version 1.1, replaced by setVisible(boolean).
102	boolean imageUpdate(Image img,int infoflags,int x,int y,int w,int h) Repaints the component when the image has changed.
103	boolean inside(int x,int y) Deprecated. As of JDK version 1.1, replaced by contains(int, int).
104	void invalidate() Invalidates this component.
105	boolean isBackgroundSet() Returns whether the background color has been explicitly set for this Component.
106	boolean isCursorSet() Returns whether the cursor has been explicitly set for this Component.
107	boolean isDisplayable() Determines whether this component is displayable.
108	boolean isDoubleBuffered() Returns true if this component is painted to an offscreen image (buffer) that's copied to the screen later.
109	boolean isEnabled() Determines whether this component is enabled.
110	boolean isFocusable() Returns whether this Component can be focused.
111	boolean isFocusCycleRoot(Container container) Returns whether the specified Container is the focus cycle root of this Component's focus traversal cycle.
112	boolean isFocusOwner() Returns true if this Component is the focus owner.
113	boolean isFocusTraversable() Deprecated. As of 1.4, replaced by isFocusable().
114	boolean isFontSet() Returns whether the font has been explicitly set for this Component.
115	boolean isForegroundSet() Returns whether the foreground color has been explicitly set for this Component.
116	boolean isLightweight() A lightweight component doesn't have a native toolkit peer.
117	boolean isMaximumSizeSet() Returns true if the maximum size has been set to a non-null value otherwise returns false.
118	boolean isMinimumSizeSet() Returns whether or not setMinimumSize has been invoked with a non-null value.
119	boolean isOpaque()

	Returns true if this component is completely opaque, returns false by default.
120	boolean isPreferredSizeSet() Returns true if the preferred size has been set to a non-null value otherwise returns false.
121	boolean isShowing() Determines whether this component is showing on screen.
122	boolean isValid() Determines whether this component is valid.
123	boolean isVisible() Determines whether this component should be visible when its parent is visible.
124	boolean keyDown(Event evt,int key) Deprecated. As of JDK version 1.1, replaced by processKeyEvent(KeyEvent).
125	boolean keyUp(Event evt,int key) Deprecated. As of JDK version 1.1, replaced by processKeyEvent(KeyEvent).
126	void layout() Deprecated. As of JDK version 1.1, replaced by doLayout().
127	void list() Prints a listing of this component to the standard system output stream System.out.
128	void list(PrintStream out) Prints a listing of this component to the specified output stream.
129	void list(PrintStream out,int indent) Prints out a list, starting at the specified indentation, to the specified print stream.
130	void list(PrintWriter out) Prints a listing to the specified print writer.
131	void list(PrintWriter out,int indent) Prints out a list, starting at the specified indentation, to the specified print writer.
132	Component locate(int x,int y) Deprecated. As of JDK version 1.1, replaced by getComponentAt(int, int).
133	Point location() Deprecated. As of JDK version 1.1, replaced by getLocation().
134	boolean lostFocus(Event evt, Object what) Deprecated. As of JDK version 1.1, replaced by processFocusEvent(FocusEvent).
135	boolean mouseDown(Event evt,int x,int y) Deprecated. As of JDK version 1.1, replaced by processMouseEvent(MouseEvent).
136	boolean mouseDrag(Event evt,int x,int y) Deprecated. As of JDK version 1.1, replaced by processMouseMotionEvent(MouseEvent).
137	boolean mouseEnter(Event evt,int x,int y) Deprecated. As of JDK version 1.1, replaced by processMouseEvent(MouseEvent).
138	boolean mouseExit(Event evt,int x,int y) Deprecated. As of JDK version 1.1, replaced by processMouseEvent(MouseEvent)..
139	boolean mouseMove(Event evt,int x,int y)

	Deprecated. As of JDK version 1.1, replaced by processMouseEvent(MouseEvent)..
140	boolean mouseUp(Event evt,int x,int y) Deprecated. As of JDK version 1.1, replaced by processMouseEvent(MouseEvent).
141	void move(int x,int y) Deprecated. As of JDK version 1.1, replaced by setLocation(int, int).
142	void nextFocus() Deprecated. As of JDK version 1.1, replaced by transferFocus().
143	void paint(Graphics g) Paints this component.
144	void paintAll(Graphics g) Paints this component and all of its subcomponents.
145	boolean postEvent(Event e) Deprecated. As of JDK version 1.1, replaced by dispatchEvent(AWTEvent).
146	boolean prepareImage(Image image,int width,int height, ImageObserver observer) Prepares an image for rendering on this component at the specified width and height.
147	void print(Graphics g) Prints this component.
148	void printAll(Graphics g) Prints this component and all of its subcomponents.
149	protected void processComponentEvent(ComponentEvent e) Processes component events occurring on this component by dispatching them to any registered ComponentListener objects.
150	protected void processEvent(AWTEvent e) Processes events occurring on this component.
151	protected void processFocusEvent(FocusEvent e) Processes focus events occurring on this component by dispatching them to any registered FocusListener objects.
152	protected void processHierarchyBoundsEvent(HierarchyEvent e) Processes hierarchy bounds events occurring on this component by dispatching them to any registered HierarchyBoundsListener objects.
153	protected void processHierarchyEvent(HierarchyEvent e) Processes hierarchy events occurring on this component by dispatching them to any registered HierarchyListener objects.
154	protected void processInputMethodEvent(InputMethodEvent e) Processes input method events occurring on this component by dispatching them to any registered InputMethodListener objects.
155	protected void processKeyEvent(KeyEvent e) Processes key events occurring on this component by dispatching them to any registered KeyListener objects.
156	protected void processMouseEvent(MouseEvent e) Processes mouse events occurring on this component by dispatching them to any registered MouseListener objects.
157	protected void processMouseMotionEvent(MouseEvent e) Processes mouse motion events occurring on this component by dispatching them to any registered MouseMotionListener objects.
158	protected void processMouseWheelEvent(MouseWheelEvent e)

	Processes mouse wheel events occurring on this component by dispatching them to any registered MouseWheelListener objects.
159	void remove(MenuComponent popup) Removes the specified popup menu from the component.
160	void removeComponentListener(ComponentListener l) Removes the specified component listener so that it no longer receives component events from this component.
161	void removeFocusListener(FocusListener l) Removes the specified focus listener so that it no longer receives focus events from this component.
162	void removeHierarchyBoundsListener(HierarchyBoundsListener l) Removes the specified hierarchy bounds listener so that it no longer receives hierarchy bounds events from this component.
163	void removeHierarchyListener(HierarchyListener l) Removes the specified hierarchy listener so that it no longer receives hierarchy changed events from this component.
164	void removeInputMethodListener(InputMethodListener l) Removes the specified input method listener so that it no longer receives input method events from this component.
165	void removeKeyListener(KeyListener l) Removes the specified key listener so that it no longer receives key events from this component.
166	void removeMouseListener(MouseListener l) Removes the specified mouse listener so that it no longer receives mouse events from this component.
167	void removeMouseMotionListener(MouseMotionListener l) Removes the specified mouse motion listener so that it no longer receives mouse motion events from this component.
168	void removeMouseWheelListener(MouseWheelListener l) Removes the specified mouse wheel listener so that it no longer receives mouse wheel events from this component.
169	void removeNotify() Makes this Component undisplayable by destroying its native screen resource.
170	void removePropertyChangeListener(PropertyChangeListener listener) Removes a PropertyChangeListener from the listener list.
171	void removePropertyChangeListener(String propertyName, PropertyChangeListener listener) Removes a PropertyChangeListener from the listener list for a specific property.
172	void repaint() Repaints this component.
173	void repaint(int x,int y,int width,int height) Repaints the specified rectangle of this component.
174	void repaint(long tm) Repaints the component.
175	void repaint(long tm,int x,int y,int width,int height) Repaints the specified rectangle of this component within tm milliseconds.
176	void requestFocus()

	Requests that this Component get the input focus, and that this Component's top-level ancestor become the focused Window.
177	protected boolean requestFocus(boolean temporary) Requests that this Component get the input focus, and that this Component's top-level ancestor become the focused Window.
178	boolean requestFocusInWindow() Requests that this Component get the input focus, if this Component's top-level ancestor is already the focused Window.
179	protected boolean requestFocusInWindow(boolean temporary) Requests that this Component get the input focus, if this Component's top-level ancestor is already the focused Window.
180	void reshape(int x,int y,int width,int height) Deprecated. As of JDK version 1.1, replaced by <code>setBounds(int, int, int, int)</code> .
181	void resize(Dimension d) Deprecated. As of JDK version 1.1, replaced by <code>setSize(Dimension)</code> .
182	void resize(int width,int height) Deprecated. As of JDK version 1.1, replaced by <code>setSize(int, int)</code> .
183	void setBackground(Color c) Sets the background color of this component.
184	void setBounds(int x,int y,int width,int height) Moves and resizes this component.
185	void setBounds(Rectangle r) Moves and resizes this component to conform to the new bounding rectangle r.
186	void setComponentOrientation(ComponentOrientation o) Sets the language-sensitive orientation that is to be used to order the elements or text within this component.
187	void setCursor(Cursor cursor) Sets the cursor image to the specified cursor.
188	void setDropTarget(DropTarget dt) Associate a DropTarget with this component.
189	void setEnabled(boolean b) Enables or disables this component, depending on the value of the parameter b.
190	void setFocusable(boolean focusable) Sets the focusable state of this Component to the specified value.
191	void setFocusTraversalKeys(int id, Set<? extends AWTKeyStroke> keystrokes) Sets the focus traversal keys for a given traversal operation for this Component.
192	void setFocusTraversalKeysEnabled(boolean focusTraversalKeysEnabled) Sets whether focus traversal keys are enabled for this Component.
193	void setFont(Font f) Sets the font of this component.
194	void setForeground(Color c) Sets the foreground color of this component.
195	void setIgnoreRepaint(boolean ignoreRepaint) Sets whether or not paint messages received from the operating system should be ignored.

196	void setLocale(Locale l) Sets the locale of this component.
197	void setLocation(int x,int y) Moves this component to a new location.
198	void setLocation(Point p) Moves this component to a new location.
199	void setMaximumSize(Dimension maximumSize) Sets the maximum size of this component to a constant value.
200	void setMinimumSize(Dimension minimumSize) Sets the minimum size of this component to a constant value.
201	void setName(String name) Sets the name of the component to the specified string.
202	void setPreferredSize(Dimension preferredSize) Sets the preferred size of this component to a constant value.
203	void setSize(Dimension d) Resizes this component so that it has width d.width and height d.height.
204	void setSize(int width,int height) Resizes this component so that it has width width and height height.
205	void setVisible(boolean b) Shows or hides this component depending on the value of parameter b.
206	void show() Deprecated. As of JDK version 1.1, replaced by setVisible(boolean).
207	void show(boolean b) Deprecated. As of JDK version 1.1, replaced by setVisible(boolean).
208	Dimension size() Deprecated. As of JDK version 1.1, replaced by getSize().
209	String toString() Returns a string representation of this component and its values.
210	void transferFocus() Transfers the focus to the next component, as though this Component were the focus owner.
211	void transferFocusBackward() Transfers the focus to the previous component, as though this Component were the focus owner.
212	void transferFocusUpCycle() Transfers the focus up one focus traversal cycle.
213	void update(Graphics g) Updates this component.
214	void validate() Ensures that this component has a valid layout.
215	Rectangle bounds() Deprecated. As of JDK version 1.1, replaced by getBounds().
216	protected AWTEvent coalesceEvents(AWTEvent existingEvent, AWTEvent newEvent) Potentially coalesce an event being posted with an existing event.
217	protected String paramString() Returns a string representing the state of this component.
218	protected void firePropertyChange(String propertyName,int oldValue,int newValue) Support for reporting bound property changes for integer properties.

219	Dimension preferredSize() Deprecated. As of JDK version 1.1, replaced by <code>getPreferredSize()</code> .
220	boolean prepareImage(Image image, ImageObserver observer) Prepares an image for rendering on this component.
221	Dimension minimumSize() Deprecated. As of JDK version 1.1, replaced by <code>getMinimumSize()</code> .

Methods Inherited

This class inherits methods from the following classes:

- `java.lang.Object`

AWT UI Elements

Following is the list of commonly used controls while designing GUI using AWT:

Sr. No.	Control & Description
1	Label A Label object is a component for placing text in a container.
2	Button This class creates a labeled button.
3	Check Box A check box is a graphical component that can be in either an on (true) or off (false) state.
4	Check Box Group The CheckboxGroup class is used to group the set of checkbox.
5	List The List component presents the user with a scrolling list of text items.
6	Text Field A TextField object is a text component that allows for the editing of a single line of text.
7	Text Area A TextArea object is a text component that allows for the editing of a multiple lines of text.
8	Choice A Choice control is used to show pop up menu of choices. Selected choice is shown on the top of the menu.

9	Canvas A Canvas control represents a rectangular area where application can draw something or can receive inputs created by user.
10	Image An Image control is superclass for all image classes representing graphical images.
11	Scroll Bar A Scrollbar control represents a scroll bar component in order to enable user to select from range of values.
12	Dialog A Dialog control represents a top-level window with a title and a border used to take some form of input from the user.
13	File Dialog A FileDialog control represents a dialog window from which the user can select a file.

End of ebook preview

If you liked what you saw...

Buy it from our store @ **<https://store.tutorialspoint.com>**