**8 Event Programming**

**8.1 Java awt components :-**

**Java AWT** (Abstract Window Toolkit) is an API to develop GUI or window-based applications in java.

## AWT Components

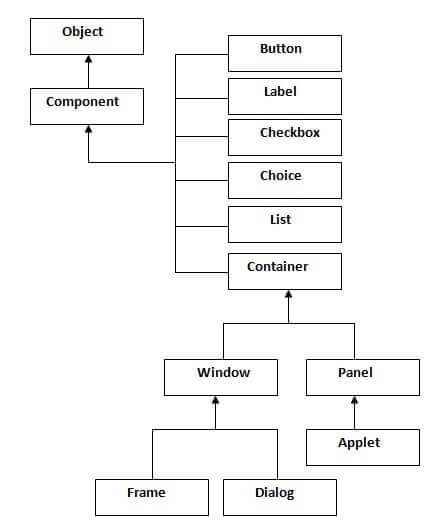
A component is an object with a graphical representation that can be displayed on the screen and that can interact with the user. The http://www.eeng.dcu.ie/~ee553/ee402notes/html/images/derek_class.gifComponent class is the abstract parent of the nonmenu-related AWT component.

Java AWT components are platform-dependent i.e. components are displayed according to the view of operating system. AWT is heavyweight i.e. its components are using the resources of OS.

The java.awt [package](https://www.javatpoint.com/package) provides [classes](https://www.javatpoint.com/object-and-class-in-java) for AWT api such as [TextField](https://www.javatpoint.com/java-awt-textfield), [Label](https://www.javatpoint.com/java-awt-label), [TextArea](https://www.javatpoint.com/java-awt-textarea), RadioButton, [CheckBox](https://www.javatpoint.com/java-awt-checkbox), [Choice](https://www.javatpoint.com/java-awt-choice), [List](https://www.javatpoint.com/java-awt-list) etc.

### Java AWT Hierarchy

The hierarchy of Java AWT classes are given below.



### Container

The Container is a component in AWT that can contain another components like [buttons](https://www.javatpoint.com/java-awt-button), textfields, labels etc. The classes that extends Container class are known as container such as Frame, Dialog and Panel.

### Window

The window is the container that have no borders and menu bars. You must use frame, dialog or another window for creating a window.

### Panel

The Panel is the container that doesn't contain title bar and menu bars. It can have other components like button, textfield etc.

### Frame

The Frame is the container that contain title bar and can have menu bars. It can have other components like button, textfield etc.

### Useful Methods of Component class

|  |  |
| --- | --- |
| **Method** | **Description** |
| public void add(Component c) | inserts a component on this component. |
| public void setSize(int width,int height) | sets the size (width and height) of the component. |
| public void setLayout(LayoutManager m) | defines the layout manager for the component. |
| public void setVisible(boolean status) | changes the visibility of the component, by default false. |

### Java AWT Example

To create simple awt example, you need a frame. There are two ways to create a frame in AWT.

* By extending Frame class (inheritance)
* By creating the object of Frame class (association)

## AWT Example by Inheritance

Let's see a simple example of AWT where we are inheriting Frame class. Here, we are showing Button component on the Frame.

**import** java.awt.\*;

**class** First **extends** Frame{

First(){

Button b=**new** Button("click me");

b.setBounds(30,100,80,30);// setting button position

add(b);//adding button into frame

setSize(300,300);//frame size 300 width and 300 height

setLayout(**null**);//no layout manager

setVisible(**true**);//now frame will be visible, by default not visible

}

**public** **static** **void** main(String args[]){

First f=**new** First();

}}

[download this example](https://static.javatpoint.com/src/awt/first.zip)

The setBounds(int xaxis, int yaxis, int width, int height) method is used in the above example that sets the position of the awt button.



## AWT Example by Association

Let's see a simple example of AWT where we are creating instance of Frame class. Here, we are showing Button component on the Frame.

**import** java.awt.\*;

**class** First2{

First2(){

Frame f=**new** Frame();

Button b=**new** Button("click me");

b.setBounds(30,50,80,30);

f.add(b);

f.setSize(300,300);

f.setLayout(**null**);

f.setVisible(**true**);

}

**public** **static** **void** main(String args[]){

First2 f=**new** First2();

}}

[download this example](https://static.javatpoint.com/src/awt/first2.zip)



A component is an object having a graphical representation that can be displayed on the screen and that can interact with the user. Examples of components are the buttons, checkboxes, and scrollbars of a typical graphical user interface.

The Component class is the abstract superclass of the nonmenu-related Abstract Window Toolkit components. Class Component can also be extended directly to create a lightweight component. A lightweight component is a component that is not associated with a native opaque window.

In this section you will learn about the different components available in the Java AWT package for developing user interface for your program.

AWT components that can be used to construct graphical user interfaces (GUIs) for Java. While it is certainly possible to create your own components, or to use those provided by graphical libraries such as Microsoft's AFC, Netscape's IFC, or the upcoming JFC components by Sun, most common user interfaces such as menus, buttons, lists, drop down lists, frames and dialogs are already available - provided by the java.awt package. Even more important is that these user-interface components fit in with the platform on which the applet/application is executed on. Thus a mac user will get a different button to a Windows user, but a button that performs the same functionality. Portability in Java is important, because you never know what type of machine or browser will be used to view your applets.

Following some components of Java AWT are explained :

1. **Labels :** This is the simplest component of Java Abstract Window Toolkit. This component is generally used to show the text or string in your application and label never perform any type of action. Syntax for defining the label only and with justification :   
     
   **Label** label\_name = new **Label** ("This is the label text.");  
   Above code simply represents the text for the label.  
   **Label** label\_name = new **Label** ("This is the label text.", **Label.CENTER**);  
   Justification of label can be left, right or centered. Above declaration used the center justification of the label using the Label.CENTER.
2. **Buttons :** This is the component of Java Abstract Window Toolkit and is used to trigger actions and other events required for your application. The syntax of defining the button is as follows :   
     
   **Button** button\_name = new **Button** ("This is the label of the button.");  
   You can change the Button's label or get the label's text by using the **Button.**setLabel(String) and **Button.**getLabel() method. Buttons are added to the it's container using the add (button\_name) method.
3. **Check Boxes :** This component of Java AWT allows you to create check boxes in your applications. The syntax of the definition of Checkbox is as follows :   
     
   **CheckBox** checkbox\_name = new **Checkbox** ("Optional check box 1", false);  
   Above code constructs the unchecked Checkbox by passing the boolean valued argument *false* with the Checkbox label through the Checkbox() constructor. Defined Checkbox is added to it's container using add (checkbox\_name) method. You can change and get the checkbox's label using the setLabel (String) and getLabel() method. You can also set and get the state of the checkbox using the setState(boolean) and getState() method provided by the **Checkbox** class.
4. **Radio Button :** This is the special case of the Checkbox component of Java AWT package. This is used as a group of checkboxes which group name is same. Only one Checkbox from a Checkbox Group can be selected at a time. Syntax for creating radio buttons is as follows :   
     
   **CheckboxGroup** chkgp = new **CheckboxGroup**();  
   add (new **Checkbox** ("One", chkgp, false);  
   add (new **Checkbox** ("Two", chkgp, false);  
   add (new **Checkbox** ("Three",chkgp, false);  
   In the above code we are making three check boxes with the label "One", "Two" and  "Three". If you mention more than one true valued for checkboxes then your program takes the last true and show the last check box as checked.
5. **Text Area:** This is the text container component of Java AWT package. The Text Area contains plain text. TextArea can be declared as follows:   
     
   **TextArea** txtArea\_name = new **TextArea**();  
   You can make the Text Area editable or not using the setEditable (boolean) method. If you pass the boolean valued argument *false* then the text area will be non-editable otherwise it will be editable. The text area is by default in editable mode. Text are set in the text area using the setText(string) method of the **TextArea** class.
6. **Text Field:** This is also the text container component of Java AWT package. This component contains single line and limited text information. This is declared as follows :   
     
   **TextField** txtfield = new **TextField**(20);  
   You can fix the number of columns in the text field by specifying the number in the constructor. In the above code we have fixed the number of columns to 20.