



Object Oriented Programming

Topic: Graphical User Interface (GUI)

Java AWT

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

Java AWT components are

Platform-dependent i.e. components are displayed according to the view of operating system.

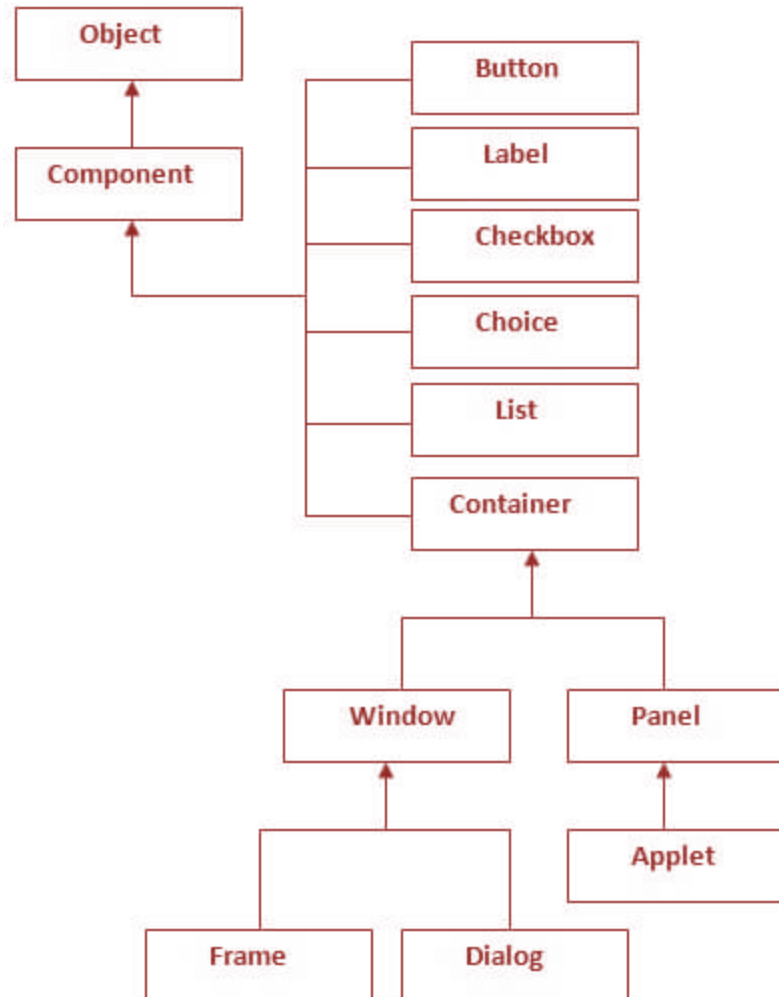
AWT is heavyweight i.e. its components are using the resources of OS.

The java.awt package provides classes for AWT api such as

1. TextField
2. Label
3. TextArea
4. RadioButton
5. CheckBox
6. Choice
7. 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, 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.



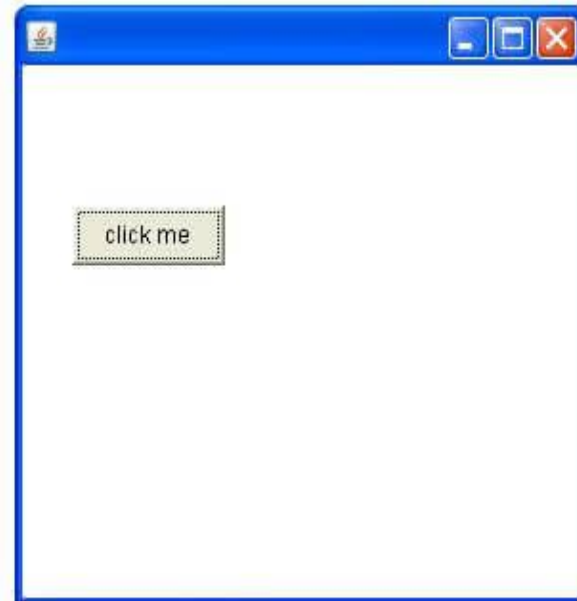
Example

There are two ways to create a frame in AWT

1. By extending Frame class (inheritance)
2. By creating the object of Frame class (association)

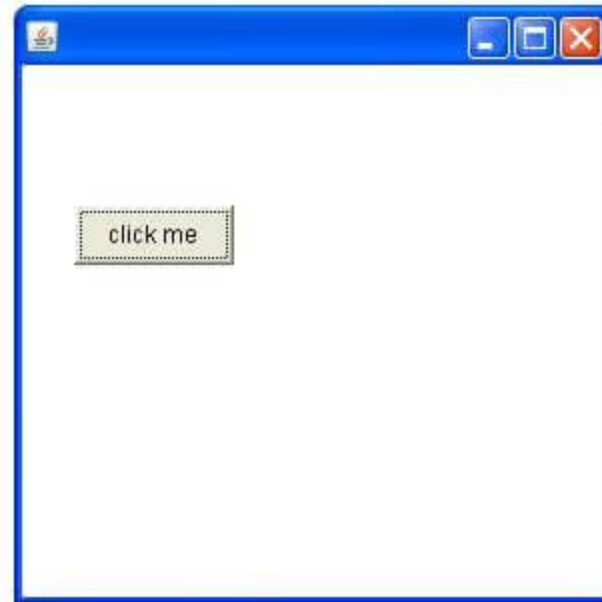
AWT Example by Inheritance

```
import java.awt.*;  
class First extends Frame{  
    First(){  
        Button b=new Button("click me");  
        b.setBounds(30,100,80,30);  
        add(b);  
        setSize(300,300);  
        setLayout(null);  
        setVisible(true);  
    }  
    public static void main(String args[]){  
        First f=new First();  
    }  
}
```



AWT Example by Association

```
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();
    }
}
```



Event and Listener (Java Event Handling)

Changing the state of an object is known as an event. For example, click on button, dragging mouse etc. The java.awt.event package provides many event classes and Listener interfaces for event handling.

Java Event classes and Listener interfaces

Event Classes	Listener Interfaces
ActionEvent	ActionListener
MouseEvent	MouseListener and MouseMotionListener
MouseEvent	MouseWheelListener
KeyEvent	KeyListener
ItemEvent	ItemListener
TextEvent	TextListener
AdjustmentEvent	AdjustmentListener
WindowEvent	WindowListener
ComponentEvent	ComponentListener
ContainerEvent	ContainerListener
FocusEvent	FocusListener

Steps to perform Event Handling

Following steps are required to perform event handling:

1. Register the component with the Listener

Registration Methods

For registering the component with the Listener, many classes provide the registration methods. For example:

Button

```
public void addActionListener(ActionListener a){ }
```

MenuItem

```
public void addActionListener(ActionListener a){ }
```

TextField

```
public void addActionListener(ActionListener a){ }
```

```
public void addTextListener(TextListener a){ }
```

TextArea

```
public void addTextListener(TextListener a){ }
```

Checkbox

```
public void addItemListener(ItemListener a){ }
```

Choice

```
public void addItemListener(ItemListener a){ }
```

List

```
public void addActionListener(ActionListener a){ }
```

```
public void addItemListener(ItemListener a){ }
```

Java ActionListener Interface

The Java ActionListener is notified whenever you click on the button or menu item. It is notified against ActionEvent. The ActionListener interface is found in **java.awt.event package**. It has only one method: **actionPerformed()**.

actionPerformed() method

The actionPerformed() method is invoked automatically whenever you click on the registered component.

```
public abstract void actionPerformed(ActionEvent e);
```



Java Event Handling Code

We can put the event handling code into one of the following places:

1. Within class
2. Other class
3. Anonymous class

Java event handling by implementing ActionListener

```
import java.awt.*;
import java.awt.event.*;
class AEvent extends Frame implements ActionListener
{
    TextField tf;
    AEvent()
    {
        tf=new TextField();
        tf.setBounds(60,50,170,20);
        Button b=new Button("click me");
        b.setBounds(100,120,80,30);
        b.addActionListener(this);
        add(b);add(tf);
        setSize(300,300);
        setLayout(null);
        setVisible(true);
    }
    public void actionPerformed(ActionEvent e)
    {
        tf.setText("Welcome");
    }
    public static void main(String args[]){
        new AEvent();
    }
}
```



Java event handling by outer class

```
import java.awt.*;
import java.awt.event.*;
class AEvent2 extends Frame{
    TextField tf;
    AEvent2(){
        //create components
        tf=new TextField();
        tf.setBounds(60,50,170,20);
        Button b=new Button("click me");
        b.setBounds(100,120,80,30);
        //register listener
        Outer o=new Outer(this);
        b.addActionListener(o);//passing outer class instance
        //add components and set size, layout and visibility
        add(b);add(tf);
        setSize(300,300);
        setLayout(null);
        setVisible(true);
    }
    public static void main(String args[]){
        new AEvent2();
    }
}
```

```
import java.awt.event.*;
class Outer implements ActionListener
{
    AEvent2 obj;
    Outer(AEvent2 obj){
        this.obj=obj;
    }
    public void actionPerformed(ActionEvent e)
    {
        obj.tf.setText("welcome");
    }
}
```

Java event handling by anonymous class

```
import java.awt.*;
import java.awt.event.*;
class AEvent3 extends Frame{
    TextField tf;
    AEvent3(){
        tf=new TextField();
        tf.setBounds(60,50,170,20);
        Button b=new Button("click me");
        b.setBounds(50,120,80,30);
        b.addActionListener(new ActionListener(){
            public void actionPerformed(){
                tf.setText("hello");
            }
        });
        add(b);add(tf);
        setSize(300,300);
        setLayout(null);
        setVisible(true);
    }
    public static void main(String args[]){
        new AEvent3();
    }
}
```



Java Swing

Java Swing is a part of Java Foundation Classes (JFC) that is *used to create window-based applications*. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java.

Unlike AWT, Java Swing provides platform-independent and lightweight components.

The javax.swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc.

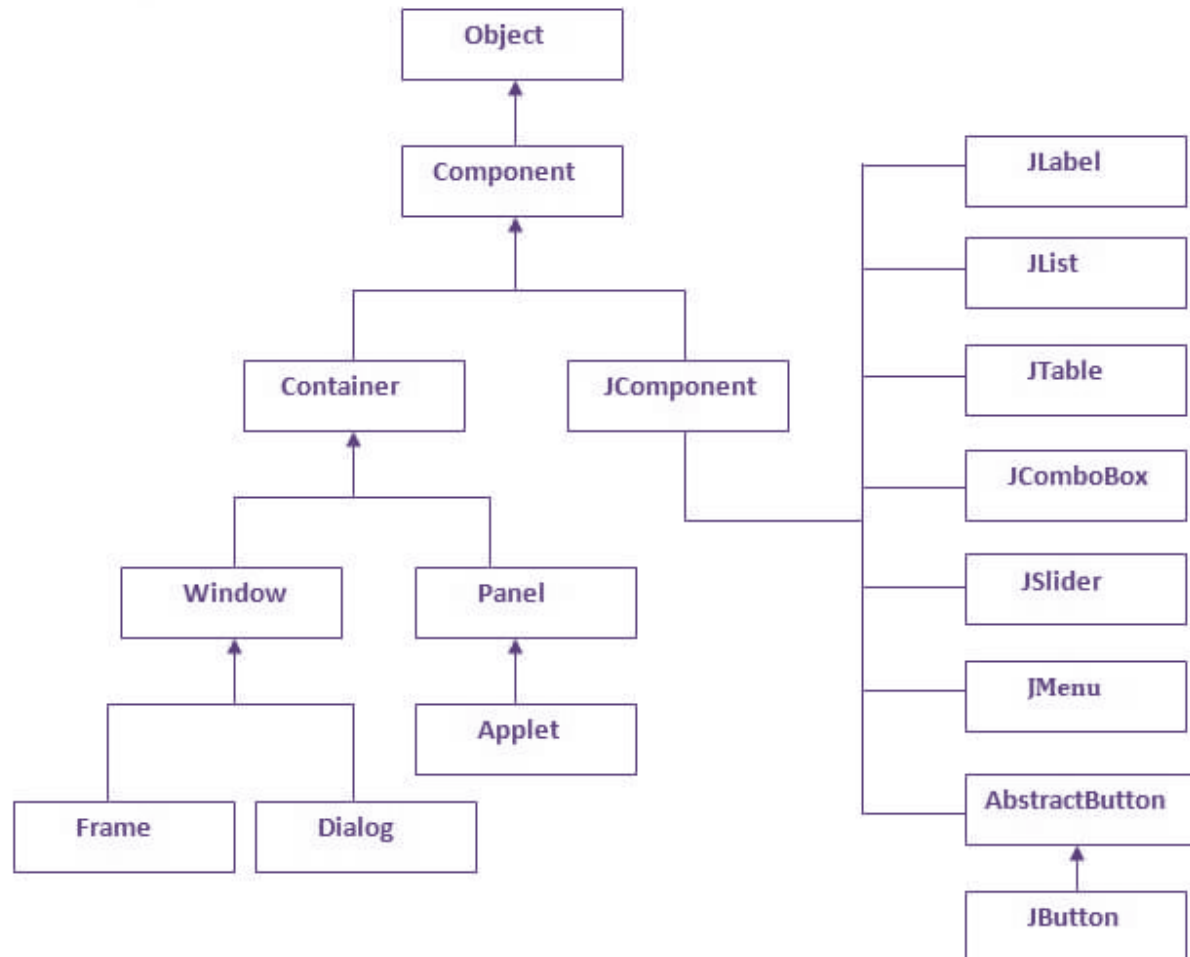
Difference between AWT and Swing

There are many differences between java awt and swing that are given below.	
Java AWT	Java Swing
AWT components are platform-dependent.	Java swing components are platform-independent.
AWT components are heavyweight.	Swing components are lightweight.
AWT doesn't support pluggable look and feel.	Swing supports pluggable look and feel.
AWT provides less components than Swing.	Swing provides more powerful components such as tables, lists, scrollpanes, colorchooser, tabbedpane etc.

JFC

The Java Foundation Classes (JFC) are a set of GUI components which simplify the development of desktop applications.

The hierarchy of java swing API is given below.



FirstSwingExample.java

```
import javax.swing.*;  
public class FirstSwingExample {  
    public static void main(String[] args) {  
        JFrame f=new JFrame();//creating instance of JFrame  
  
        JButton b=new JButton("click");//creating instance of JButton  
        b.setBounds(130,100,100, 40);//x axis, y axis, width, height  
  
        f.add(b);//adding button in JFrame  
  
        f.setSize(400,500);//400 width and 500 height  
        f.setLayout(null);//using no layout managers  
        f.setVisible(true);//making the frame visible  
    }  
}
```

Java JButton

The JButton class is used to create a labeled button that has platform independent implementation. The application result in some action when the button is pushed. It inherits AbstractButton class.

JButton class declaration

```
public class JButton extends AbstractButton implements Accessible
```

Commonly used Constructors:

Constructor	Description
JButton()	It creates a button with no text and icon.
JButton(String s)	It creates a button with the specified text.
JButton(Icon i)	It creates a button with the specified icon object.

Commonly used Methods of AbstractButton class

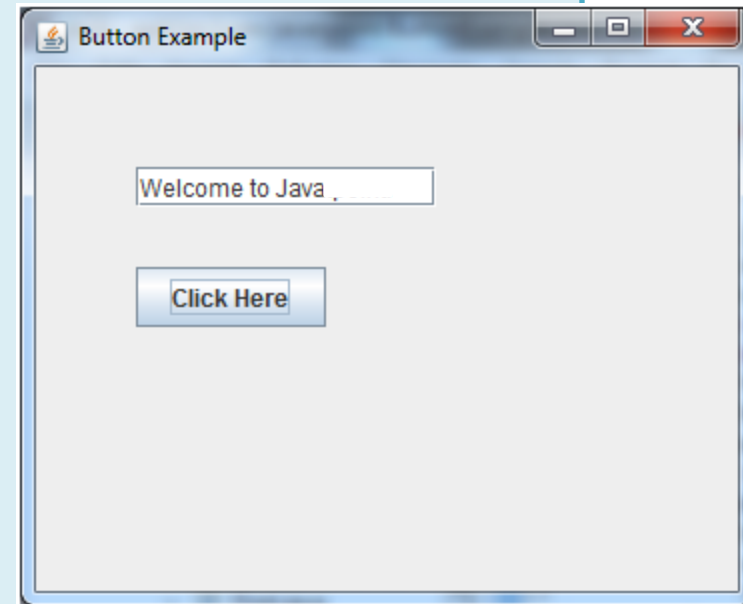
Methods	Description
<code>void setText(String s)</code>	It is used to set specified text on button
<code>String getText()</code>	It is used to return the text of the button.
<code>void setEnabled(boolean b)</code>	It is used to enable or disable the button.
<code>void setIcon(Icon b)</code>	It is used to set the specified Icon on the button.
<code>Icon getIcon()</code>	It is used to get the Icon of the button.
<code>void setMnemonic(int a)</code>	It is used to set the mnemonic on the button.
<code>void addActionListener(ActionListener a)</code>	It is used to add the action listener to this object.

Example1

```
import javax.swing.*;
public class ButtonExample {
public static void main(String[] args) {
    JFrame f=new JFrame("Button Example");
    JButton b=new JButton("Click Here");
    b.setBounds(50,100,95,30);
    f.add(b);
    f.setSize(400,400);
    f.setLayout(null);
    f.setVisible(true);
}
}
```

Example2

```
import java.awt.event.*;
import javax.swing.*.*;
public class ButtonExample {
    public static void main(String[] args) {
        JFrame f=new JFrame("Button Example");
        final JTextField tf=new JTextField();
        tf.setBounds(50,50, 150,20);
        JButton b=new JButton("Click Here");
        b.setBounds(50,100,95,30);
        b.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent e){
                tf.setText("Welcome to Java.");
            }
        });
        f.add(b);f.add(tf);
        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);
    }
}
```



Example3

```
import javax.swing.*;
public class ButtonExample{
    ButtonExample(){
        JFrame f=new JFrame("Button Example");
        JButton b=new JButton(new ImageIcon("D:\\icon.png"));
        b.setBounds(100,100,100, 40);
        f.add(b);
        f.setSize(300,400);
        f.setLayout(null);
        f.setVisible(true);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
    public static void main(String[] args) {
        new ButtonExample();
    }
}
```

