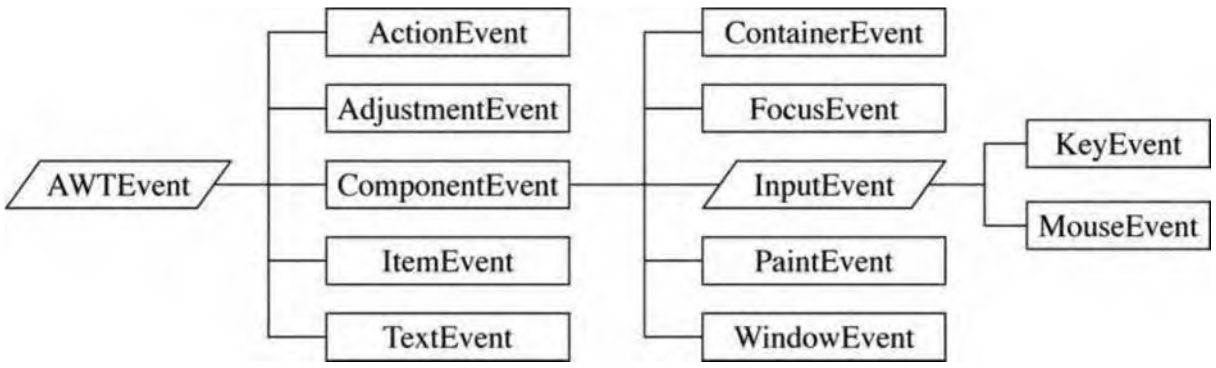
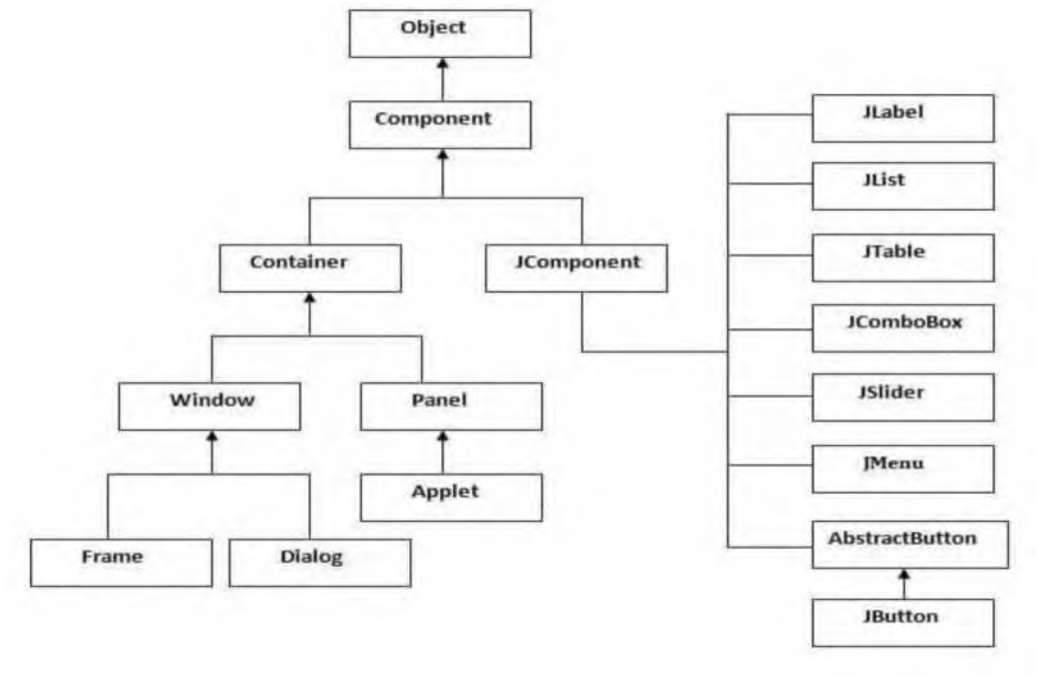
**AWT EVENT HIERARCHY**



**Swing**

- Java Swing tutorial 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.

The hierarchy of java swing API is given below



## Difference between AWT and Swing

There are many differences between java awt and swing that are given below.

| No. | Java AWT | Java Swing |
|---|---|---|
| 1) | AWT components are **platform dependent**. | Java swing components are **platform-independent**. |
| 2) | AWT components are **heavyweight**. | Swing components are **lightweight**. |
| 3) | AWT **doesn't support pluggable look and feel**. | Swing **supports pluggable look and feel**. |
| 4) | AWT provides **less components** than Swing. | Swing provides **more powerful components** such as tables, lists, scrollpanes, colorchooser, tabbedpane etc. |
| 5) | AWT **doesn't follows MVC**(Model View Controller) where model represents data, view represents presentation and controller acts as an interface between model and view. | Swing **follows MVC**. |

**Layout management**

**Java LayoutManagers**

The LayoutManagers are used to arrange components in a particular manner. LayoutManager is an interface that is implemented by all the classes of layout managers. There are following classes that represents the layout managers:

AWT Layout Manager Classes

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

| **Sr.No.** | **LayoutManager & Description** |
|---|---|
| 1 | **BorderLayout**<br>The borderlayout arranges the components to fit in the five regions: east, west, north, south, and center. |
| 2 | **CardLayout**<br>The CardLayout object treats each component in the container as a card. Only one card is visible at a time. |
| 3 | **FlowLayout**<br>The FlowLayout is the default layout. It layout the components in a directional flow. |
| 4 | **GridLayout**<br>The GridLayout manages the components in the form of a rectangular grid. |
| 5 | **GridBagLayout**<br>This is the most flexible layout manager class. The object of GridBagLayout aligns the component vertically, horizontally, or along their baseline without requiring the components of the same size. |
| 6 | **GroupLayout**<br>The GroupLayout hierarchically groups the components in order to position them in a Container. |
| 7 | **SpringLayout**<br>A SpringLayout positions the children of its associated container according to a set of constraints. |
| 8 | **BoxLayout**<br>The BoxLayout is used to arrange the components either vertically or horizontally. |
| 9 | **ScrollPaneLayout**<br>The layout manager used by JScrollPane. JScrollPaneLayout is responsible for nine components: a viewport, two scrollbars, a row header, a column header, and four "corner" components. |

**Border layout:**

**Example**:

```
import java.awt.*;
import javax.swing.*;
  public class Border {
JFrame f;
Border(){
    f=new JFrame();
        JButton b1=new JButton("NORTH");;
    JButton b2=new JButton("SOUTH");;
    JButton b3=new JButton("EAST");;
    JButton b4=new JButton("WEST");;
    JButton b5=new JButton("CENTER");;
          f.add(b 1 ,BorderLayout.NORTH);
    f.add(b2,BorderLayout.SOUTH);
    f.add(b3,BorderLayout.EAST);
    f.add(b4,BorderLayout.WEST);
    f. add(b5,B orderLayout.CENTER);
      f.setSize(300,300);
    f.setVisible(true);
}
public static void main(String[] args) { new Border();
} }
```

**ScrollPaneLayout:**

```
import j avax. swing.ImageIcon;
import j avax. swing.JFrame;
import javax.swing.JLabel;
import j avax. swing.J ScrollPane;
public class ScrollPaneDemo extends JFrame
{
public ScrollPaneDemo() { super("ScrollPane Demo");
Imageicon img = new ImageIcon("child.png");
```

```
  JScrollPane png = new JScrollPane(new JLabel(img));
  getContentPane().add(png);
setSize(300,250);
setVisible(true);
}
  public static void main(String[] args) {
new ScrollPaneDemo();
} }
```

**Boxlayout**

```
import java.awt.*;
import javax.swing.*;
  public class BoxLayoutExamplel extends Frame {
Button buttons[];
   public BoxLayoutExamplel () {
   buttons = new Button [5];
      for (int i = 0;i<5;i++) {
      buttons[i] = new Button ("Button " + (i + 1));
      add (buttons[i]);
   }
  setLayout (new BoxLayout (this, BoxLayout.Y_AXIS));
setSize(400,400);
setVisible(true);
}
  public static void main(String args[]){
BoxLayoutExamplel b=new BoxLayoutExample1();
}
```

**Group layout:**

**Example**

```
public class GroupExample
{
    public static void main(String[] args)
{
        JFrame frame = new JFrame("GroupLayoutExample");
        frame. setD efaultClo seOp eration(JF rame.EXIT_ON_CLO SE);
        Container contentPanel = frame.getContentPane();
        GroupLayout groupLayout = new GroupLayout(contentPanel);
         contentPanel. setLayout(groupLayout);
         JLabel clickMe = new JLabel("Click Here");
        JButton button = new JButton("This Button");
         groupLayout. setHorizontalGroup(
                    groupLayout.createSequentialGroup()
                            .addC omponent(cli ckMe)
                            .addGap(10, 20, 100)
                            .addC omponent(button));
        groupLayout. setVerticalGroup(
                     groupLayout.createParallelGroup(GroupLayout.Alignment.BASELINE)
                            .addC omponent(cli ckMe)
                            .addC omponent(button));
        frame.pack();
        frame. setVisible(true);
    }   } }
```

**Swing components:**

**Text Fields**

The object of a JTextField class is a text component that allows the editing of a single line text. It inherits JTextComponent class.

**Text Areas**

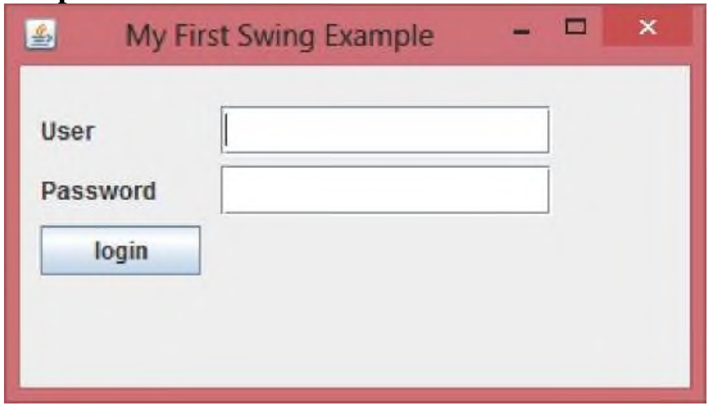
The object of a JTextArea class is a multi line region that displays text. It allows the editing of multiple line text. It inherits JTextComponent class

**Buttons**

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.

```
import j avax. swing.JButton;
import j avax. swing.JFrame;
import j avax. swi ng. JLab el;
import j avax. swi ng. JPanel;
import j avax. swi ng. JPasswordF ield;
import javax.swing.JTextField;
public class SwingFirstExample {
        public static void main(String[] args) {
        // Creating instance of JFrame
        JFrame frame = new JFrame("My First Swing Example");
        // Setting the width and height of frame
        frame.setSize(350, 200);
        frame. setD efaultClo seOp eration(JF rame.EXIT_ON_CLO SE);
        /* Creating panel. This is same as a div tag in HTML
         *   We can create several panels and add them to specific
         *   positions in a JFrame. Inside panels we can add text
         *   fields, buttons and other components.
         *   /
        JPanel panel = new JPanel();
        // adding panel to frame frame.add(panel);
        /* calling user defined method for adding components
         * to the panel.
         */
        placeComponents(panel);
        // Setting the frame visibility to true
        frame.setVisible(true); }
    private static void placeComponents(JPanel panel) {
        /* We will discuss about layouts in the later sections          * of this tutorial. For now we are setting the
layout        * to null         */
        panel.setLayout(null);
        // Creating JLabel
        JLabel userLabel = new JLabel("User");
        /* This method specifies the location and size
         *   of component. setBounds(x, y, width, height)
         *   here (x,y) are cordinates from the top left
         *   corner and remaining two arguments are the width
         *   and height of the component.
         *   /
        userLabel.setBounds(10,20,80,25);
        panel.add(userLabel);
        /* Creating text field where user is supposed to
         *   enter user name.
         *   /
        JTextField userText = new JTextField(20);
        userText. setBounds(100,20,165,25);
        panel.add(userText);
        // Same process for password label and text field.
        JLabel passwordLabel = new JLabel("Password");
        passwordLabel.setBounds(10,50,80,25);
        panel .add(passwordLab el);
        /*This is similar to text field but it hides the user
         *   entered data and displays dots instead to protect
         *   the password like we normally see on login screens.
         *   /
        JPasswordField passwordText = new JPasswordField(20); passwordText. setBounds(100,50,165,25);
        panel .add(passwordT ext);
        // Creating login button
        JButton loginButton = new JButton("login");
        loginButton.setBounds(10, 80, 80, 25);
        panel.add(loginButton);
    }}
```

**Output**:



**Check Boxes**

The JCheckBox class is used to create a checkbox. It is used to turn an option on (true) or off (false). Clicking on a CheckBox changes its state from "on" to "off" or from "off" to "on ".It inherits JToggleButton class.

**Example:**

```
import javax.swing.*;
public class CheckBoxExample
{
    CheckBoxExample(){
      JFrame f= new JFrame("CheckBox Example");
      JCheckBox checkBox1 = new JCheckBox("C++");
      checkBox1.setBounds(100,100, 50,50);
      JCheckBox checkBox2 = new JCheckBox("Java", true);
      checkBox2.setBounds(100,150, 50,50);
      f.add(checkBox1);
      f.add(checkBox2);
      f.setSize(400,400);
      f.setLayout(null);
      f.setVisible(true);
    }
public static void main(String args[])
   {
   new CheckBoxExample();
   }}
```

**Radio Buttons**

The JRadioButton class is used to create a radio button. It is used to choose one option from multiple options. It is widely used in exam systems or quiz. It should be added in ButtonGroup to select one radio button only.

```
import javax.swing.*;
import java.awt.event.*;
class RadioButtonExample extends JFrame implements ActionListener{
```

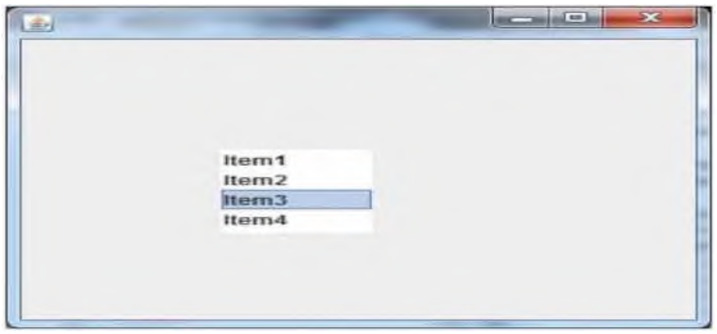
```
JRadioButton rb1,rb2;
JButton b;
RadioButtonExample(){
rb1=new JRadioButton("Male");
rb1.setBounds(100,50,100,30);
rb2=new JRadioButton("Female");
rb2.setBounds(100,100,100,30);
ButtonGroup bg=new ButtonGroup();
bg.add(rb1);bg.add(rb2);
b=new JButton("click");
b.setBounds(100,150,80,30);
b.addActionListener(this);
add(rb1);add(rb2);add(b);
setSize(300,300);
setLayout(null);
setVisible(true);
}
public void actionPerformed(ActionEvent e){
if(rb1.isSelected()){
JOptionPane.showMessageDialog(this,"You are Male.");
}
if(rb2.isSelected()){
JOptionPane.showMessageDialog(this,"You are Female.");
}   }
public static void main(String args[]){
new RadioButtonExample();
}}
```

**Lists**

The object of JList class represents a list of text items. The list of text items can be set up so that the user can choose either one item or multiple items. It inherits JComponent class.

```
import javax.swing.*;
```

```
public class ListExample
{
     ListExample(){
        JFrame f= new JFrame();
        DefaultListModel<String> l1 = new DefaultListModel<>();
         l1.addElement("Item1");
         l1.addElement("Item2");
         l1.addElement("Item3");
         l1.addElement("Item4");
         JList<String> list = new JList<>(l1);
         list.setBounds(100,100, 75,75);
         f.add(list);
         f.setSize(400,400);
         f.setLayout(null);
         f.setVisible(true);
     }
public static void main(String args[])
   {
   new ListExample();
   }}
```



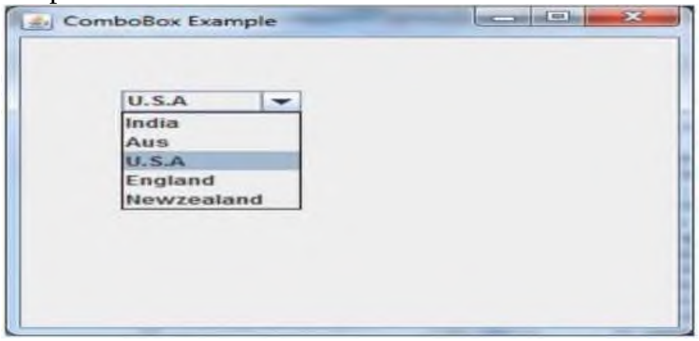
**Choices (JComboBox)**

The object of Choice class is used to show popup menu of choices. Choice selected by user is shown on the top of a menu. It inherits JComponent class.

```
import javax.swing.*;
```

```
public class ComboBoxExample {
JFrame f;
ComboBoxExample(){
   f=new JFrame("ComboBox Example");
   String country[]={"India","Aus","U.S.A","England","Newzealand"};
   JComboBox cb=new JComboBox(country);
   cb.setBounds(50, 50,90,20);
   f.add(cb);
   f.setLayout(null);
   f.setSize(400,500);
   f.setVisible(true);
}
public static void main(String[] args) {
   new ComboBoxExample();
}   }
```

Output:

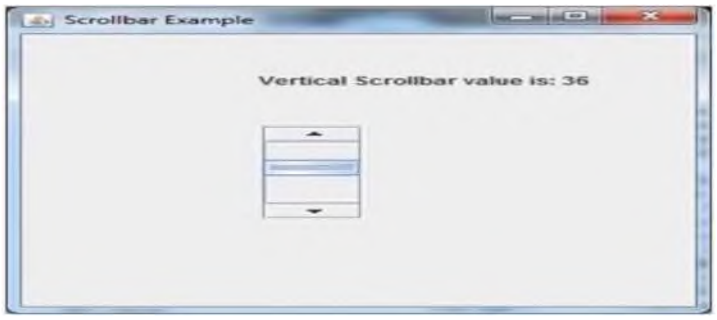


**Scrollbars**

The object of JScrollbar class is used to add horizontal and vertical scrollbar. It is an implementation of a scrollbar. It inherits JComponent class.

```
mport javax.swing.*;
class ScrollBarExample
{
ScrollBarExample(){
   JFrame f= new JFrame("Scrollbar Example");
JScrollBar s=new JScrollBar();
s. setBounds(100,100, 50,100);
f.add(s);
f. setSize(400,400);
f. setLayout(null);
f. setVisible(true);
}
public static void main(String args[])
{
new ScrollBarExample();
}}
```

Output:



**Windows**

The class JWindow is a container that can be displayed but does not have the title bar

**Menus**

The JMenuBar class is used to display menubar on the window or frame. It may have several menus.

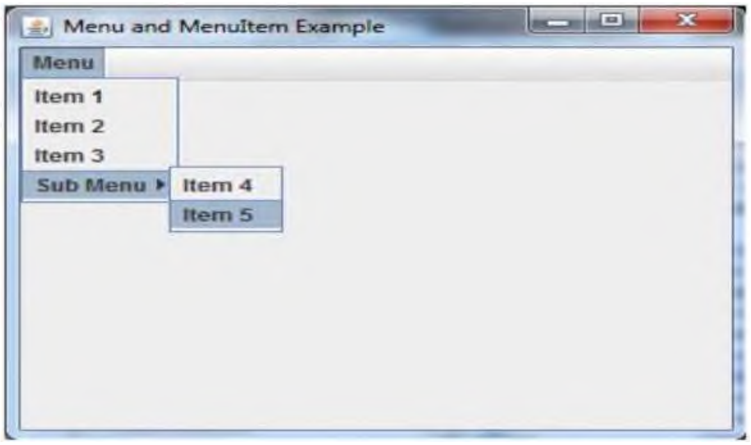
The object of JMenu class is a pull down menu component which is displayed from the menu bar. It inherits the JMenuItem class.

The object of JMenuItem class adds a simple labeled menu item. The items used in a menu must belong to the JMenuItem or any of its subclass.

```
import javax.swing.*;
class MenuExample
{
    JMenu menu, submenu;
    JMenuItem i1, i2, i3, i4, i5;
    MenuExample(){
    JFrame f= new JFrame("Menu and MenuItem Example");
    JMenuBar mb=new JMenuBar();
    menu=new JMenu("Menu");
    submenu=new JMenu("Sub Menu");
    i1=new JMenuItem("Item 1");
    i2=new JMenuItem("Item 2");
    i3=new JMenuItem("Item 3");
    i4=new JMenuItem("Item 4");
    i5=new JMenuItem("Item 5");
    menu.add(i1); menu.add(i2); menu.add(i3);
    submenu.add(i4); submenu.add(i5);
```

```
            menu.add(submenu);
            mb.add(menu);
            f.setJMenuBar(mb);
            f.setSize(400,400);
            f.setLayout(null);
            f.setVisible(true);
        }
        public static void main(String args[])
        {
        new MenuExample();
        }}
```

**Output** :



**Dialog Boxes.**

The JDialog control represents a top level window with a border and a title used to take some form of input from the user. It inherits the Dialog class.Unlike JFrame, it doesn't have maximize and minimize buttons.

**Example:**

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class DialogExample {
   private static JDialog d;
   DialogExample() {
      JFrame f= new JFrame();
      d = new JDialog(f , "Dialog Example", true); d.setLayout( new FlowLayout() );
      JButton b = new JButton ("OK");
      b.addActionListener ( new ActionListener()

         public void actionPerformed( ActionEvent e )

            DialogExample.d.setVisible(false); }}};
      d.add( new JLabel ("Click button to continue."));
      d.add(b);
      d.setSize(300,300);
      d.setVisible(true);
   }
   public static void main(String args[]) {
      new DialogExample(); }}
```

**Output:**

```
Dialog Example
      ClicK button to continue. OK
```