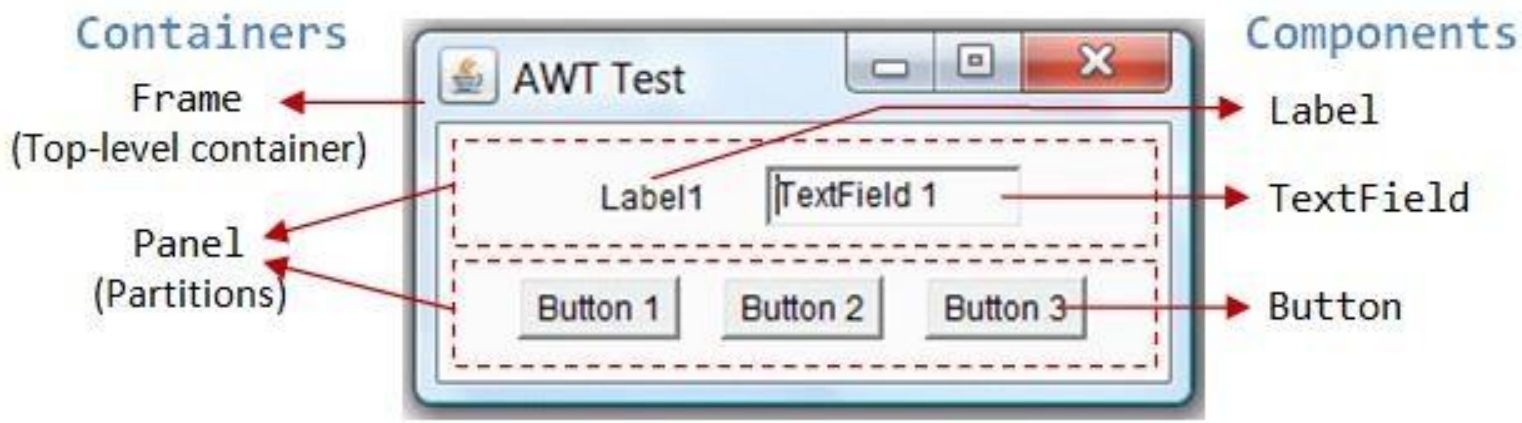
Java GUI: Swing

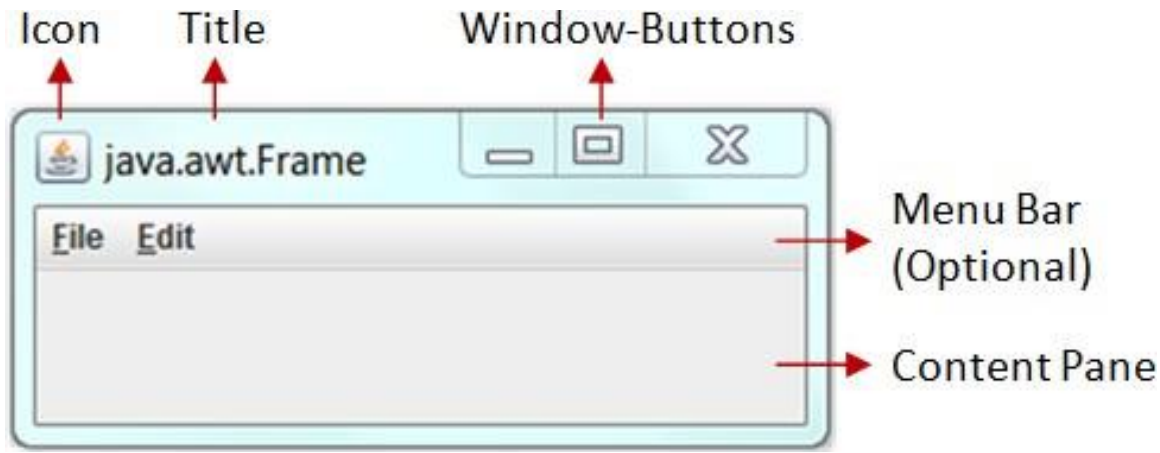
- **Java Swing** is a part of Java Foundation Classes (JFC) that is *used to create window-based applications*. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java.
- Unlike AWT, Java Swing provides platform-independent and lightweight components.
- All of Swing's components are represented by classes defined within the package **javax.swing**
- The javax.swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc.

- ***Container***: The Container is a component 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.

- ***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.


Containers
Frame
(Top-level container)
Panel
(Partitions)
AWT Test
Label1
TextField 1
Button 1
Button 2
Button 3
Components
Label
TextField
Button


Icon
Title
Window-Buttons
java.awt.Frame
File Edit
Menu Bar
(Optional)
Content Pane

Component and Layout Manager in Swing

- **JComponent:** Swing components are derived from the **JComponent** class. **JComponent** inherits the AWT classes **Container** and **Component**. Thus, a Swing component is built on and compatible with an AWT component.
- **Layout Managers:** The layout manager controls the position of components within a container.

Commonly used Methods of Component class

The methods of Component class are widely used in java swing that are given below.

Method	Description
public void add(Component c)	add a component on another component.
public void setSize(int width,int height)	sets size of the component.
public void setLayout(LayoutManager m)	sets the layout manager for the component.
public void setVisible(boolean b)	sets the visibility of the component. It is by default false.

Java Swing Examples

There are two ways to create a frame:

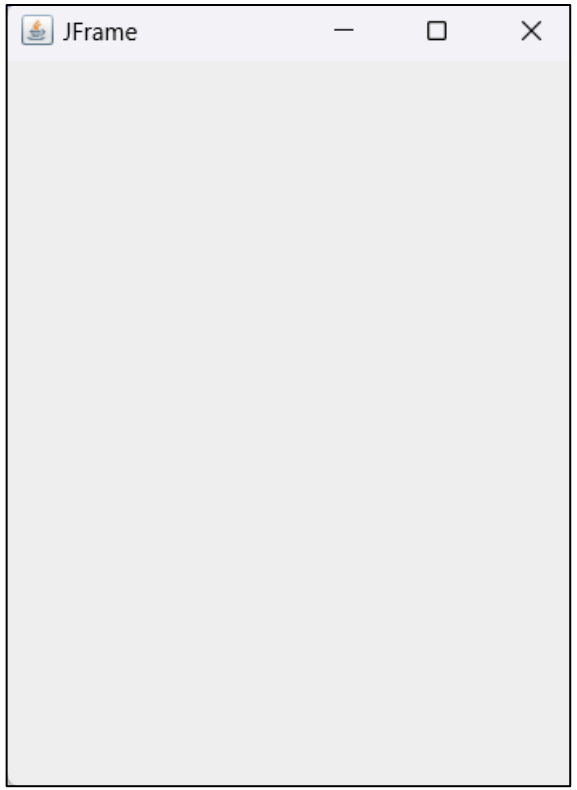
- By creating the object of Frame class
- By extending Frame class

Simple Java Swing Example

Let's see a simple swing example where we are creating a JFrame object

```
import javax.swing.*;
public class Simple{
    JFrame frame;
    public Simple(){
        frame = new JFrame("Button");
        frame.setLayout(null);
        frame.setVisible(true);
        frame.setBounds(150, 150, 300, 400);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
    public static void main(String[] args) {
        Simple b = new Simple();
    }
}
```

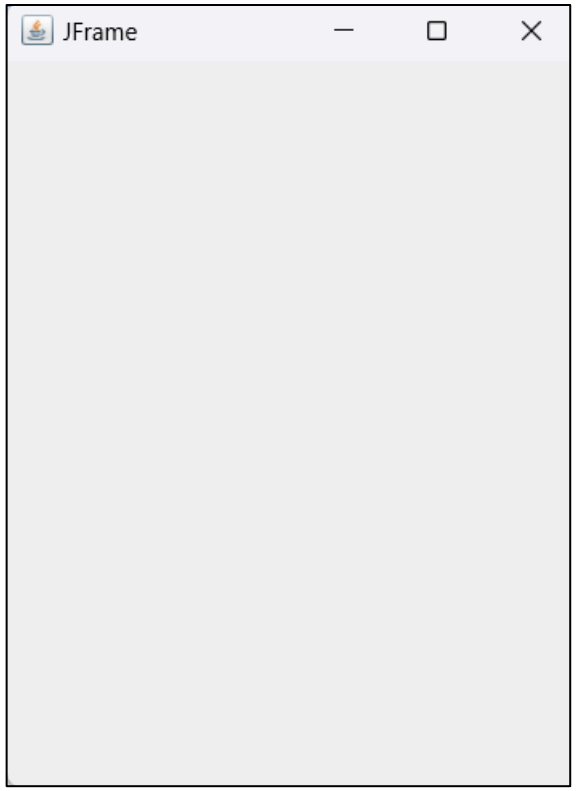
Output:



Simple example of Swing by inheritance

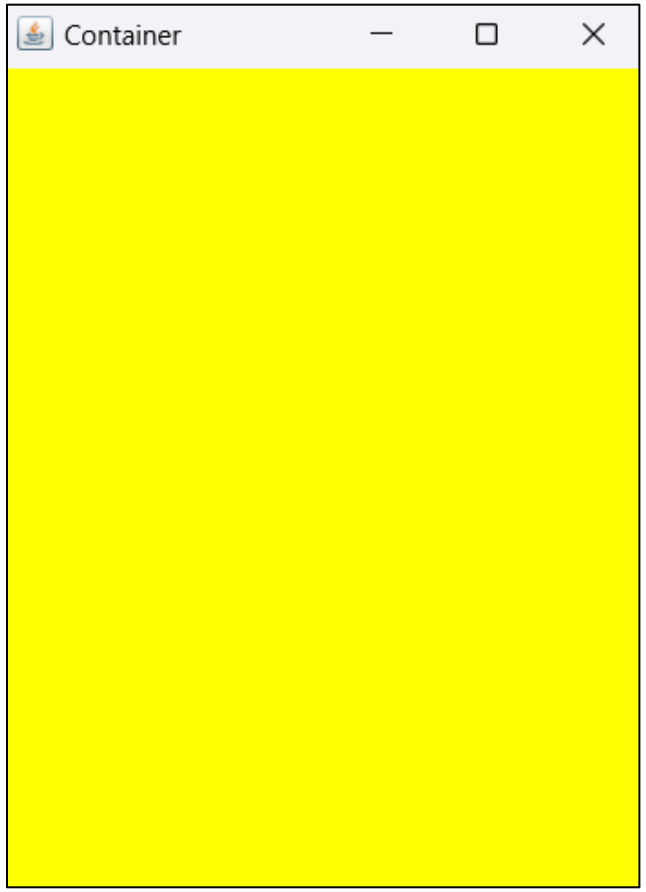
```
import javax.swing.*;

public class Simple2 extends JFrame{
    public Simple2(){
        setTitle("JFrame");
        setLayout(null);
        setVisible(true);
        setBounds(150, 150, 300, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
    public static void main(String[] args) {
        Simple2 b = new Simple2();
    }
}
```

Output:



Java AWT: Container

```
import javax.swing.*;
import java.awt.*;
public class Simple{
   JFrame frame;
   Container c;
   public Simple(){
      frame = new JFrame("Container");
      c = frame.getContentPane();
      c.setLayout(null);
      c.setBackground(Color.yellow);
      frame.setVisible(true);
      frame.setBounds(150, 150, 300, 400);
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  }
   public static void main(String[] args) {
      Simple b = new Simple();
   }
}
```



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.

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

Java JButton Example

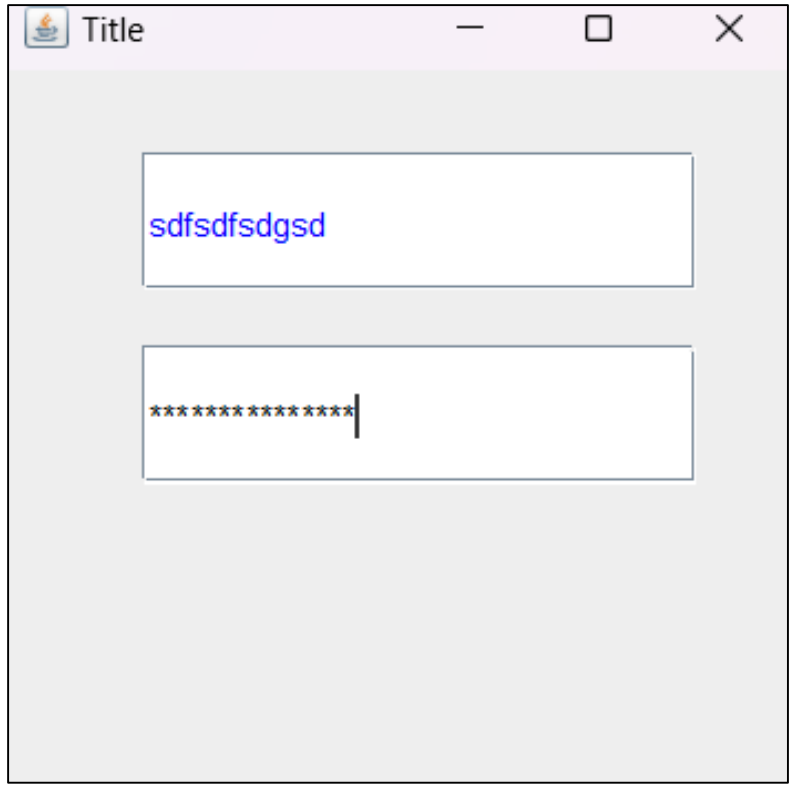
```
import javax.swing.*;
public class Simple{
    JFrame frame;
    JButton btn; Container c;
    public Simple(){
        frame = new JFrame("Button");
        btn = new JButton("Click");
        btn.setBounds(100, 100, 100, 30);
        c = frame.getContentPane();
        c.setLayout(null);
        c.add(btn);
        frame.setVisible(true);
        frame.setBounds(150, 150, 300, 400);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
    public static void main(String[] args) {
        Simple b = new Simple();
    }
}
```

JTextField and JPasswordField

- **JTextField** is a lightweight component that allows the editing of a single line of text.
- **JPasswordField** is a lightweight component that allows the editing of a single line of text where the view indicates something was typed, but does not show the original characters.

Java swing: JTextField & JPasswordField

```
JTextField text = new JTextField();
text.setText(“type here”);
text.setBounds(50, 30, 100, 20);
text.setForeground(Color.blue);
text.setBackground(Color.white);
text.setEditable(true);

JPasswordField pf = new JPasswordField();
pf.setBounds(50, 100, 200, 50);
pf.setForeground(Color.black);
pf.setBackground(Color.white);
pf.setEchoChar('*');

c.add(text);
c.add(pf);
```

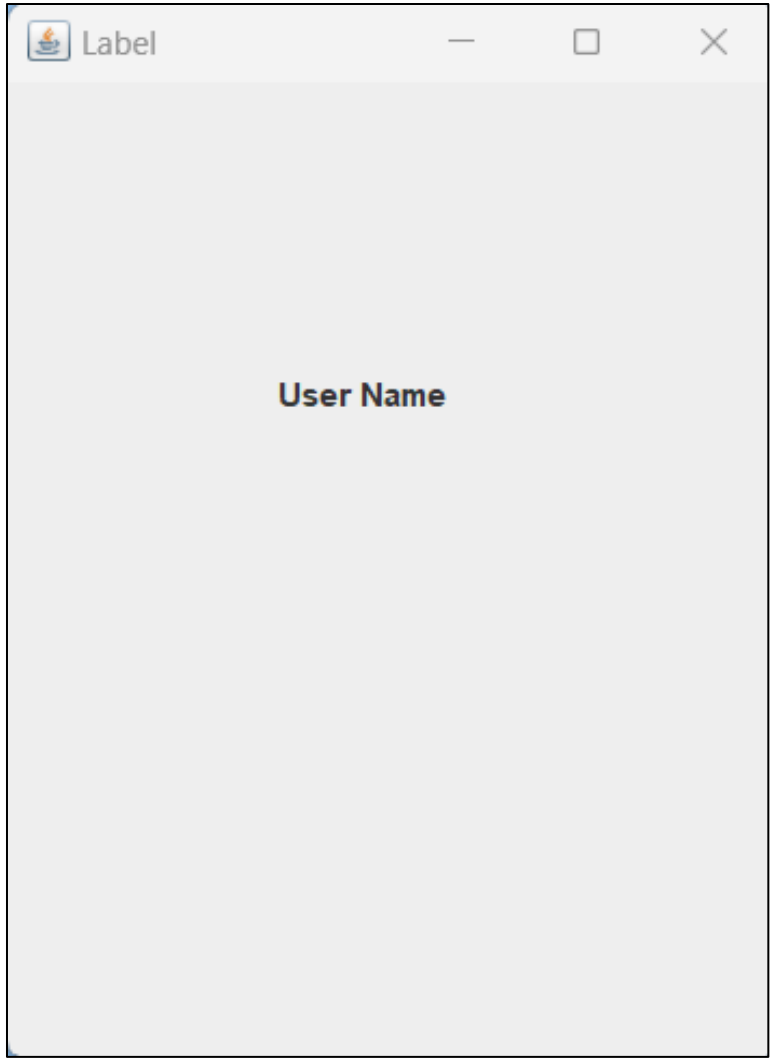


Java Swing: JLabel class

- A JLabel object can display either text, an image, or both.
- It is used to display a single line of read only text.
- The text can be changed by an application but a user cannot edit it directly.

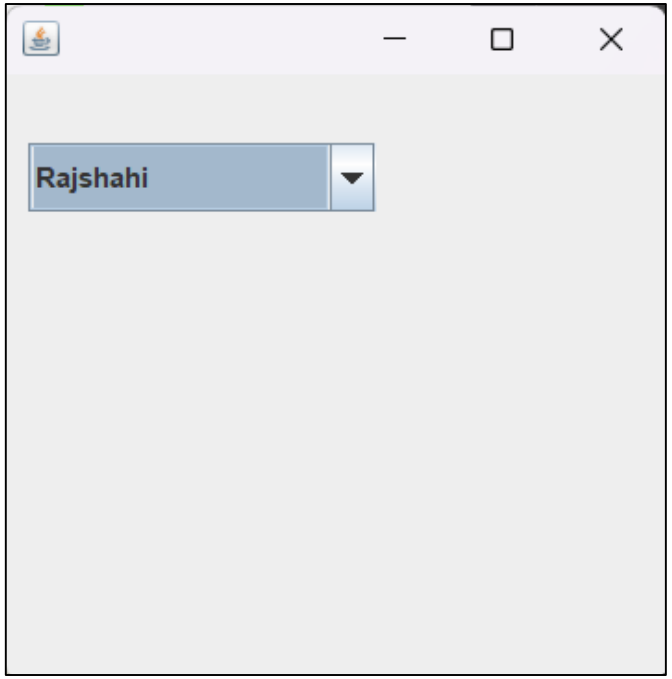
Java swing: JLabel class

```
import javax.swing.*;
public class Simple{
   JFrame frame;
   JLabel L1; Container C;
   public Simple(){
      frame = new JFrame("Label");
      L1 = new JLabel("User Name");
      L1.setBounds(100, 100, 100, 30);
      c = frame.getContentPane();
      c.setLayout(null);
      c.add(L1);
      frame.setVisible(true);
      frame.setBounds(150, 150, 300, 400);
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   }
   public static void main(String[] args) {
      Simple b = new Simple();
   }
}
```



Java Swing: JComboBox class

```
String[] city = {"Rajshahi","Dhaka","Natore","Bogura"};
JComboBox combo = new JComboBox(city);
combo.setSelectedIndex(0);
combo.setBounds(10, 30, 150, 30);
c.add(combo);
```



Event Handling

- Mechanism that controls the event
- Decides what should happen if an event occurs
- Steps involved in event handling:
  - The User clicks the button and the event is generated
  - Object of concerned event class is created automatically
  - Event object is forwarded to the method of registered listener class
  - The method is now get executed and returns

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

How to write ActionListener

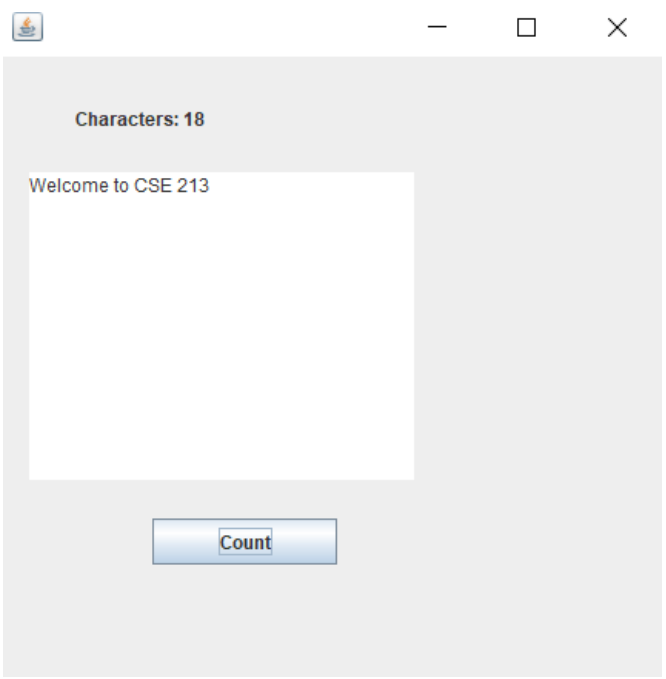
- Implement the ActionListener interface in the class:
  - public class ActionListenerExample Implements ActionListener
- Register the component with the Listener:
  - component.addActionListener(instanceOfListenerclass);
- Override the actionPerformed() method:
  - public void actionPerformed(ActionEvent e){

```
        //Write the code here
}
```

Java JTextArea Example with ActionListener

```
import javax.swing.*;
import java.awt.event.*;
public class TextAreaExample implements ActionListener{
JLabel l1;
JTextArea area;
JButton b;
TextAreaExample() {
    JFrame f= new JFrame();
    l1=new JLabel(“Characters”);
    l1.setBounds(50,25,100,30);
    area=new JTextArea();
    area.setBounds(20,75,250,200);
    b=new JButton("Count");
    b.setBounds(100,300,120,30);
    b.addActionListener(this);
    f.add(l1);f.add(area);f.add(b);
    f.setSize(450,450);
    f.setLayout(null);
    f.setVisible(true);
}
public void actionPerformed(ActionEvent e){
    String text=area.getText();
    l1.setText("Characters: "+text.length());
}
public static void main(String[] args) {
    new TextAreaExample();
}
}
```

Output:



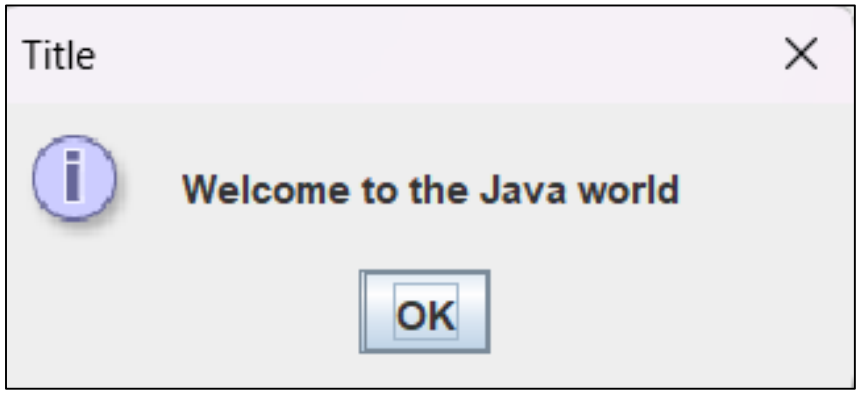
SWING - JOptionPane Class

- **JOptionPane** is a component providing standard methods to pop up a standard dialog box
- It is used to provide standard dialog boxes such as message dialog box, confirm dialog box and input dialog box.

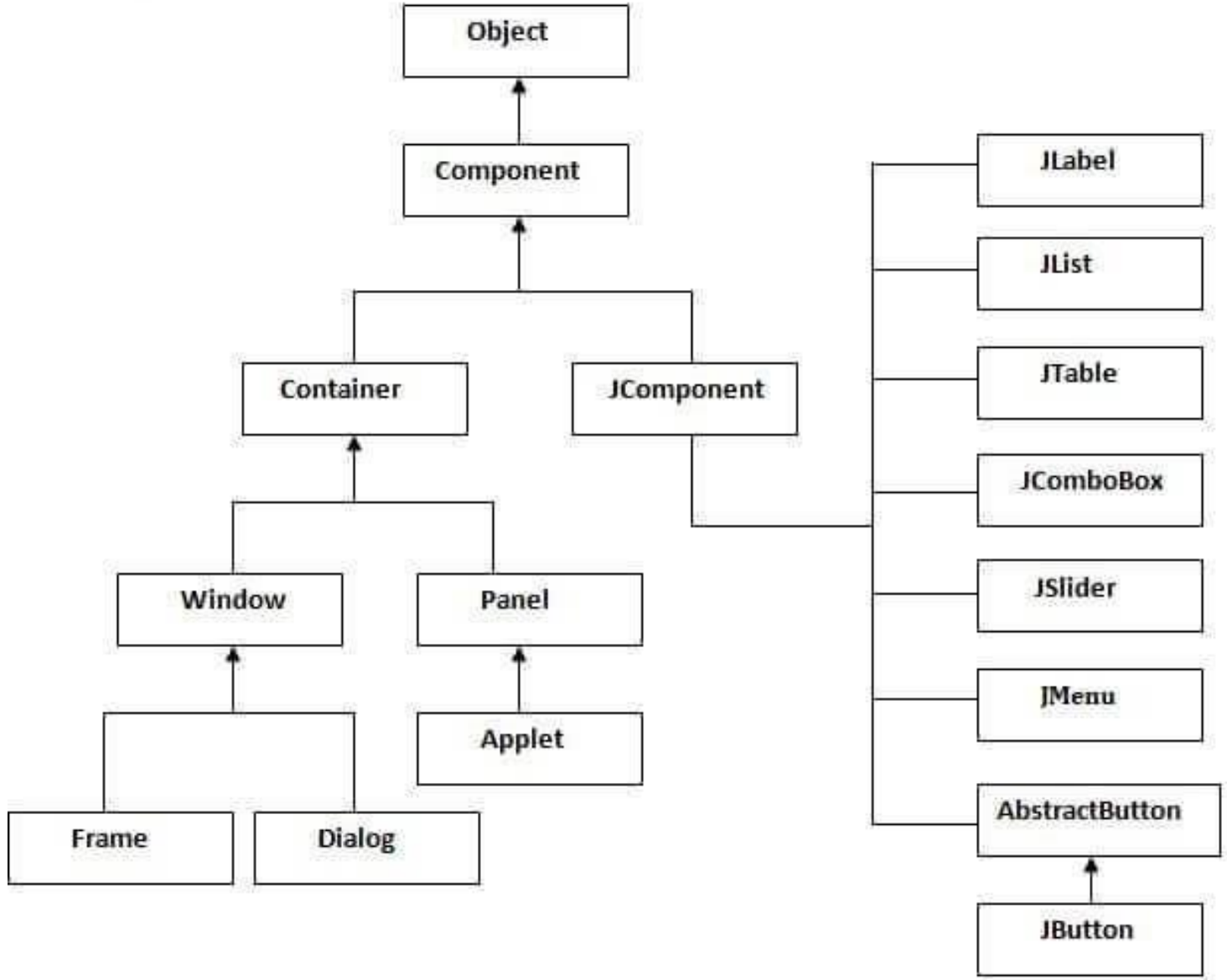
JOptionPane – showMessageDialog()

- It is used to create a message dialog with given title and messageType.

```
import javax.swing.JOptionPane;


public class DialogBox {
    public static void main(String[] args) {
        JOptionPane.showMessageDialog(null, "Welcome to the
Java world", "Title",JOptionPane.INFORMATION_MESSAGE);
    }
}
```



* **Message type can be:** ERROR_MESSAGE, INFORMATION_MESSAGE, WARNING_MESSAGE, QUESTION_MESSAGE, PLAIN_MESSAGE


Object
Component
Container
JComponent
Window
Panel
Applet
Frame
Dialog
JLabel
JList
JTable
JComboBox
JSlider
JMenu
AbstractButton
JButton