Introduction to Java Swing

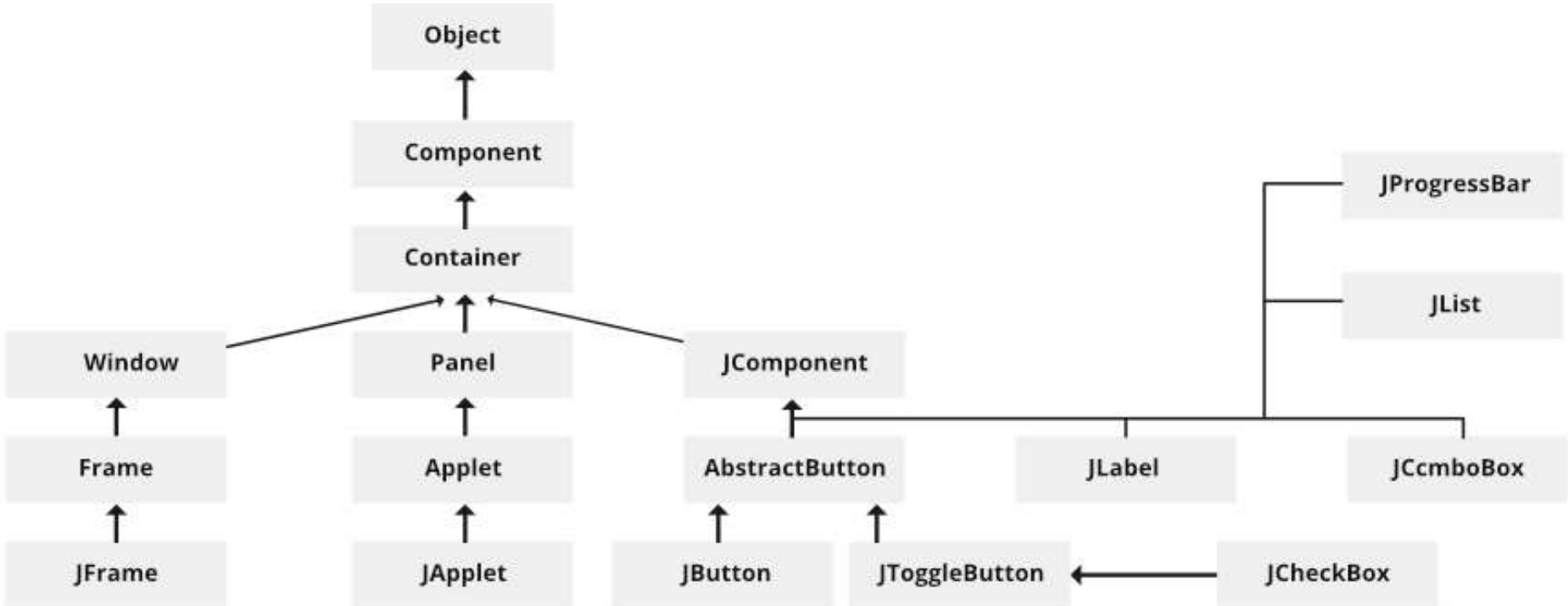
What is Java Swing?

- Java Swing is a graphical user interface (GUI) toolkit and a part of the Java Foundation Classes (JFC).
- It provides a set of components and tools for building desktop applications with a rich and interactive user interface.
- Swing was introduced as an alternative to the earlier Abstract Window Toolkit (AWT) in Java, offering a more extensive and flexible set of GUI components.
- It provides a platform-independent framework for creating windows, dialog boxes, buttons, checkboxes, menus, and many other GUI elements.
- Swing was introduced as an alternative to the earlier Abstract Window Toolkit (AWT) in Java, offering a more extensive and flexible set of GUI components.
- It provides a platform-independent framework for creating windows, dialog boxes, buttons, checkboxes, menus, and many other GUI elements.

How it differs from AWT (Abstract Window Toolkit).

	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**

Swing Components:



Methods of Component class

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.

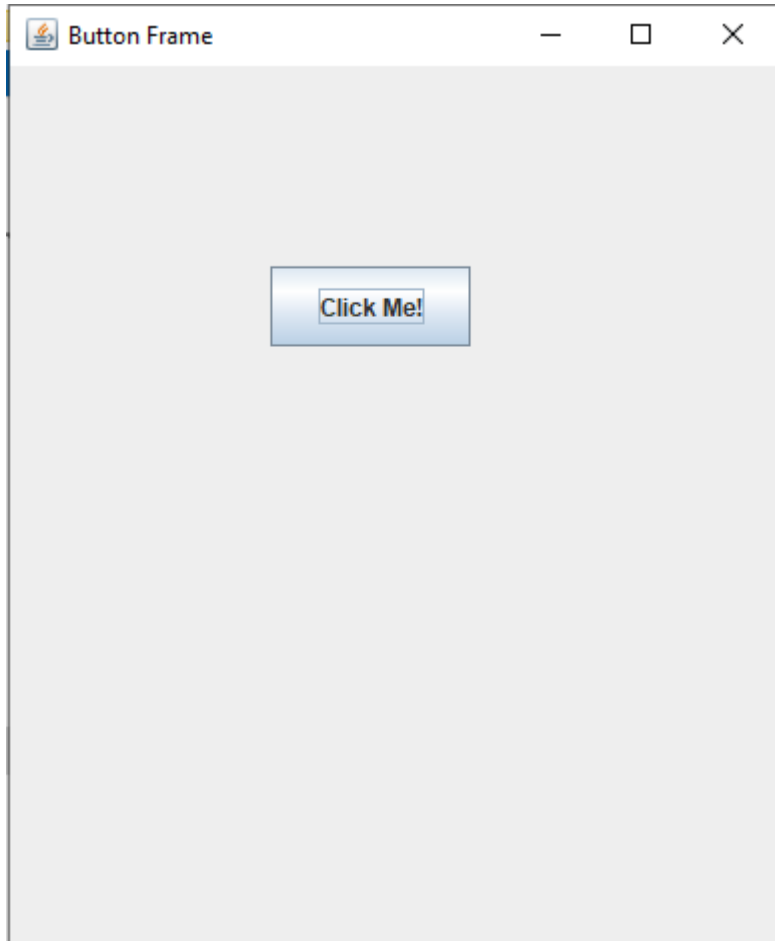
Java Swing Examples

- There are two ways to create a frame:
- By creating the object of Frame class (association)
- By extending Frame class (inheritance)

Simple Java Swing Example

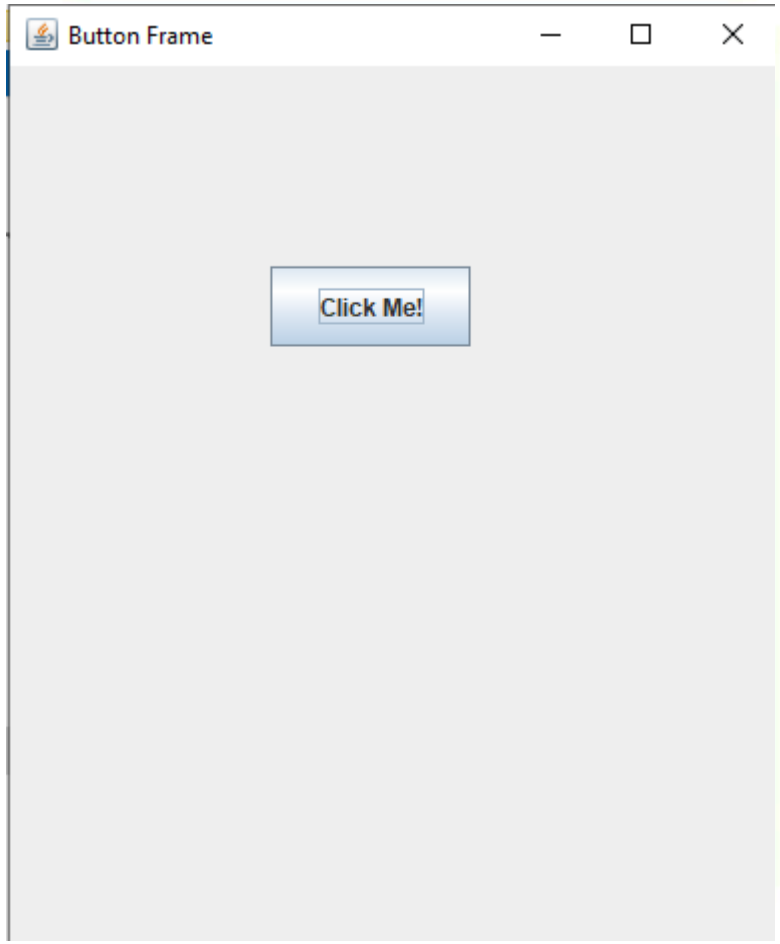
```
import javax.swing.JButton;
import javax.swing.JFrame;

public class ButtonFrame {
   public static void main(String[] args) {
     JFrame frame = new JFrame("Button Frame");
       JButton button = new JButton("Click Me!");
       button.setBounds(130,100,100, 40);
       frame.add(button);
       frame.setSize(400, 500);
      frame.setLayout(null);
       frame.setVisible(true);
   }
}
```



Simple example of Swing by inheritance

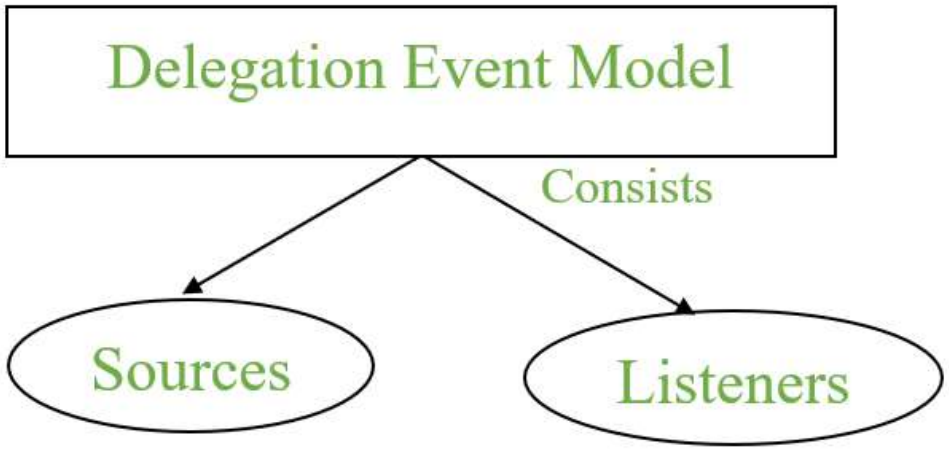
```
import javax.swing.JButton;
import javax.swing.JFrame;
public class ButtonFrame1 extends JFrame {
ButtonFrame1()
{
 JButton button = new JButton("Click Me!");
       button.setBounds(130,100,100, 40);
     add(button);
   setSize(400, 500);
                    setLayout(null);
   setVisible(true);
 }
public static void main(String[] args)
{
       ButtonFrame1 b1=new ButtonFrame1();
}
}
```



Event Handling

- An **event** can be defined as changing the state of an object or behavior by performing actions. Actions can be a button click, cursor movement, keypress through keyboard or page scrolling, etc.
- The **java.awt.event** package can be used to provide various event classes.
- It is a mechanism to **control the events** and to **decide what should happen after an event** occur. To handle the events, Java follows the ***Delegation Event model.***
- It has Sources and Listeners.
- **Source:** Events are generated from the source.

  There are various sources like buttons, checkboxes, list, menu-item, choice, scrollbar, text components, windows, etc., to generate events.
- **Listeners:** Listeners are used for handling the events generated from the source. Each of these listeners represents interfaces that are responsible for handling events.



Event Class	Listener Interface	Description
ActionEvent	ActionListener	An event that indicates that a component-defined action occurred like a button click or selecting an item from the menu-item list.
AdjustmentEvent	AdjustmentListener	The adjustment event is emitted by an Adjustable object like Scrollbar.
ComponentEvent	ComponentListener	An event that indicates that a component moved, the size changed or changed its visibility.
ContainerEvent	ContainerListener	When a component is added to a container (or) removed from it, then this event is generated by a container object.
FocusEvent	FocusListener	These are focus-related events, which include focus, focusin, focusout, and blur.
ItemEvent	ItemListener	An event that indicates whether an item was selected or not.
KeyEvent	KeyListener	An event that occurs due to a sequence of keypresses on the keyboard.
MouseEvent	MouseListener & MouseMotionListener	The events that occur due to the user interaction with the mouse (Pointing Device).
MouseWheelEvent	MouseWheelListener	An event that specifies that the mouse wheel was rotated in a component.
TextEvent	TextListener	An event that occurs when an object's text changes.
WindowEvent	WindowListener	An event which indicates whether a window has changed its status or not.

Listener Interface	Methods
ActionListener	•actionPerformed()
AdjustmentListener	•adjustmentValueChanged()
ComponentListener	•componentResized() •componentShown() •componentMoved() •componentHidden()
ContainerListener	•componentAdded() •componentRemoved()
FocusListener	•focusGained() •focusLost()
ItemListener	•itemStateChanged()
KeyListener	•keyTyped() •keyPressed() •keyReleased()
MouseListener	•mousePressed() •mouseClicked() •mouseEntered() •mouseExited() •mouseReleased()
MouseMotionListener	•mouseMoved() •mouseDragged()
MouseWheelListener	•mouseWheelMoved()
TextListener	•textChanged()
WindowListener	•windowActivated() •windowDeactivated() •windowOpened() •windowClosed() •windowClosing() •windowIconified() •windowDeiconified()

```
import javax.swing.*;
import javax.swing.JFrame;
import java.awt.event.*;
public class ButtonFrameEvent extends JFrame implements ActionListener {
JTextField text;
ButtonFrameEvent()
{                    JButton button = new JButton("Click Me!");
      button.setBounds(130,100,100, 40);


                     text = new JTextField(20);
                     text.setBounds(130,150,100,40);
                     button.addActionListener(this);
                     add(button);
                     add(text);
                     setSize(800, 800);
                     setLayout(null);
                     setVisible(true);


 }
                     public void actionPerformed(ActionEvent e)
                     {
         text.setText("Welcome...");
      }
public static void main(String[] args)
{
      ButtonFrameEvent b1=new ButtonFrameEvent();
}}
```