EVENT HANDLING:

Event:

- all activity that goes on between the user and the application.
- Can be called as action that takes place within a program, often initiated by user interactions or other external factors.
- Plays a vital role in developing GUI applications as it allows to respond to the user input.

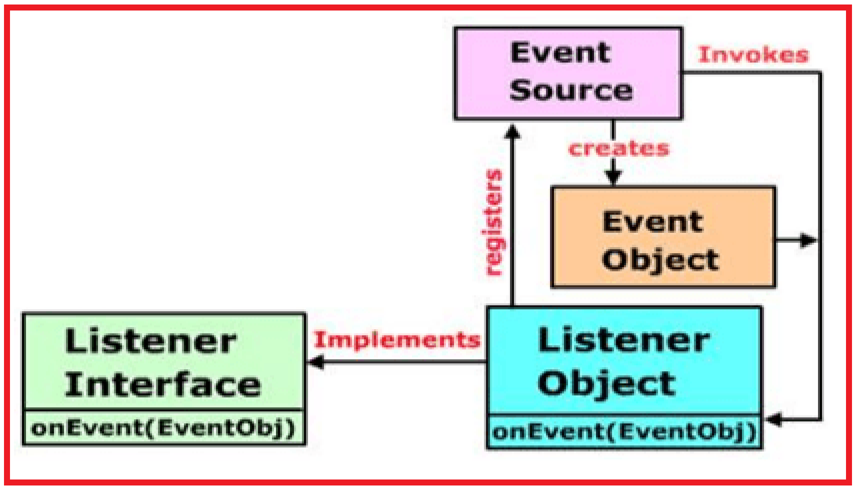
Elements of Event in Java:

Source/Object: The component or object that generates or triggers the event. In Java, this can be a button, menu item, text field, mouse, keyboard, and other.
Event type: Each event has particular type. Example: In GUI programming using Java Swing, event types consists of action events (triggered by button clicks), mouse events (triggered by mouse interactions), key events (triggered by keyboard inputs), etc.
Event Listener: A class or method which listens to an event and uses the code to respond to that event. Event listeners implement interfaces provided by Java libraries, such as ActionListener, MouseListener, KeyListener, etc. These interfaces define methods that need to be implemented to handle events.

Delegation Event Model

- Defines standard way for getting and processing events
- Mechanism:
  - Event Generation:
    - a source generates an event and sends it to listener(s)
    - E.g. Button Click event, etc
  - Event Listen & Handle:

- Listeners waits until some event occurs, once an event is received, the listener processes the event and then returns.
- Implement the interface in the listener so that it will receive the type of event desired .E.g. ActionListener is implemented for handling Button Click event.



Advantages of Delegation Event Model

- The processing of events is separated from the interface logic which generates those events
- An interface element is in a position to “delegate” the processing of an occasion to a separate piece of code.
- In the delegation event model, listeners must register with a source so as to receive an occasional notification:
    - Hence, notifications are sent only to listeners that want to receive them.
    - This is a more efficient way to handle events.

	Events	**Source Object**	**Listener Interface**	**Methods**
1	ActionEvent	Button, List, MenuItem, TextField	ActionListener	ActionPerformed( )
2	AdjustmentEvent	Component	ComponentListener	AdjustmentValueChanged( )

3	FocusEvent	Component	FocusListener	focusGained( )
				focusLost( )

4	TextEvent	Text Component	TextListener	TextChanged( )
5	ItemEvent	Checkbox,choice	ItemListener	ItemStateChanged( )

6	MouseEvent	Mouse Movement	MouseListener	MousePressed( )
				mouseClicked( )
				mouseEntered( )
				mouseExited( )
				mouseReleased( )

7	WindowEvent	Window	WindowListener	windowActivated( )
				windowDeactivated( )
				windowOpened( )
				windowClosed( )
				windowClosing( )

8	KeyEvent	TextComponent	KeyListener	keyTyped( )
				keyReleased()
				keyPressed( )

Example showing Steps to Handle Event- ActionListener
For click event handling in a button

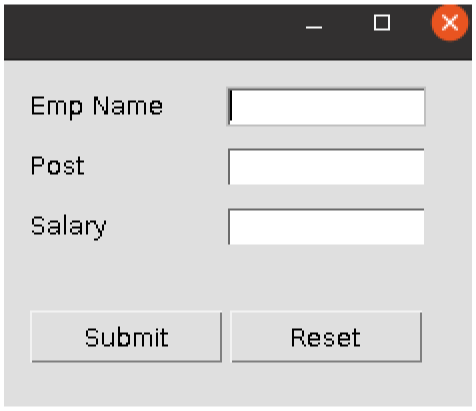
1. **Event Generation:** Whenever the user clicks the button an event is generated.
2. **Object Creation:** Object of the concerned event class will be automatically created and information about the source and the event gets populated within the same object.
3. **Listener Invocation:** Then the event object is forwarded to the method of the registered listener class.
4. **Process Event:** Now the method will get executed and returned.

Programs(Extra example of AWT and Swing Concepts)

```
import java.awt.*;

public class AWTDemo extends Frame {

AWTDemo(){

Label empName = new Label("Emp Name");

empName.setBounds(20, 50, 80, 20);


Label post = new Label("Post");

post.setBounds(20, 80, 80, 20);


Label salary = new Label("Salary");
```

```
salary.setBounds(20, 110, 80, 20);


TextField empNameTF = new TextField();

empNameTF.setBounds(120, 50, 100, 20);


TextField postTF = new TextField();

postTF.setBounds(120, 80, 100, 20);


TextField salaryTF = new TextField();

salaryTF.setBounds(120, 110, 100, 20);


Button sbmt = new Button("Submit");

sbmt.setBounds(20, 160, 100, 30);


Button reset = new Button("Reset");

reset.setBounds(120,160,100,30);


add(empName);

add(post);

add(salary);
add(empNameTF);

add(salaryTF);

add(postTF);

add(sbmt);

add(reset);


setSize(250,250);

setLayout(null);

setVisible(true);

}

public static void main(String[] args) {


AWTDemo demo = new AWTDemo();

}

}
```

Output:

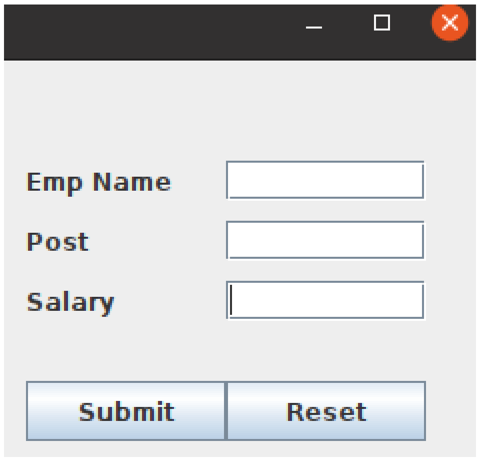


Program:Using Swing:

Using Swing:

```
import javax.swing.*;
public class SwingDemo {
SwingDemo(){
JFrame f = new JFrame();

JLabel empName = new JLabel("Emp Name");
empName.setBounds(20, 50, 80, 20);

JLabel post = new JLabel("Post");
post.setBounds(20, 80, 80, 20);

JLabel salary = new JLabel("Salary");
salary.setBounds(20, 110, 80, 20);

JTextField empNameTF = new JTextField();
empNameTF.setBounds(120, 50, 100, 20);

JTextField postTF = new JTextField();
postTF.setBounds(120, 80, 100, 20);
JTextField salaryTF = new JTextField();
salaryTF.setBounds(120, 110, 100, 20);

JButton sbmt = new JButton("Submit");
sbmt.setBounds(20, 160, 100, 30);

JButton reset = new JButton("Reset");
reset.setBounds(120,160,100,30);

f.add(empName);
f.add(post);
f.add(salary);
f.add(empNameTF);
f.add(postTF);
f.add(salaryTF);
f.add(sbmt);
f.add(reset);

f.setSize(250,250);
f.setLayout(null);
f.setVisible(true);
}

public static void main(String[] args) {

SwingDemo s = new SwingDemo();
}
}
```

Output:

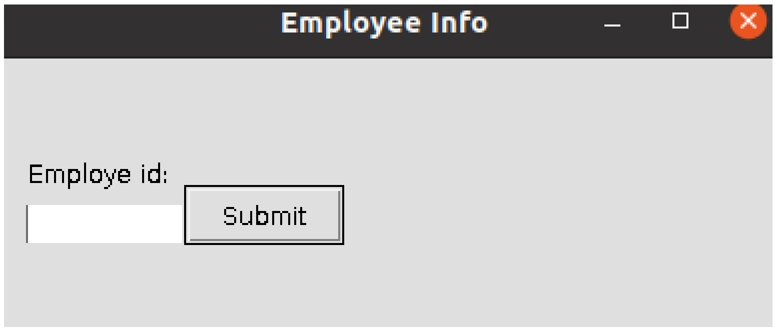


Program :
```
import java.awt.*;

// class AWTExample directly creates instance of Frame class
class AWTExample {

  // initializing using constructor
  AWTExample() {

       // creating a Frame
       Frame f = new Frame("Employee Info");

       // creating a Label
       Label l = new Label("Employe id:");

       // creating a Button
       Button b = new Button("Submit");

       // creating a TextField
       TextField t = new TextField();
```

```
        // setting position of above components in the frame
        l.setBounds(20, 80, 80, 30);
        t.setBounds(20, 100, 80, 30);
        b.setBounds(100, 100, 80, 30);

        // adding components into frame
        f.add(b);
        f.add(l);
        f.add(t);

        // frame size 300 width and 300 height
        f.setSize(400,300);

        // setting the title of frame  Note: either we can use
f.setTitle("Employee info") or we can use Frame f = new
Frame("Employee id:");
        // f.setTitle("Employee info");

        // no layout
        f.setLayout(null);

        // setting visibility of frame
        f.setVisible(true);
}

// main method
public static void main(String args[]) {

// creating instance of Frame class
AWTSecond awtobj = new AWTSecond();

}
```

```
}
```

Output:

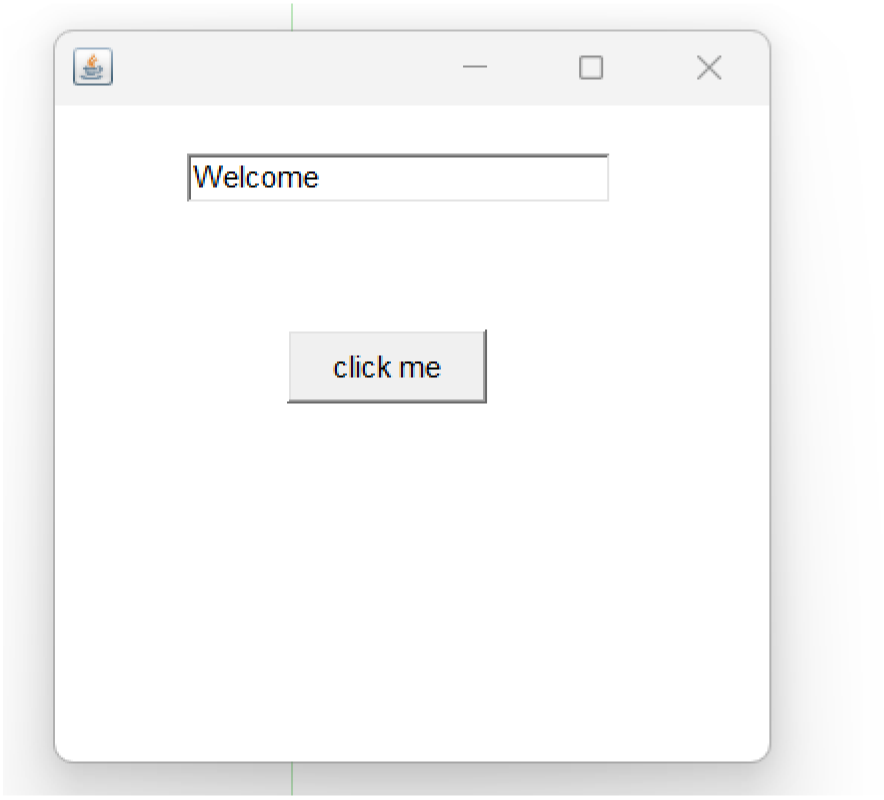


Program 3:

```
import java.awt.*;
import java.awt.event.*;
class AWTThird extends Frame implements ActionListener {
TextField tf;
Button b;
AWTThird() {
//create components
tf=new TextField();
tf.setBounds(60,50,170,20);
b=new Button("click me");
b.setBounds(100,120,80,30);
//register listener
b.addActionListener(this); //passing current instance
//add components and set size, layout and visibility
addWindowListener(new WindowAdapter(){
public void windowClosing(WindowEvent e){
System.exit(0);
}
});
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[]){
AWTThird x=new AWTThird();
}
}
```

Output:

If we click on “Click me “ then the Text will appear as shown below:


Welcome
click me