## What is Event Handling?

Event Handling is the mechanism that controls the event and decides what should happen if an event occurs. This mechanism has a code which is known as an event handler, that is executed when an event occurs.

Java uses the Delegation Event Model to handle the events. This model defines the standard mechanism to generate and handle the events.

The Delegation Event Model has the following key participants.

* **Source** − The source is an object on which the event occurs. Source is responsible for providing information of the occurred event to it's handler. Java provide us with classes for the source object.
* **Listener** − It is also known as event handler. The listener is responsible for generating a response to an event. The listener waits till it receives an event. Once the event is received, the listener processes the event and then returns.

The benefit of this approach is that the user interface logic is completely separated from the logic that generates the event. The user interface element is able to delegate the processing of an event to a separate piece of code.

## Steps Involved in Event Handling

**Step 1** − The user clicks the button and the event is generated.

**Step 2** − The object of concerned event class is created automatically and information about the source and the event get populated within the same object.

**Step 3** − Event object is forwarded to the method of the registered listener class.

**Step 4** − The method is gets executed and returns.

## Points to Remember About the Listener

* In order to design a listener class, you have to develop some listener interfaces. These Listener interfaces forecast some public abstract callback methods, which must be implemented by the listener class.
* If you do not implement any of the predefined interfaces, then your class cannot act as a listener class for a source object.

## Callback Methods

These are the methods that are provided by API provider and are defined by the application programmer and invoked by the application developer. Here the callback methods represent an event method. In response to an event, java jre will fire callback method. All such callback methods are provided in listener interfaces.

If a component wants some listener to listen ot its events, the source must register itself to the listener.

## Event Handling Example

Create the following Java program using any editor of your choice in say **D:/ > SWING > com > netparam > gui >**

***SwingControlDemo.java***

*package com.netparam.gui;*

*import java.awt.\*;*

*import java.awt.event.\*;*

*import javax.swing.\*;*

*public class SwingControlDemo {*

*private JFrame mainFrame;*

*private JLabel headerLabel;*

*private JLabel statusLabel;*

*private JPanel controlPanel;*

*public SwingControlDemo(){*

*prepareGUI();*

*}*

*public static void main(String[] args){*

*SwingControlDemo swingControlDemo = new SwingControlDemo();*

*swingControlDemo.showEventDemo();*

*}*

*private void prepareGUI(){*

*mainFrame = new JFrame("Java SWING Examples");*

*mainFrame.setSize(400,400);*

*mainFrame.setLayout(new GridLayout(3, 1));*

*headerLabel = new JLabel("",JLabel.CENTER );*

*statusLabel = new JLabel("",JLabel.CENTER);*

*statusLabel.setSize(350,100);*

*mainFrame.addWindowListener(new WindowAdapter() {*

*public void windowClosing(WindowEvent windowEvent){*

*System.exit(0);*

*}*

*});*

*controlPanel = new JPanel();*

*controlPanel.setLayout(new FlowLayout());*

*mainFrame.add(headerLabel);*

*mainFrame.add(controlPanel);*

*mainFrame.add(statusLabel);*

*mainFrame.setVisible(true);*

*}*

*private void showEventDemo(){*

*headerLabel.setText("Control in action: Button");*

*JButton okButton = new JButton("OK");*

*JButton submitButton = new JButton("Submit");*

*JButton cancelButton = new JButton("Cancel");*

*okButton.setActionCommand("OK");*

*submitButton.setActionCommand("Submit");*

*cancelButton.setActionCommand("Cancel");*

*okButton.addActionListener(new ButtonClickListener());*

*submitButton.addActionListener(new ButtonClickListener());*

*cancelButton.addActionListener(new ButtonClickListener());*

*controlPanel.add(okButton);*

*controlPanel.add(submitButton);*

*controlPanel.add(cancelButton);*

*mainFrame.setVisible(true);*

*}*

*private class ButtonClickListener implements ActionListener{*

*public void actionPerformed(ActionEvent e) {*

*String command = e.getActionCommand();*

*if( command.equals( "OK" )) {*

*statusLabel.setText("Ok Button clicked.");*

*} else if( command.equals( "Submit" ) ) {*

*statusLabel.setText("Submit Button clicked.");*

*} else {*

*statusLabel.setText("Cancel Button clicked.");*

*}*

*}*

*}*

*}*

**output.**

