Code Example of Event Handling

<u>Example # 01</u> (Simple Button press demo by using External Handler)

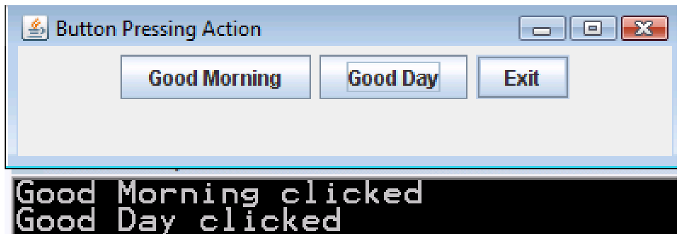
ButtonPressDemo.java

```
// use of External Hadnler class
import javax.swing.*; import java.awt.*;  import java.awt.event.*

public class ButtonPressDemo extends JFrame {

    public ButtonPressDemo(){
          JButton b;

           //Handler Class object
            MyActionListener a = new MyActionListener();

            Container c = getContentPane();
            c.setLayout(new FlowLayout());

            //add component and register with handler
            c.add(b = new JButton("Good Morning"));
            b.addActionListener(a);

            c.add(b = new JButton("Good Day"));
            b.addActionListener(a);

            c.add(b = new JButton("Exit"));
            b.addActionListener(a);

            this.setVisible(true);
            this.setSize(400,100);
            this.setTitle("Button Pressing Action");

    }
    public static void main(String[] args){
        ButtonPressDemo btdemo = new ButtonPressDemo();
  }   }
```

MyActionListener.java (Handler Class)

```
import java.awt.event.*;

public class MyActionListener implements ActionListener {
  public void actionPerformed(ActionEvent ae) {
      String s = ae.getActionCommand();
      if (s.equals("Exit")) {
          System.exit(0);
          }
          else {
                  System.out.println(s + " clicked");
                  }
        }
}
```

<u>Output:</u>



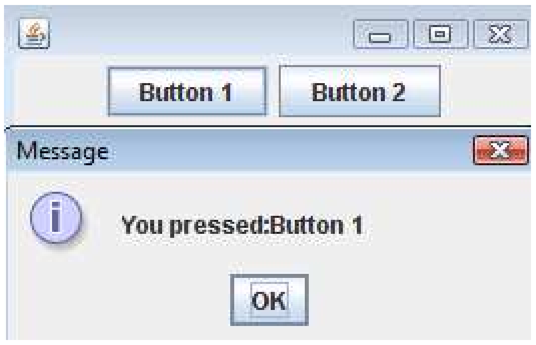
```
Good Morning clicked
Good Day clicked
```

<u>Example # 02</u> (Simple button press demo by using Inner classes)

AwtEvent.java

```java
// use of Inner class by ActionListner
import java.awt.event.*; import java.awt.*; import javax.swing.*;
public class AwtEvent extends JFrame
{
   private JButton btn1,btn2;
   public AwtEvent()
   {
      Container c = getContentPane();
      c.setLayout( new FlowLayout() );

      btn1 = new JButton( "Button 1" );
      c.add( btn1 );
      btn2 = new JButton( "Button 2" );
      c.add( btn2 );

      // create Handler for button event handling
      ButtonHandler handler = new ButtonHandler();
      btn1.addActionListener( handler );
      btn2.addActionListener( handler );
   }
   public static void main( String args[] )
   {
      AwtEvent AE = new AwtEvent(); // create ButtonFrame
      AE.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
      AE.setSize( 275, 110 ); // set frame size
      AE.setVisible( true ); // display frame
   } // end main
  // inner class for button event handling
   private class ButtonHandler implements ActionListener
   {  // handle button event
    public void actionPerformed( ActionEvent event )
    {JOptionPane.showMessageDialog( AwtEvent.this, "You pressed:"+ event.getActionCommand() );}
   }  }
```

<u>Output:</u>



You pressed:Button 1

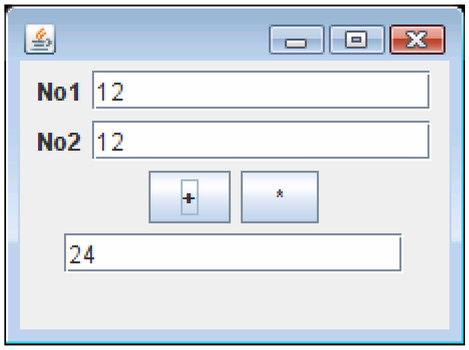
Code Example of Event Handling

<u>**Example # 03**</u> **(Simple calculator)**

SmallCalcApp.java

```java
import java.awt.*;  import javax.swing.*; import java.awt.event

 public class SmallCalcApp implements ActionListener{
 JFrame frame;
 JLabel firstOperand, secondOperand;
 JTextField op1, op2, ans;
 JButton plus, mul;

public void initGUI ( ) {
    frame = new JFrame();
        firstOperand = new JLabel("No1");
         secondOperand = new JLabel("No2");

         op1 = new JTextField (15);
         op2 = new JTextField (15);
         ans = new JTextField (15);
         plus = new JButton("+");
         mul = new JButton("*");

        Container cont = frame.getContentPane();
        cont.setLayout(new FlowLayout());

        cont.add(firstOperand);
        cont.add(op1);
        cont.add(secondOperand);
        cont.add(op2);
        cont.add(plus);
        cont.add(mul);
        cont.add(ans);
        plus.addActionListener(this);
        mul.addActionListener(this);

        Container cont = frame.getContentPane();
        cont.setLayout(new FlowLayout());

        cont.add(firstOperand);
        cont.add(op1);
        cont.add(secondOperand);
        cont.add(op2);
        cont.add(plus);
        cont.add(mul);
        cont.add(ans);
        plus.addActionListener(this);
        mul.addActionListener(this);

        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(230, 170);
        frame.setVisible(true);
  }

  //constructor
  public SmallCalcApp ( ) {
  initGUI();
  }

  public void actionPerformed(ActionEvent event) {
  String oper, result;
  int num1, num2, res;
 /* All the information regarding an event is contained
 inside the event object. Here we are calling the
 getSource() method on the event object to figure out
 the button that has generated that event. */
  if (event.getSource() == plus) {
  oper = op1.getText();
  num1 = Integer.parseInt(oper);
  oper = op2.getText();
  num2 = Integer.parseInt (oper);
  res = num1+num2;
  result = res+"";
  ans.setText(result);
  }
  else if (event.getSource() == mul) {
  oper = op1.getText();
  num1 = Integer.parseInt(oper);
  oper = op2.getText();
  num2 = Integer.parseInt (oper);
  res = num1*num2;
   result = res+"";
   ans.setText(result);
   }
}
   public static void main(String args[]) {
   SmallCalcApp scApp = new SmallCalcApp();
   }  }// end class
```

Output:



Code Example of Event Handling

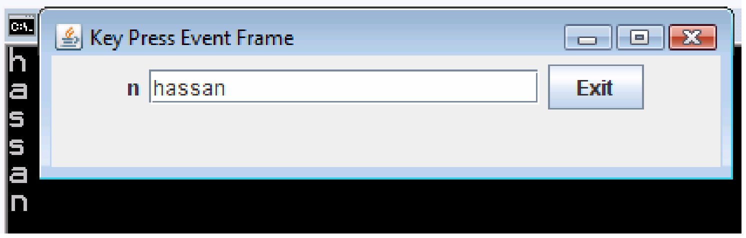
<u>Example # 04</u> (Key Listener Example)
GridLayoutFrame.java

```
// Adapter class use
// This class implements two Listner "KeyListner" and "ActionListner"

import javax.swing.*; import java.awt.*;
import java.awt.event.*;

public class KeyPress extends JFrame{
     JLabel label;
     JTextField txtField;
     JButton btn;

     public static void main(String[] args) {
          KeyPress k = new KeyPress();
     }

     public KeyPress(){
          super("Key Press Event Frame");

          MyKeyListener a = new MyKeyListener();
          MyActionListener b = new MyActionListener();

          Container c = getContentPane();
          c.setLayout(new FlowLayout());

          label = new JLabel();
          txtField = new JTextField(20);
          btn = new JButton("Exit");

          c.add(label);
          c.add(txtField);
          c.add(btn);
```

```
          c.add(label);
          c.add(txtField);
          c.add(btn);
```

```
          txtField.addKeyListener(a);
          btn.addActionListener(b);

          setSize(400,100);
          setVisible(true);
     }

     // This class implement keyListner functionalities
private class MyKeyListener extends KeyAdapter{
          public void keyPressed(KeyEvent ke){
               char i = ke.getKeyChar();
               System.out.println(i);
               String str = Character.toString(i);
               label.setText(str);
          }
     }

// This class implement ActionListner functionalities
private class MyActionListener implements ActionListener {
        public void actionPerformed(ActionEvent ae) {
             String s = ae.getActionCommand();
             if (s.equals("Exit")) {
                  System.out.println("Exiting.....");
                  System.exit(0);
                  }
             }
     }
}
```

<u>Output:</u>



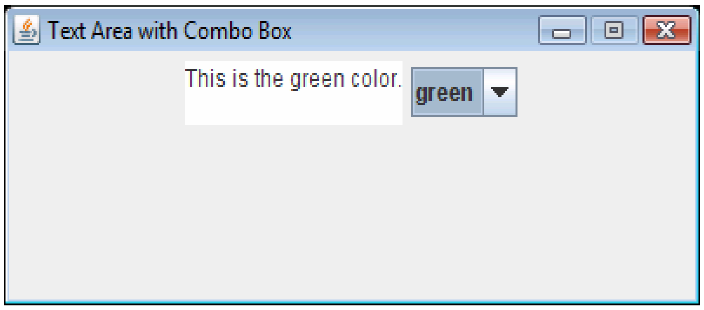
```
h
a
s
s
a
n
```

Example # 05 (Example of Item Listener)

AwtItemEvent.java

```
// use of Anonymous Inner classes for ItemListner
import java.awt.*; import java.awt.event.*;  import javax.swing.*;

public class AwtItemEvent extends JFrame{
    JTextArea txtArea;

    //Constructor
    public AwtItemEvent(String title){
        super("Text Area with Combo Box");

        Container c = getContentPane();
        c.setLayout( new FlowLayout() );

        txtArea = new JTextArea();
        c.add(txtArea);
        JComboBox jcb = new JComboBox();
        jcb.addItem("red");
        jcb.addItem("green");
        jcb.addItem("blue");
        c.add(jcb);

        jcb.addItemListener(new ItemListener(){
            public void itemStateChanged(ItemEvent e){
                txtArea.setText("This is the " + e.getItem() + " color.\n");
            }
        });

        setSize(400,150); setVisible(true); setResizable(false);
        setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
    }
    public static void main(String[] args){
        AwtItemEvent f = new AwtItemEvent("AWT Demo");
    } }
```

Output:



This is the green color.

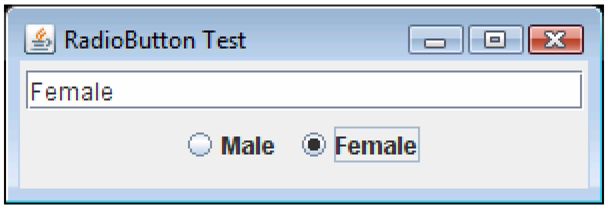
Code Example of Event Handling

<u>Example # 06</u> (Another example of Item Listener by using radiobutton)

RadioButtonFrame.java

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class RadioButtonFrame extends JFrame
{
   private JTextField textField;
   private JRadioButton male;
   private JRadioButton female;
   private ButtonGroup radioGroup;

   // RadioButtonFrame constructor adds JRadioButtons to JFrame
   public RadioButtonFrame()
   {
      super( "RadioButton Test" );
      Container c = getContentPane();
      c.setLayout( new FlowLayout() ); // set frame layout

      textField = new JTextField( "Gender",25);
      c.add( textField ); // add textField to JFrame

      // create radio buttons
      male = new JRadioButton( "Male", true );
      female = new JRadioButton( "Female", false );

      c.add( male );
      c.add( female );


      radioGroup = new ButtonGroup(); // create ButtonGroup
      radioGroup.add( male );
      radioGroup.add( female );

     // register events for JRadioButtons
     male.addItemListener( new RadioButtonHandler( ) );
     female.addItemListener(   new RadioButtonHandler( ) );
      }

  public static void main( String args[] )
  {
     RadioButtonFrame radioButtonFrame = new RadioButtonFrame();
     radioButtonFrame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
     radioButtonFrame.setSize( 300, 100 ); // set frame size
     radioButtonFrame.setVisible( true ); // display frame
  } // end main


  private class RadioButtonHandler implements ItemListener
  {
     public void itemStateChanged( ItemEvent event )
     {
        if ( event.getSource() == male )
        {
            textField.setText("Male");
        }
        else if( event.getSource() == female )
        {
            textField.setText("Female");
        }

     }
  }
```

<u>Output:</u>



```
Female
```

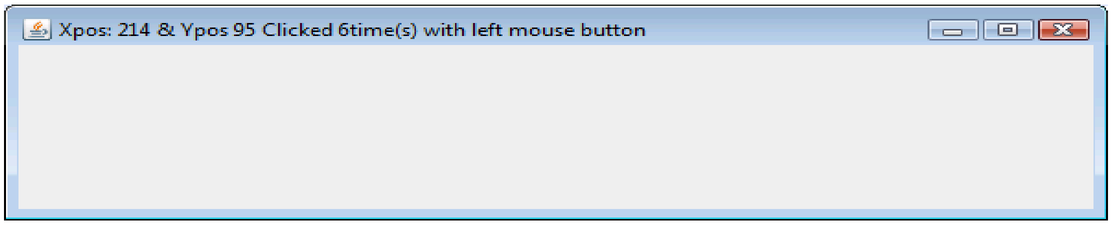
Code Example of Event Handling

<u>Example # 07</u> (Example of MouseAdapter class)

MouseDetailsFrame.java

```
import java.awt.*; import java.awt.event.*; import javax.swing.*;

public class MouseDetailsFrame extends JFrame
{
   private String details;

   // constructor sets title bar String and register mouse listener
   public MouseDetailsFrame()
   {
      super( "Mouse clicks and buttons" );
      Container c = getContentPane();
      c.setLayout( new FlowLayout() );

      addMouseListener( new MouseClickHandler() ); // add handler
   }

   public static void main( String args[] )
   {
      MouseDetailsFrame mouseDetailsFrame = new MouseDetailsFrame();
      mouseDetailsFrame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
      mouseDetailsFrame.setSize( 600, 150 );
      mouseDetailsFrame.setVisible( true );
   }

   // inner class to handle mouse events
   private class MouseClickHandler extends MouseAdapter
   {
      public void mouseClicked( MouseEvent event )
      {
         int xPos = event.getX(); // get x position of mouse
         int yPos = event.getY(); // get y position of mouse

         details = "Xpos: " + xPos+ " & Ypos " + yPos + " Clicked " +
                   event.getClickCount() + "time(s)";

         if ( event.isMetaDown() ) // right mouse button
            details += " with right mouse button";

         else if ( event.isAltDown() ) // middle mouse button
            details += " with center mouse button";

         else // left mouse button
            details += " with left mouse button";

         setTitle( details ); // display message in statusBar
      } // end method mouseClicked

   } // end private inner class MouseClickHandler
} // end class MouseDetailsFrame
```

<u>Output:</u>



```
Xpos: 214 & Ypos 95 Clicked 6time(s) with left mouse button
```

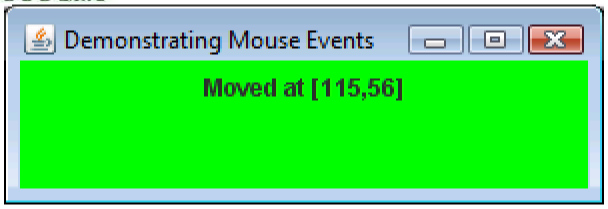
Code Example of Event Handling

<u>Example # 08</u> (Example of MouseMotionAdapter class)

MouseTrackerFrame.java

```
// A class implements two listners at the same time
 import java.awt.*; import java.awt.event.*; import javax.swing.*;

 public class MouseTrackerFrame extends JFrame
 {
    JLabel status;
    public MouseTrackerFrame()
    {
      super( "Demonstrating Mouse Events" );
      Container c = getContentPane();
      c.setLayout(new FlowLayout());

      status = new JLabel();
      c.add( status);

      // create and register listener for mouse and mouse motion events
       MouseHandler handler = new MouseHandler();
       this.addMouseListener( handler );
       this.addMouseMotionListener( handler );
    }

    public static void main( String args[] )
    {
       MouseTrackerFrame mouseTrackerFrame = new MouseTrackerFrame();
       mouseTrackerFrame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
       mouseTrackerFrame.setSize( 300, 100 ); // set frame size
       mouseTrackerFrame.setVisible( true ); // display frame
    } // end main

    private class MouseHandler implements MouseListener,MouseMotionListener
    {
    // handle event when mouse released immediately after press
    public void mouseClicked( MouseEvent event )
    {
       status.setText("Clicked at [" + event.getX() + ","+ event.getY() + "]") ;
    }

    // handle event when mouse pressed
    public void mousePressed( MouseEvent event )
    {
       status.setText( "Pressed at [" + event.getX() + ","+ event.getY() + "]");
    }

    // handle event when mouse released after dragging
    public void mouseReleased( MouseEvent event )
    {
       status.setText( "Released at [" + event.getX() + ","+ event.getY() + "]");
    }

    // handle event when mouse enters area
    public void mouseEntered( MouseEvent event )
    {
       status.setText( "Mouse entered at [" + event.getX() + ","+ event.getY() );
       getContentPane().setBackground( Color.GREEN );
    }

    // handle event when mouse exits area
    public void mouseExited( MouseEvent event )
    {
       status.setText( "Mouse outside" );
       getContentPane().setBackground( Color.WHITE );
    }
     // handle event when user drags mouse with button pressed
     public void mouseDragged( MouseEvent event )
     {
        status.setText( "Dragged at [" + event.getX() + ","+ event.getY()+ "]" );
     }

     // handle event when user moves mouse
     public void mouseMoved( MouseEvent event )
     {
        status.setText( "Moved at [" + event.getX() + ","+ event.getY() + "]");
     }

  } // end inner class MouseHandler
// end class MouseTrackerFrame
```

<u>Output:</u>



```
Moved at [115,56]
```