Madan Bhandari Memorial College
Department of Computer Science and Information Technology (B.Sc.CSIT)
Ninayak Nagar, New Baneshwor, Kathmandu

Practical Sheet

Submitted By:- Utsav Acharya Sharma
Program No:- 03
Submitted To Debesh Adhikari
Lab Date:- 2080/08/15
Submission Date:- 2080/08/11
T.U.Roll.No. :- 24179

Title: Implementation of Event Handling.

Introduction:

Event:

An event is an occurrence that happens during the execution of a program often triggered by user interactions or system events. Example: button clicks, key presses or mouse movements. There are various types of events in Java, broadly categorized into user interface events (e.g: button clicks, mouse events) and system events (e.g: timer events, window events).

Event Handling:

Event Handling is the process of capturing, processing and responding to events generated by user actions or system occurrences. It involves defining methods to handle specific events.

Delegation Event ~~Metho~~ Model:

Delegation Event Model involves using delegation to handle events. In this model, event handling is delegated to a separate object (listener) that is registered to receive specific types of events from a source object.

Callback Methods:

Methods that are predefined or user-defined and are called automatically in response to a specific event is known as callback method. In event handling, callback methods are implemented to respond to events.

Steps involved in event handling

i) Event Source Registration: Identify the source of the event and register it with a listener.

ii) Listener Interface Implementation: Implement the appropriate listener interface and override its methods to define the behaviour when the event occurs.

iii) Listener Register: Register the listener with the event source using registration methods.

iv) Event Dispatching: The event source generates the event, and the registered listener's callback method is invoked to handle the event.

Listener Interfaces

These are interfaces provided by Java that define methods to handle specific types of events. Example: ActionListener, MouseListener, etc.

Adapter Classes

These are classes that provide empty implementations for all methods in a listener interface. They allow a class to implement only the methods it needs, making it easier to handle events.

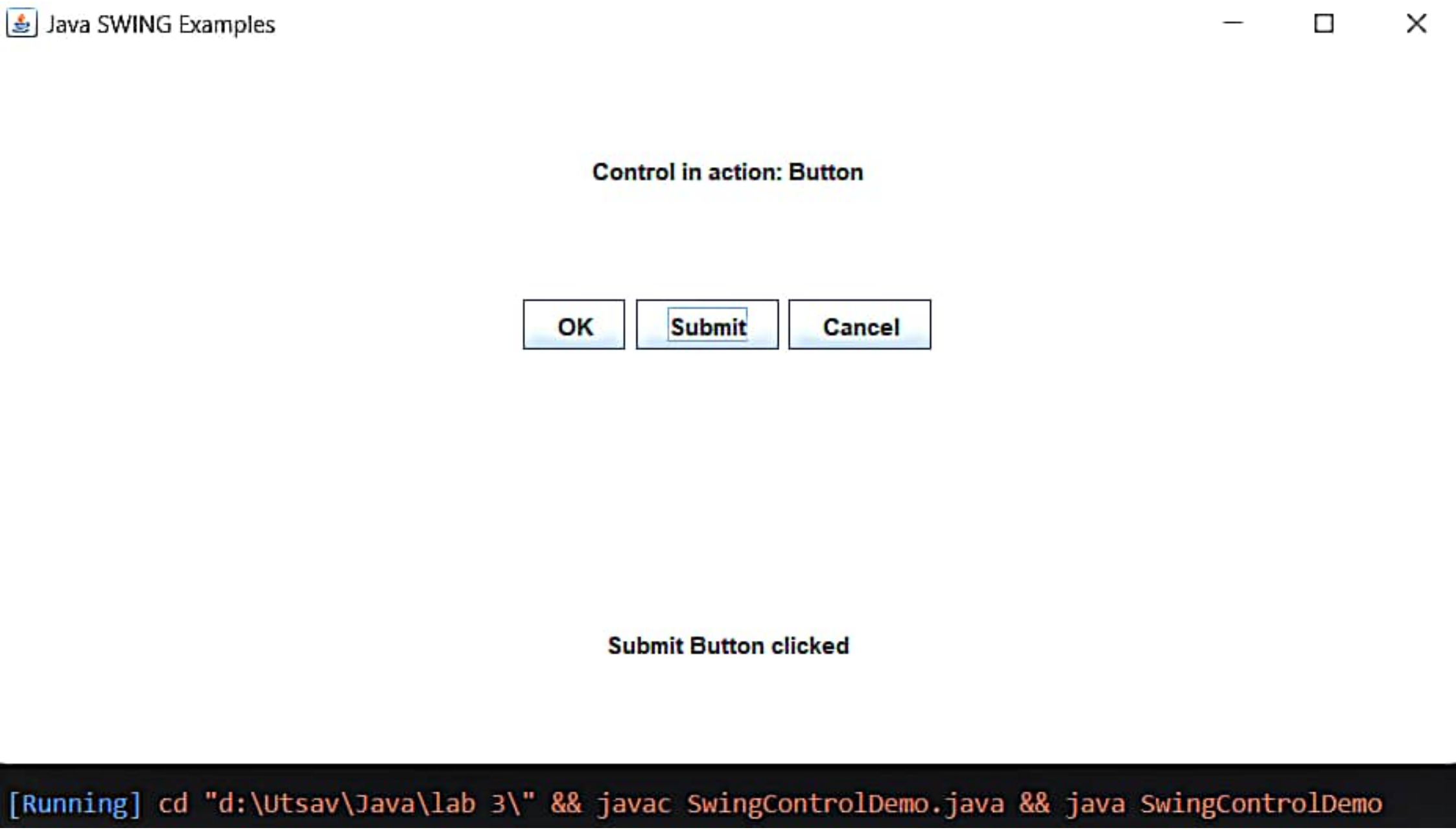
Code:

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

public class SwingContainerDemo{
    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel containerPanel;

    public SwingContainerDemo(){
        prepareGUI();
    }
    public static void main(String[] args){
        SwingContainerDemo swingContainerDemo = new SwingContainerDemo();
        swingContainerDemo.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(350,200);
        mainFrame.addWindowListener(new WindowAdapter(){
            public void windowClosing(WindowEvent windowEvent){
                System.exit(0);
            }
        });
        containerPanel = new JPanel();
        containerPanel.setLayout(new FlowLayout());
        mainFrame.add(headerLabel);
        mainFrame.add(containerPanel);
        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 = new JButton("Submit");
    cancelButton = new JButton("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.");
        }
    }
}
}
```


Java SWING Examples
Control in action: Button
OK
Submit
Cancel
Submit Button clicked
[Running] cd "d:\Utsav\Java\lab 3\" && javac SwingControlDemo.java && java SwingControlDemo

Code:

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

public class SwingAddTwoNumbers {
    private JFrame frame;
    private JTextField numField1, numField2;
    private JLabel numLabel1, numLabel2, resultLabel;

    public SwingAddTwoNumbers() {
        frame = new JFrame("Add Two Numbers");
        frame.setLayout(null);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        numLabel1 = new JLabel("Number 1:");
        numLabel1.setBounds(20, 50, 80, 30);

        numField1 = new JTextField();
        numField1.setBounds(100, 50, 100, 30);
        numLabel2 = new JLabel("Number 2:");
        numLabel2.setBounds(20, 90, 80, 30);
        numField2 = new JTextField();
        numField2.setBounds(100, 90, 100, 30);
        resultLabel = new JLabel("Result:");
        resultLabel.setBounds(50, 170, 200, 30);
        JButton addButton = new JButton("Add");
        addButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                try {
                    int num1 = Integer.parseInt(numField1.getText());
                    int num2 = Integer.parseInt(numField2.getText());
                    int sum = num1 + num2;
                    resultLabel.setText("Result: " + sum);
                }
                catch (NumberFormatException ex) {
                    resultLabel.setText("Result: Invalid Input");
                }
```

```
        }
    });
    addButton.setBounds(50, 130, 80, 30);
    frame.add(numLabel1);
    frame.add(numField1);
    frame.add(numLabel2);
    frame.add(numField2);
    frame.add(addButton);
    frame.add(resultLabel);
    frame.setSize(300, 250);
    frame.setVisible(true);
}
public static void main(String[] args){
    SwingUtilities.invokeLater(new Runnable(){
        public void run(){
            new SwingAddTwoNumbers();
        }
    });
}
}
```


Add Two Numbers
Number 1:
24
Number 2:
179
Add
Result: 203
[Running] cd "d:\Utsav\Java\lab 3\" && javac SwingAddTwoNumbers.java && java SwingAddTwoNumbers