Green University of Bangladesh
Department of Computer Science and Engineering (CSE)
Faculty of Sciences and Engineering
Semester:(Spring, Year:2024), B.Sc. in CSE (Day)

Lab Report NO: 09
Course Title: Object-Oriented Programming Lab
Course Code: CSE 202 Section: 223 D9

Lab Experiment Name: AWT Event Handling (ActionListener)

<u>Student Details</u>

Name		ID
1.	Promod Chandra Das	231002005

Lab Date :
Submission Date :
Course Teacher's Name : **Noyan Ali**

<u>Lab Report Status</u>
Marks: ……………………………… **Signature:**....................
Comments:... **Date:**............................

TITLE OF THE LAB REPORT EXPERIMENT:-

AWT Event Handling (ActionListener)

OBJECTIVES/AIM :

Understanding ActionListener and ActionEvent and using them to add events

Problem analysis:

We can use awt in java too add events. There are many kinds of events that we can add. We will focus on ActionListener here. But apart from that are there are other kinds of events too such as MouseListener, MouseMotionListener, ItemListener, WindowListener etc.

• We will add events to swing based programs

• We will create a window where we take 2 integers and divide them. We calculate the division when a button division is clicked. We also use swing to send messages to the user while handling exception.

Terms

ActionListener

Java ActionListener is an interface. It gets notified whenever user click on some button and performs some form of work. In Java ActionListener gets notified through ActionEvent. The ActionListener interface is found
in java.awt.event package. It has only one method: actionPerformed(). While using ActionListener one needs
to override this method.

actionPerformed()

actionPerformed() is the only method under ActionListener. As Java ActionListener is an interface and action-
Performed() is provided, we need to override this method during writing the event. We register components

such as buttons and when the button is used actionPerformed() gets invoked.

```
public abstract void actionPerformed(ActionEvent e);
```

➢ Steps to use ActionListener

• Implement the ActionListener class

```
public class Lab implements ActionListener
```

• Registering the component to ActionListener

```
component.addActionListener(instanceOfListenerclass);
```

• Then override the actionPerformed() method

```
public void actionPerformed(ActionEvent e){
//write code here
3 }
```

- Implementation

• as before create a java project
• create a class extending JFrame and implementing ActionListener interface
• create necessary buttons, textfields and add them to container
• take 2 integers using textfield
• override actionPerformed() and write code for division
• add exception handling according to requirement

Lab Exercise (Submit as a report)

• Build a swing based application that takes 2 strings, concatenates them, compares them when 2 buttons clicked, compare and concatenate and shows the output as labels.
• Build a swing based application where you can read some data from file and when when clicked a button the application render the data from file and show it on a textfield.

✓ Answer:01

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

public class StringManipulationApp extends JFrame implements ActionListener {
   private JTextField string1Field;
   private JTextField string2Field;
   private JLabel resultLabel;
   private JButton concatenateButton;
   private JButton compareButton;

   public StringManipulationApp() {
      super("String Manipulation");

      // Create components
      string1Field = new JTextField(10);
      string2Field = new JTextField(10);
      resultLabel = new JLabel("Result: ");
      concatenateButton = new JButton("Concatenate");
      compareButton = new JButton("Compare");

      // Add action listeners
      concatenateButton.addActionListener(this);
      compareButton.addActionListener(this);

      // Create panels and set layout
      JPanel inputPanel = new JPanel();
      inputPanel.add(new JLabel("String 1:"));
      inputPanel.add(string1Field);
      inputPanel.add(new JLabel("String 2:"));
```

```
        inputPanel.add(string2Field);

        JPanel buttonPanel = new JPanel();
        buttonPanel.add(concatenateButton);
        buttonPanel.add(compareButton);

        JPanel resultPanel = new JPanel();
        resultPanel.add(resultLabel);

        // Add panels to frame
        add(inputPanel, BorderLayout.NORTH);
        add(buttonPanel, BorderLayout.CENTER);
        add(resultPanel, BorderLayout.SOUTH);

        // Setup frame
        setSize(400, 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(true);
    }

    public void actionPerformed(ActionEvent e) {
        String str1 = string1Field.getText();
        String str2 = string2Field.getText();

        if (e.getSource() == concatenateButton) {
            String concatenated = str1 + str2;
            resultLabel.setText("Result: " + concatenated);
        } else if (e.getSource() == compareButton) {
            if (str1.equals(str2)) {
                resultLabel.setText("Result: Strings are equal");
            } else {
                resultLabel.setText("Result: Strings are not equal");
            }
        }
    }

    public static void main(String[] args) {
        new StringManipulationApp();
    }
}
```

✓ **Answer:02**

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class FileReaderApp extends JFrame implements ActionListener {
    private JTextField filePathField;
    private JTextArea fileContentArea;
    private JButton loadButton;

    public FileReaderApp() {
        super("File Reader");

        // Create components
        filePathField = new JTextField(20);
        fileContentArea = new JTextArea(10, 30);
        fileContentArea.setLineWrap(true);
        fileContentArea.setWrapStyleWord(true);
        fileContentArea.setEditable(false);
        loadButton = new JButton("Load File");

        // Add action listener
        loadButton.addActionListener(this);

        // Create panels and set layout
        JPanel inputPanel = new JPanel();
        inputPanel.add(new JLabel("File Path:"));
        inputPanel.add(filePathField);
        inputPanel.add(loadButton);

        JPanel contentPanel = new JPanel();
        contentPanel.add(new JScrollPane(fileContentArea));

        // Add panels to frame
        add(inputPanel, BorderLayout.NORTH);
        add(contentPanel, BorderLayout.CENTER);

        // Setup frame
        setSize(400, 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(true);
    }

    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == loadButton) {
            String filePath = filePathField.getText();
            try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
                fileContentArea.setText("");
                String line;
                while ((line = reader.readLine()) != null) {
                    fileContentArea.append(line + "\n");
                }
            } catch (IOException ex) {
                JOptionPane.showMessageDialog(this, "Error reading file: " + ex.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
            }
        }
    }
    public static void main(String[] args) {
        new FileReaderApp();
    }
}
```

➢ Discussion & Conclusion

AWT's ActionListener interface facilitates event handling in Java GUI applications. By implementing its actionPerformed method, developers can respond to user-triggered actions, such as button clicks. This approach promotes interactive and responsive user interfaces, enhancing the overall user experience. ActionListener fosters modular design, separating user interface logic from application logic, thus improving code organization and maintainability.