UNIT-5

AWT Components andEvent Handlers:

AWT Components:

AWT (Abstract Window Toolkit) is Java's original platform-dependent windowing, graphics, and user-interface widget toolkit. Below are some key AWT components:

1. **Frame**: A top-level window with a title and a border.
2. **Panel**: A generic container for holding AWT components.
3. **Button**: A labeled button that can trigger an action when clicked.
4. **Label**: A display area for a short text string.
5. **TextField**: A single-line text input field.
6. **TextArea**: A multi-line text input area.
7. **Checkbox**: A box that can be checked or unchecked.
8. **Choice**: A drop-down list of items.
9. **List**: A list of items that allows multiple selections.

Event Handlers:

In Java, event handling is the mechanism that controls the events, such as actions, mouse movements, and key presses. Key event handler interfaces and classes include:

1. **ActionListener**: Used for handling action events like button clicks.
2. **MouseListener**: Used for handling mouse events.

3. **KeyListener**: Used for handling keyboard events.

Simple Calculator Program Using AWT

Here is an example of a simple calculator program in Java using AWT components and event handlers:

SimpleCalculator.java

```
import java.awt.*;
import java.awt.event.*;
public class SimpleCalculator extends Frame implements
ActionListener {
    // Declare components
    TextField input1, input2, output;
    Button addButton, subButton, mulButton, divButton;
// Constructor to set up GUI
    public SimpleCalculator() {
// Create components
    Label label1 = new Label("Number 1:");
    Label label2 = new Label("Number 2:");
    Label label3 = new Label("Result:");
    input1 = new TextField(10);
    input2 = new TextField(10);
    output = new TextField(10);
    output.setEditable(false); // Result field should be
non-editable

    addButton = new Button("+");
    subButton = new Button("-");
    mulButton = new Button("*");
    divButton = new Button("/");
    // Register event listeners
    addButton.addActionListener(this);
```

```
        subButton.addActionListener(this);
        mulButton.addActionListener(this);
        divButton.addActionListener(this);
    // Set layout manager and add components
        setLayout(new FlowLayout());
        add(label1);
        add(input1);
        add(label2);
        add(input2);
        add(label3);
        add(output);
        add(addButton);
        add(subButton);
        add(mulButton);
        add(divButton);
        // Frame settings
        setTitle("Simple Calculator");
        setSize(250, 200);
        setVisible(true);
        // Add window closing event
        addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent we) {
        System.exit(0);
        }
        });
    }
    // Event handling for button clicks
        public void actionPerformed(ActionEvent ae) {
        try {
        double num1 = Double.parseDouble(input1.getText());
        double num2 = Double.parseDouble(input2.getText());
        double result = 0;
        if (ae.getSource() == addButton) {
```

```
        result = num1 + num2;
        } else if (ae.getSource() == subButton) {
        result = num1 - num2;
        } else if (ae.getSource() == mulButton) {
        result = num1 * num2;
        } else if (ae.getSource() == divButton) {
        result = num1 / num2;
        }
        output.setText(String.valueOf(result));
        } catch (NumberFormatException e) {
        output.setText("Invalid input");
        } catch (ArithmeticException e) {
        output.setText("Error");
        }
}
// Main method to run the calculator
public static void main(String[] args) {
new SimpleCalculator();
        }
}
```

Swings:

Introduction to Swing

Swing is a part of Java Foundation Classes (JFC) that provides a rich set of GUI components. Unlike AWT, Swing components are lightweight, more flexible, and platform-independent. Here are some reasons why Swing is considered better than AWT:

1. **Lightweight Components**: Swing components are written entirely in Java and do not rely on platform-specific code.

2. **Pluggable Look and Feel**: Swing allows users to change the look and feel of applications at runtime.

3. **Advanced Components**: Swing provides a richer set of GUI components than AWT, such as trees, tables, and text components with more functionalities.

4. **MVC Architecture**: Swing follows the Model-View-Controller (MVC) architecture, making it easier to separate the data (model) from the presentation (view) and the logic (controller).

Hierarchy of Swing Components

The hierarchy of Swing components is built on top of the AWT hierarchy. Here are the primary components:

- **JComponent**: The base class for all Swing components.
    - **JButton**
    - **JLabel**
    - **JTextField**
    - **JTextArea**
    - **JCheckBox**
    - **JRadioButton**
    - **JComboBox**
    - **JList**
    - **JPanel**
    - **JFrame**: The top-level container for Swing applications.

- ❒ **JDialog**
- ❒ **JScrollPane**
- ❒ **JTabbedPane**
- ❒ **JSplitPane**
- ❒ **JToolBar**

Simple Calculator Program Using Swing

Here is the simple calculator program using Swing:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class SimpleCalculatorSwing extends JFrame
implements ActionListener {
// Declare components
JTextField input1, input2, output;
JButton addButton, subButton, mulButton, divButton;
// Constructor to set up GUI
public SimpleCalculatorSwing() {
// Create components
JLabel label1 = new JLabel("Number 1:");
JLabel label2 = new JLabel("Number 2:");
JLabel label3 = new JLabel("Result:");
input1 = new JTextField(10);
input2 = new JTextField(10);
output = new JTextField(10);
output.setEditable(false); // Result field should be non-
editable
```

```
addButton = new JButton("+");
subButton = new JButton("-");
mulButton = new JButton("*");
divButton = new JButton("/");
// Register event listeners
addButton.addActionListener(this);
subButton.addActionListener(this);
mulButton.addActionListener(this);
divButton.addActionListener(this);
// Set layout manager and add components
setLayout(new FlowLayout());
add(label1);
add(input1);
add(label2);
add(input2);
add(label3);
add(output);
add(addButton);
add(subButton);
add(mulButton);
add(divButton);
// Frame settings
setTitle("Simple Calculator");
setSize(300, 200);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setVisible(true);
}
// Event handling for button clicks
public void actionPerformed(ActionEvent ae) {
try {
    double num1 = Double.parseDouble(input1.getText());
    double num2 = Double.parseDouble(input2.getText());
    double result = 0;
```

```
        if (ae.getSource() == addButton) {
        result = num1 + num2;
        } else if (ae.getSource() == subButton) {
        result = num1 - num2;
        } else if (ae.getSource() == mulButton) {
     result = num1 * num2;
        } else if (ae.getSource() == divButton) {
        result = num1 / num2;
        }
        output.setText(String.valueOf(result));
        } catch (NumberFormatException e) {
        output.setText("Invalid input");
        } catch (ArithmeticException e) {
        output.setText("Error");
        }
    }
    // Main method to run the calculator
        public static void main(String[] args) {
        new SimpleCalculatorSwing();
        }
    }
```