**UNIT 7 GUI using Swing**

**Introduction to Swing and GUI Development**

**Objective:** By the end of this lesson, students will understand the basics of creating graphical user interfaces (GUIs) using Swing in Java, explore different types of GUIs, and discuss the future of GUI development.

**Introduction to GUIs**

1. **What is a GUI?**
   * A graphical user interface (GUI) allows users to interact with electronic devices through graphical icons and visual indicators as opposed to text-based interfaces, typed command labels, or text navigation.
2. **Types of GUIs:**
   * **Desktop GUIs:** Traditional GUIs found in desktop applications like Microsoft Word or Adobe Photoshop.
   * **Web GUIs:** GUIs in web applications accessed via browsers, e.g., Google Docs, Gmail.
   * **Mobile GUIs:** GUIs designed for mobile devices, e.g., apps on iOS and Android.
   * **Command Line Interfaces (CLI):** Text-based interfaces used to interact with software by typing commands.

**The Future of GUI Development**

1. **Evolution of GUIs:**
   * **Voice Interfaces:** Use of voice commands to interact with devices (e.g., Siri, Alexa).
   * **Augmented Reality (AR):** Overlaying digital content on the real world (e.g., AR apps on smartphones).
   * **Virtual Reality (VR):** Immersive 3D environments (e.g., VR gaming, VR training simulations).
   * **Gesture-based Interfaces:** Using hand and body gestures to interact with devices (e.g., Xbox Kinect).
2. **Emerging Trends:**
   * **Cross-platform Development:** Tools and frameworks that allow the development of GUIs that work seamlessly across different platforms (e.g., Flutter, React Native).
   * **AI and Machine Learning Integration:** Personalized and adaptive interfaces based on user behaviour.
   * **Minimalistic and User-centric Design:** Focus on simplicity and enhancing user experience.

**Introduction to Swing**

* **Swing Basics:**
  + Swing is part of Java's standard library and provides a set of 'lightweight' (all-Java language) components that, to the maximum degree possible, work the same on all platforms.
  + Components include buttons, checkboxes, labels, and more, typically prefixed with 'J', such as **JButton**, **JLabel**, etc.
* **Key Swing Components:**
  + **JFrame:** The main window container.
  + **JButton:** A clickable button.
  + **JLabel:** A display area for a short text string or an image icon.
  + **JTextField:** A single-line text input field.
  + **JPanel:** A generic container for lightweight components.

**Example 1: Simple Swing Application**

**Description:** Create a simple application with a JFrame, a JLabel, and a JButton.

import javax.swing.\*;

public class SimpleSwingApp {

public static void main(String[] args) {

// Create a new JFrame

JFrame frame = new JFrame("Simple Swing Application");

frame.setSize(400, 200);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

// Create a new JLabel

JLabel label = new JLabel("Hello, Swing!", SwingConstants.CENTER);

// Create a new JButton

JButton button = new JButton("Click Me");

button.addActionListener(e -> JOptionPane.showMessageDialog(null, "Button Clicked!"));

// Add components to the frame

frame.setLayout(new BorderLayout());

frame.add(label, BorderLayout.CENTER);

frame.add(button, BorderLayout.SOUTH);

// Make the frame visible

frame.setVisible(true);

}

}

**Example 2: Calculator Application**

**Description:** Develop a simple calculator using JFrame, JTextFields, JButtons, and ActionListener.

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class CalculatorApp extends JFrame implements ActionListener {

private JTextField input1, input2, result;

private JButton addButton, subtractButton, multiplyButton, divideButton;

public CalculatorApp() {

// Set up the frame

setTitle("Simple Calculator");

setSize(300, 200);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLayout(new GridLayout(5, 2));

// Initialize components

input1 = new JTextField();

input2 = new JTextField();

result = new JTextField();

result.setEditable(false);

addButton = new JButton("Add");

subtractButton = new JButton("Subtract");

multiplyButton = new JButton("Multiply");

divideButton = new JButton("Divide");

// Add action listeners

addButton.addActionListener(this);

subtractButton.addActionListener(this);

multiplyButton.addActionListener(this);

divideButton.addActionListener(this);

// Add components to the frame

add(new JLabel("Number 1:"));

add(input1);

add(new JLabel("Number 2:"));

add(input2);

add(new JLabel("Result:"));

add(result);

add(addButton);

add(subtractButton);

add(multiplyButton);

add(divideButton);

setVisible(true);

}

@Override

public void actionPerformed(ActionEvent e) {

try {

double num1 = Double.parseDouble(input1.getText());

double num2 = Double.parseDouble(input2.getText());

double res = 0;

if (e.getSource() == addButton) {

res = num1 + num2;

} else if (e.getSource() == subtractButton) {

res = num1 - num2;

} else if (e.getSource() == multiplyButton) {

res = num1 \* num2;

} else if (e.getSource() == divideButton) {

if (num2 != 0) {

res = num1 / num2;

} else {

JOptionPane.showMessageDialog(this, "Cannot divide by zero", "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

}

result.setText(String.valueOf(res));

} catch (NumberFormatException ex) {

JOptionPane.showMessageDialog(this, "Invalid input", "Error", JOptionPane.ERROR\_MESSAGE);

}

}

public static void main(String[] args) {

new CalculatorApp();

}

}

**Using NetBeans for Drag-and-Drop GUI Design**

1. **Introduction to NetBeans:**
   * NetBeans is a popular Integrated Development Environment (IDE) for Java development.
   * It provides a drag-and-drop GUI builder for Swing applications, making it easier to design GUIs visually.
2. **Creating a GUI using NetBeans:**
   * **Step 1:** Open NetBeans and create a new Java project.
   * **Step 2:** Create a new JFrame form by right-clicking on the project, selecting **New > JFrame Form**.
   * **Step 3:** Use the palette to drag and drop components like **JButton**, **JLabel**, **JTextField**, etc., onto the form.
   * **Step 4:** Customize the properties of components using the Properties window.
   * **Step 5:** Double-click on components to generate event handling code, such as action listeners for buttons.
   * **Example: Building a Simple GUI:**
     + Create a new JFrame form named **SimpleGUI**.
     + Drag a **JLabel** and a **JButton** onto the form.
     + Set the text of the **JLabel** to "Hello, NetBeans!".
     + Set the text of the **JButton** to "Click Me".
     + Double-click the **JButton** to generate an action listener and add the following code:

**private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) { JOptionPane.showMessageDialog(this, "Button Clicked!"); }**

1. **Running the GUI:**
   * Click the **Run** button in NetBeans to compile and run the application.
   * The application window should display the **JLabel** and **JButton**. Clicking the button should show a message dialog.

**The Status of Swing in Java**

1. **Swing's Place in Java:**
   * Swing is a part of the Java Foundation Classes (JFC) and has been a core part of Java's standard library since J2SE 1.2.
   * Despite newer GUI frameworks, Swing remains popular due to its maturity, flexibility, and integration with Java SE.
2. **Current Trends and Alternatives:**
   * **JavaFX:** A modern alternative to Swing, offering richer UI controls, CSS styling, and hardware-accelerated graphics.
   * **Web-based GUIs:** Many applications are moving to web-based interfaces using HTML, CSS, and JavaScript.
   * **Cross-platform frameworks:** Tools like Electron, React Native, and Flutter allow developers to build GUIs that work across multiple platforms, including desktop and mobile.
3. **Future of Swing:**
   * Swing is still maintained, but development has slowed as the focus shifts to JavaFX and other modern frameworks.
   * Many legacy applications continue to use Swing, and it remains a valuable skill for maintaining and updating these applications.

**Conclusion**

* Summarize the key points about Swing, GUI development, and the use of NetBeans for visual GUI design.
* Encourage students to explore more complex Swing components, layout managers, and the capabilities of JavaFX.
* Discuss the importance of user experience in GUI design and the future trends in GUI development.