**JAVA SWING**

Swing is a more advanced and flexible library built on top of AWT. Unlike AWT, Swing components are platform-independent, meaning that they are entirely written in Java and do not rely on the native OS.

This allows Swing components to look consistent across different platforms. Swing also provides richer components like tables, trees, lists, and a more powerful event handling mechanism.

Swing is a Set of API (API- Set of Classes and Interfaces)

Swing is Provided to Design Graphical User Interfaces

Swing is an Extension library to the AWT (Abstract Window Toolkit)

Swing Follows MVC.

The MVC Connection

In general, a visual component is a composite of three distinct aspects:

The way that the component looks when rendered on the screen.

The way such that the component reacts to the user.

The state information associated with the component.

--Swing components requires javax.swing package.

---

AWT components are heavyweight, meaning they are tied directly to the native windowing system (platform-dependent)

Swing components are lightweight, meaning they are written entirely in Java and do not depend on native OS components. Swing provides a consistent look-and-feel across different platforms.

--AWT Frame and Button:

Frame is a direct subclass of java.awt.Window and java.awt.Container.

Button is a subclass of java.awt.Component.

--Swing JFrame and JButton:

JFrame is a subclass of java.awt.Frame, but Swing adds many features like support for pluggable look-and-feel, more advanced layout and event handling.

JButton is a subclass of javax.swing.AbstractButton, providing richer functionality and appearance control compared to AWT's Button.

--AWT Components:

AWT components inherit the native look and feel of the operating system, meaning they look different on each platform (e.g., Windows buttons will look different from Mac buttons).

Swing Components:

Swing uses the Java look and feel (Metal) by default but can also support various custom themes and styles via the pluggable look-and-feel mechanism. You can make Swing components look consistent across different operating systems.

--AWT components like Frame and Button are considered outdated for most modern applications.

Swing components like JFrame and JButton are more commonly used due to their advanced features, flexibility, and ease of use.

1. JFrame (Main Window):

- The main window of the application where all other components (like the table and button) are displayed.

- frame.setSize(500, 500); sets the window's width and height to 500x500 pixels.

- frame.setLayout(null); tells the frame to use no layout manager, meaning components will need to be manually positioned.

- frame.setVisible(true); makes the frame visible to the user.

2. JButton (Submit Button):

- A clickable button labeled "Submit.

- button.setBounds(150, 200, 220, 50); sets its position and size. The button is placed at coordinates (150, 200) with a width of 220 pixels and a height of 50 pixels.

- frame.add(button); adds the button to the frame.

3. JTable (Table):

- Displays data in a tabular format.

- String[] columnNames = { "id", "Name" }; defines the column headers for the table: "id" and "Name."

- String[][] data = { { "1", "Debadatta" }, { "2", "Tushar" } }; specifies the data for the table, where each array is a row. There are two rows of data: (1, Debadatta) and (2, Tushar).

- JTable table = new JTable(data, columnNames);` creates a table with the specified data and column names.

4. JScrollPane (Scroll Pane):

- A container that provides scrollbars if the content inside it (in this case, the table) exceeds the visible area.

- JScrollPane s = new JScrollPane(table); puts the table inside the scroll pane.

- s.setBounds(50, 60, 300, 100); sets the size and position of the scroll pane within the frame. It is placed at coordinates (50, 60) with a width of 300 pixels and a height of 100 pixels.

- frame.add(s); adds the scroll pane (with the table inside) to the frame.

5. Action Listener for Button:

- Event Handling: When the "Submit" button is clicked, the program should perform some action. This is handled using an ActionListener.

- button.addActionListener(new ActionListener() {...}); adds a listener to the button, which listens for click events.

- Inside the actionPerformed method: System.out.println("Button Clicked!"); prints "Button Clicked!" to the console when the button is pressed.

6. Window Listener:

- WindowAdapter: Used to handle window-related events, such as closing the window.

- frame.addWindowListener(new WindowAdapter() {...}); adds a listener to the frame to detect when the window is closing.

- frame.dispose(); ensures the frame is properly closed when the window is closed by the user.

Flow of the Program:

1. Create the JFrame: A 500x500 window is created.

2. Add Components:

- A "Submit" button is placed in the window.

- A table is created with two rows of data ("1, Debadatta" and "2, Tushar") and two columns ("id" and "Name").

- The table is placed inside a scroll pane, which is positioned inside the frame.

3. Event Handling:

- When the button is clicked, "Button Clicked!" is printed to the console.

- When the user closes the window, the program disposes of the frame and exits gracefully.

Why Use a JScrollPane for JTable?

The table is placed inside a JScrollPane because tables can sometimes have more data than can fit in the visible area of the window.

The scroll pane adds scrollbars automatically when needed, allowing the user to view all the data. Without the scroll pane, the table could be cut off if it exceeds the window's size.

Output:

- The program opens a GUI window with a table showing two rows ("Debadatta" and "Tushar") and a "Submit" button.

- When you click the "Submit" button, "Button Clicked!" is printed to the console.

- Closing the window exits the program properly.