UNIT V – MULTITHREADING, GUI PROGRAMMING & EVENT HANDLING

Introduction to AWT and Swing:

AWT:

- AWT (Abstract Window Toolkit) is a part of Java's standard library used to create Graphical User Interfaces (GUI).
- It is platform-dependent, meaning it relies on the operating system's native GUI components.
- It belongs to the java.awt package. Provides GUI components like Button, Label, TextField, TextArea, Checkbox, List, Choice, Frame, Panel.
- Uses peer classes (native resources of the operating system).
- Event-driven programming model (actions like clicking a button trigger events).

AWT Hierarchy

- At the top level, AWT components are derived from:
  Object → Component → Container → Window → Frame/Dialog
- Component: Basic buildingblock (Button, Label, TextField).
- Container: Can hold other components (Panel, Frame).
- Window: Top-level window (cannot be contained inside another container).
- Frame: Main window with title bar, resize, minimize/maximize options.
- Dialog: Popup window for interaction.

Commonly Used AWT Classes

- ✓ Label – displays text.
- ✓ Button – clickable button.
- ✓ TextField – single-line text input.
- ✓ TextArea – multi-line text input.
- ✓ Checkbox – selectable checkbox.
- ✓ CheckboxGroup – radio buttons.
- ✓ Choice – dropdown menu.
- ✓ List – list of items.
- ✓ Frame – top-level window.

Swing:

- Swing is a part of Java Foundation Classes (JFC) that is used to create Graphical User Interface (GUI) applications.

- It is built on top of AWT (Abstract Window Toolkit) but is more powerful, flexible, and lightweight.
- Swing provides a rich set of components like buttons, labels, text fields, tables, trees, and more.
- All Swing classes are defined in the package: javax.swing.*

Difference Between AWT and Swing:

Feature	AWT	Swing
Package	java.awt	javax.swing
Component Type	Heavyweight (depends on OS)	Lightweight (pure Java)
Speed	Faster (direct OS calls)	Slightly slower (runs above AWT)
Component Variety	Limited (basic controls only)	Rich set (tables, trees, sliders, tooltips, tabbed panes, etc.)
Architecture	Simpler	MVC-based (Model-View-Controller)
Extensibility	Less flexible	More flexible & customizable
Event Handling	Delegation event model (introduced later)	Uses same event model, more consistent
Example	Button, Label, TextField, Frame	JButton, JLabel, JTextField, JFrame

Common Swing Components:

Component	Description
JFrame	Top-level window for GUI applications
JLabel	Displays a text or image
JButton	A clickable button
JTextField	Allows user to enter a single line of text
JTextArea	Multi-line text input
JCheckBox	Checkbox for true/false input
JRadioButton	Radio button for single-choice selection
JList	Displays a list of items
JTable	Displays tabular data
JMenu, JMenuItem	Menus and menu items

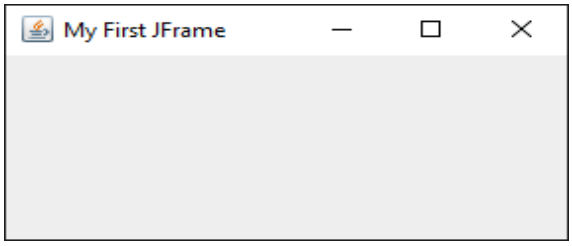
1. JFrame:

- JFrame is a class in the javax.swing package that represents a top-level window with a title bar, border, and buttons (close, minimize, maximize).
- It is the main container in a Swing application where we place other GUI components like buttons, labels, text fields, menus, panels, etc.
- Can hold menus (JMenuBar), panels (JPanel), and many Swing components.
- Often used as the main window in desktop applications.

Useful JFrame Methods:

Method	Description
setTitle(String title)	Sets the window title
setSize(int w, int h)	Sets width & height
setLocation(int x, int y)	Sets window position
setLocationRelativeTo(null)	Centers the window
setResizable(boolean flag)	Allows/disallows resizing
setDefaultCloseOperation(int op)	Defines behavior when closing window
add(Component c)	Adds a component
setLayout(LayoutManager m)	Sets layout manager
setVisible(true)	Makes window visible

```
import javax.swing.*; public class SimpleJFrame
{
    public static void main(String[] args)
    {
        JFrame frame = new JFrame("My First JFrame"); frame.setSize(300,
        150);  // set size
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLocationRelativeTo(null);
        frame.setVisible(true);  // make it visible
    }
}
```



2. JLabel:

- JLabel is a GUI component in Swing used to display a short string of text, an image, or both.
- It is non-editable (unlike JTextField), meaning users cannot modify its content directly.
- Supports alignment (left, center, right).
- Often used for labels beside text fields, buttons, or images.

Common Constructors:

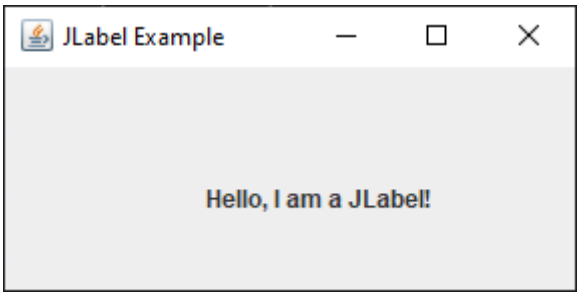
```
JLabel()                    // empty
label JLabel(String text)   // label
with text
JLabel(Icon image)          // label with image
JLabel(String text, int alignment)  // text with
alignment
JLabel(String text, Icon icon, int alignment) // text + image + alignment
```

Common Methods:

Method	Description
setText(String text)	Sets the text
getText()	Returns the text
setIcon(Icon icon)	Sets an image
setHorizontalAlignment(int align)	Aligns LEFT, CENTER, RIGHT
setForeground(Color c)	Sets text color
setFont(Font f)	Sets font style and size

```
import
javax.swing.*;
public class
JLabelExample
{
    public static void main(String[] args)
    {
        JFrame frame = new JFrame("JLabel Example");
        JLabel label = new JLabel("Hello, I am a JLabel!"); //Create a label with
        text label.setBounds(100, 50, 200, 30);
        frame.add(label);        // Add label to frame
        frame.setSize(300, 150);       // Frame settings
```

```
        frame.setLayout(null);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_C
        LOSE); frame.setVisible(true);
    }
}
```



3. JButton:

- JButton is a Swing component that represents a push button.
- When the user clicks it, an action event is triggered.
- It can display text, icon (image), or both.
- It often used for submitting forms, triggering actions, opening dialogs, etc.

Constructors:

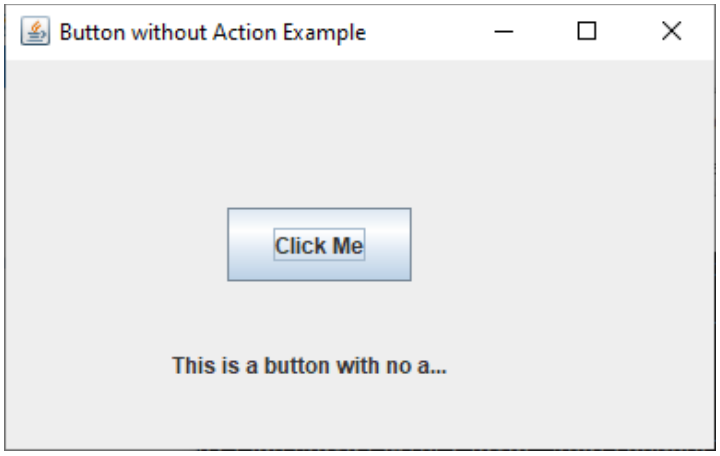
```
JButton()                    // Empty button
JButton(String text)         // Button with
text JButton(Icon image)     // Button
with image
JButton(String text, Icon image) // Button with text + image
```

Useful Methods:

Method	Description
setText(String text)	Sets button label
getText()	Gets current text
setIcon(Icon icon)	Sets an image icon
setEnabled(boolean b)	Enables/disables button
addActionListener(ActionListener l)	Adds click listener
setToolTipText(String text)	Shows tooltip on hover

```
import
javax.swing.*;
public class
JButtonNoAction
{
```

```
    public static void main(String[] args)
    {
        JFrame frame = new JFrame("Button without Action
        Example"); JButton button = new JButton("Click Me");    //
        Create a button button.setBounds(120, 80, 100, 40);
        JLabel label = new JLabel("This is a button with no action.");
        label.setBounds(90, 150, 150, 30);
        frame.add(button);        // Add
        components frame.add(label);
        frame.setSize(400, 250);        // Frame settings
        frame.setLayout(null);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_C
        LOSE); frame.setVisible(true);
    }
}
```



4. JTextField:

- JTextField is a Swing component used to enter and edit a single line of text.
- Commonly used for inputs like username, email, numbers, search boxes, etc.
- It allows user input (editable text).
- It supports event handling (e.g., pressing Enter triggers an ActionEvent).
- We can set the default text, columns, font, colors, etc.

Constructors:

```
JTextField()                    // empty text field
JTextField(String text)         // with initial text
JTextField(int columns)         // sets width (in characters)
JTextField(String text, int cols)   // with text and width
```

Useful Methods:

Method	Description
setText(String text)	Sets the text
getText()	Returns the current text
setEditable(boolean b)	Makes field editable or read-only
setColumns(int n)	Sets number of columns
setFont(Font f)	Changes font style
setForeground(Color c)	Changes text color
setBackground(Color c)	Changes background color
addActionListener(ActionListener l)	Triggers event on Enter key

```
import javax.swing.*;
public class JTextFieldExample
{
    public static void main(String[] args)
    {
        JFrame frame = new JFrame("JTextField Example");
        JTextField textField = new JTextField("Enter your name");
        textField.setBounds(100, 50, 200, 30);
        frame.add(textField);
        frame.setSize(400, 150);
        frame.setLayout(null);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```