Abstract Window Toolkit (AWT) in Java

1. What is AWT?

- AWT (Abstract Window Toolkit) is **Java's original GUI (Graphical User Interface) framework**.
- Part of the `java.awt` package.
- Provides classes for creating **windows, buttons, text fields, menus, checkboxes, etc.**
- It is **platform-dependent** (uses native OS components → "heavyweight components").

2. Commonly Used AWT Classes

- **Frame** → Main window.
- **Button** → Clickable button.
- **Label** → Displays text.
- **TextField** → Single-line text input.
- **TextArea** → Multi-line text input.
- **Checkbox / CheckboxGroup** → Option selections.
- **List** → Multiple item selection.
- **Menu, MenuItem, MenuBar** → Menus in window.
- **Panel** → Container for grouping components.

3. Basic Example (AWT Frame with Button & TextField)

```
import java.awt.*;
import java.awt.event.*;

public class AWTExample {
    public static void main(String[] args) {
        // Create Frame
        Frame f = new Frame("AWT Demo");

        // Create Label
        Label l = new Label("Enter Name:");
        l.setBounds(50, 50, 100, 30);
        // Create TextField
        TextField tf = new TextField();
        tf.setBounds(160, 50, 150, 30);

        // Create Button
        Button b = new Button("Click Me");
        b.setBounds(120, 100, 80, 30);

        // Add action to button
        b.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String name = tf.getText();
                f.setTitle("Hello " + name);
            }
        });

        // Add components to Frame
        f.add(l);
        f.add(tf);
        f.add(b);

        // Frame settings
        f.setSize(400, 200);
        f.setLayout(null); // No default layout
        f.setVisible(true);

        // Close on exit
        f.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent we) {
                System.exit(0);
            }
        });
    }
}
```

Advantages of AWT

1. **Simple and Easy to Use**
    a. Beginner-friendly, straightforward API for creating GUI.
2. **Part of Core Java**

   a. Comes with the standard JDK (`java.awt` package), no external setup needed.

3. **Uses Native OS Components**
   a. AWT is "**heavyweight**" → it uses the platform's native widgets (buttons, menus, etc.), so it looks consistent with the OS UI.
4. **Better Performance in Some Cases**
   a. Since it directly uses system resources, some operations can be faster.
5. **Good for Small Desktop Tools**
   a. Handy for simple applications like notepads, calculators, or small admin tools.
6. **Event Handling Support**
   a. Provides a strong **event delegation model** (e.g., `ActionListener`, `MouseListener`, etc.) to handle user interactions.

Disadvantages of AWT

1. **Platform Dependent (Non-Portable Look & Feel)**
   a. UI appearance changes across operating systems (Windows, Linux, Mac).
2. **Limited Set of Components**
   a. Only basic GUI components available (no advanced controls like tables, trees, tabbed panes).
3. **Heavyweight Components**
   a. Each AWT component has a corresponding native peer → consumes more system resources and can cause inconsistency.
4. **Not Suitable for Modern UI**
   a. Cannot create advanced, stylish, or rich GUIs compared to **Swing** or **JavaFX**.
5. **Less Flexible Layouts**
   a. Layout managers are harder to work with compared to newer frameworks.
6. **Threading Issues**
   a. GUI updates should run only on the **Event Dispatch Thread (EDT)**, making multi-threaded GUIs more complex.