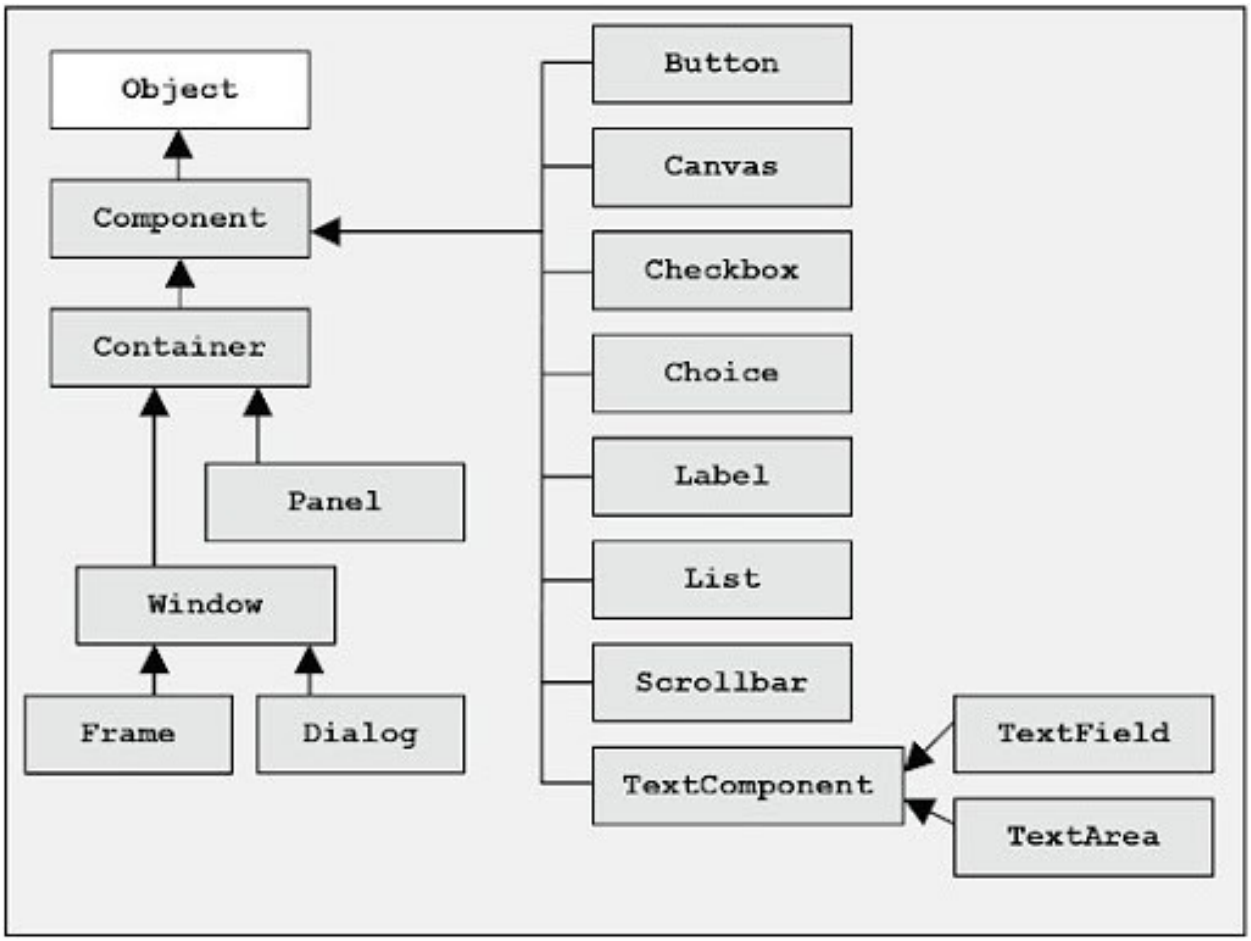
Java AWT(Abstract Windowing ToolKit)

What is AWT in Java?

AWT (Abstract Window Toolkit) is a set of **Java classes** used for creating **Graphical User Interface (GUI)** components. It is part of **Java's standard library (java.awt package)** and provides the basic tools to build **window-based applications**.



Key Features of AWT:

Feature	Description

Platform Independent	AWT is part of Java's standard API and works across platforms.
Heavyweight Components	AWT components are dependent on the native system's GUI resources (e.g., Windows buttons), meaning they are *"heavyweight"*.
Event-Driven	AWT uses the **Event Delegation Model** for handling events like clicks, key presses, etc.
Part of JDK	Built into the Java Development Kit (JDK) under the `java.awt` package.

Common AWT Components:

Component	Description
`Label`	Displays a single line of read-only text.
`Button`	Triggers an action when clicked.
`TextField`	Allows for single-line user input.
`TextArea`	Allows for multi-line user input.
`Checkbox`	Represents a checkbox that can be selected or deselected.
`CheckboxGroup`	Groups checkboxes to behave like radio buttons.
`Choice`	A drop-down list of items.
`List`	A scrollable list of items.
`Frame`	A top-level window with a title bar and border.
`Panel`	A generic container used to hold components.
`Canvas`	An area where custom graphics can be drawn.
`Scrollbar`	Adds scroll functionality to components.

Basic AWT Program Example:

```
import java.awt.*;

public class AWTExample {
    AWTExample() {
        Frame f = new Frame("AWT Example");  // Create a frame
```

```
        Label l = new Label("Enter Name:");  // Create label
        l.setBounds(50, 50, 100, 30);

        TextField tf = new TextField();      // Create text field
        tf.setBounds(160, 50, 150, 30);

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

        f.add(l);
        f.add(tf);
        f.add(b);

        f.setSize(400, 200);
        f.setLayout(null);  // No layout manager
        f.setVisible(true);
    }

    public static void main(String[] args) {
        new AWTExample();
    }
}
```

Advantages of AWT:

- Simple to use for basic GUIs
- Built-in support in Java
- Works cross-platform

Limitations of AWT:

- **Heavyweight**: Slower and less flexible due to native peer usage.
- **Limited GUI components** compared to Swing or JavaFX.
- Less attractive UIs compared to modern frameworks.

Here's a
complete and organized reference for Java AWT components

`java.awt.Button` Class

Class Declaration:

```
public class Button extends Component implements Accessible
```

Constructors:

Constructor	Description
`Button()`	Creates a button with no label.
`Button(String label)`	Creates a button with the specified label.

Important Methods:

Method	Description
`setLabel(String label)`	Sets the button’s label text.
`getLabel()`	Returns the current label text of the button.
`addActionListener(ActionListener l)`	Adds an action listener to the button.
`removeActionListener(ActionListener l)`	Removes the action listener.
`getActionCommand()`	Returns the action command string.
`setActionCommand(String command)`	Sets a custom action command string. Button Example:

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

public class ButtonExample {
    public static void main(String[] args) {
        Frame f = new Frame("Button Example");
```

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

        b.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                System.out.println("Button Clicked!");
            }
        });

        f.add(b);
        f.setSize(300, 200);
        f.setLayout(null);
        f.setVisible(true);
    }
}
```

`java.awt.Label` Class

Class Declaration:

```
public class Label extends Component implements Accessible
```

Constructors:

Constructor	Description
`Label()`	Creates an empty label.
`Label(String text)`	Creates a label with specified text.
`Label(String text, int alignment)`	Creates a label with specified text and alignment.

- `alignment`:
  - `Label.LEFT`
  - `Label.CENTER`

- `Label.RIGHT`

Method	Description
`setText(String text)`	Sets the label's text.
`getText()`	Returns the label's text.
`setAlignment(int alignment)`	Sets alignment (`Label.LEFT` , etc.).
`getAlignment()`	Returns the current alignment.

Label Example:

```
java
CopyEdit
import java.awt.*;

public class LabelExample {
    public static void main(String[] args) {
        Frame f = new Frame("Label Example");

        Label l1 = new Label("Hello AWT!", Label.CENTER);
        l1.setBounds(50, 100, 200, 30);

        f.add(l1);
        f.setSize(300, 200);
        f.setLayout(null);
        f.setVisible(true);
    }
}
```

`TextField` Class

Class Declaration:

```
public class TextField extends TextComponent implements Accessible
```

Constructors:

Constructor	Description
`TextField()`	Creates an empty text field.
`TextField(String text)`	Creates a text field with specified initial text.
`TextField(int columns)`	Creates an empty text field with specified number of columns.
`TextField(String text, int columns)`	Creates a text field with text and specified width.

Important Methods:

Method	Description
`setText(String text)`	Sets the text content.
`getText()`	Returns the text content.
`setEchoChar(char c)`	Masks input (e.g., for passwords).
`getEchoChar()`	Gets the masking character.
`setEditable(boolean)`	Enables or disables editing.
`isEditable()`	Checks if the field is editable.
`addActionListener(ActionListener l)`	Adds an action listener for Enter key.

Example:

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

public class TextFieldExample {
    public static void main(String[] args) {
        Frame f = new Frame("TextField Example");
```

```
        TextField tf = new TextField("Enter Name");
        tf.setBounds(100, 100, 150, 30);

        f.add(tf);
        f.setSize(400, 300);
        f.setLayout(null);
        f.setVisible(true);
    }
}
```

`Checkbox` Class

Class Declaration:

public class Checkbox extends Component implements ItemSelectable, Accessible
Constructors:

Constructor	Description
`Checkbox()`	Creates an empty checkbox.
`Checkbox(String label)`	Creates a checkbox with label.
`Checkbox(String label, boolean state)`	Creates a checkbox with label and initial state.
`Checkbox(String label, boolean state, CheckboxGroup group)`	Creates a radio-style checkbox.

Important Methods:

Method	Description
`setLabel(String)`	Sets the label.
`getLabel()`	Gets the label.
`setState(boolean)`	Sets checked or unchecked.

`getState()`	Returns checked (true/false).
`addItemListener(ItemListener l)`	Registers listener for state changes.
`setCheckboxGroup(CheckboxGroup g)`	Sets group for radio button behavior.

Example:

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

public class CheckboxExample {
    public static void main(String[] args) {
        Frame f = new Frame("Checkbox Example");

        Checkbox cb1 = new Checkbox("Java");
        cb1.setBounds(100, 100, 80, 30);

        Checkbox cb2 = new Checkbox("Python");
        cb2.setBounds(100, 130, 80, 30);

        f.add(cb1);
        f.add(cb2);

        f.setSize(300, 250);
        f.setLayout(null);
        f.setVisible(true);
    }
}
```

`Choice` Class

Class Declaration:

```
public class Choice extends Component implements ItemSelectable, Accessib
le
```

Constructors:

Constructor	Description
`Choice()`	Creates a dropdown list.

Important Methods:

Method	Description
`add(String item)`	Adds an item to the dropdown.
`getItem(int index)`	Gets item at index.
`getItemCount()`	Gets total number of items.
`getSelectedItem()`	Returns the selected item.
`select(int index)`	Selects item at index.
`addItemListener(ItemListener l)`	Listens for selection changes.

Example:

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

public class ChoiceExample {
    public static void main(String[] args) {
        Frame f = new Frame("Choice Example");

        Choice c = new Choice();
        c.setBounds(100, 100, 100, 30);

        c.add("C++");
        c.add("Java");
```

```
        c.add("Python");

        f.add(c);
        f.setSize(300, 200);
        f.setLayout(null);
        f.setVisible(true);
    }
}
```

`TextArea` Class

Class Declaration:

```
public class TextArea extends TextComponent implements Accessible
```

Constructors:

Constructor	Description
`TextArea()`	Creates an empty text area.
`TextArea(String text)`	Creates a text area with specified text.
`TextArea(int rows, int columns)`	Creates with given rows and columns.
`TextArea(String text, int rows, int columns)`	Text with specified rows and columns.
`TextArea(String text, int rows, int columns, int scrollbars)`	Includes scroll bar settings.

Important Methods:

Method	Description
`append(String text)`	Appends text to the end.
`insert(String text, int position)`	Inserts text at specified position.
`replaceRange(String str, int start, int end)`	Replaces text in given range.
`getRows()`, `getColumns()`	Get dimensions.
`setText(String text)`, `getText()`	Set/get content.

Example:

```
import java.awt.*;

public class TextAreaExample {
    public static void main(String[] args) {
        Frame f = new Frame("TextArea Example");

        TextArea ta = new TextArea("Welcome to AWT TextArea", 5, 20);
        ta.setBounds(50, 50, 250, 100);

        f.add(ta);
        f.setSize(400, 250);
        f.setLayout(null);
        f.setVisible(true);
    }
}
```

`List` Class

Class Declaration:

```
public class List extends Component implements ItemSelectable, Accessible
```

Constructors:

Constructor	Description
`List()`	Creates an empty single-selection list.
`List(int rows)`	With specified visible rows.
`List(int rows, boolean multipleMode)`	Allows multiple selections if true.

Important Methods:

Method	Description
`add(String item)`	Adds an item.
`add(String item, int index)`	Adds at index.
`getItem(int index)`	Returns item at index.
`getSelectedItem()`	Gets selected item (single).
`getSelectedItems()`	Gets all selected items.
`select(int index)`	Selects item by index.

```
import java.awt.*;

public class ListExample {
    public static void main(String[] args) {
        Frame f = new Frame("List Example");

        List list = new List(3, true);
        list.setBounds(100, 100, 100, 75);

        list.add("Java");
        list.add("C++");
        list.add("Python");

        f.add(list);
        f.setSize(300, 200);
        f.setLayout(null);
        f.setVisible(true);
    }
}
```

`Scrollbar` Class

Class Declaration:

```
public class Scrollbar extends Component implements Adjustable, Accessible
```

Constructors:

Constructor	Description
`Scrollbar()`	Creates a vertical scrollbar.
`Scrollbar(int orientation)`	Orientation: `Scrollbar.HORIZONTAL` or `Scrollbar.VERTICAL`.
`Scrollbar(int orientation, int value, int visible, int min, int max)`	Full control scrollbar.

Important Methods:

Method	Description
`setValues(int value, int visible, int min, int max)`	Sets all scrollbar values.
`getValue()`	Gets current value.
`setValue(int value)`	Sets current value.
`addAdjustmentListener(AdjustmentListener l)`	Listens for scroll events.

Example:

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

public class ScrollbarExample {
    public static void main(String[] args) {
        Frame f = new Frame("Scrollbar Example");

        Scrollbar s = new Scrollbar();
        s.setBounds(100, 100, 20, 100);
```

```
        f.add(s);
        f.setSize(300, 250);
        f.setLayout(null);
        f.setVisible(true);
    }
}
```

🔷 `Canvas` Class

✅ Class Declaration:

```
public class Canvas extends Component
```

✅ Constructors:

Constructor	Description
`Canvas()`	Creates a blank canvas for drawing.

✅ Commonly Overridden Methods:

Method	Description
`paint(Graphics g)`	Override this to draw on the canvas using `Graphics` class.
`update(Graphics g)`	Called when canvas is updated.

✅ Example:

```
import java.awt.*;

public class CanvasExample extends Canvas {
```

```
    public void paint(Graphics g) {
        g.setColor(Color.red);
        g.fillOval(75, 75, 100, 100);
    }

    public static void main(String[] args) {
        Frame f = new Frame("Canvas Example");

        CanvasExample c = new CanvasExample();
        c.setSize(250, 250);

        f.add(c);
        f.setSize(400, 400);
        f.setLayout(null);
        f.setVisible(true);
    }
}
```

🔷 `Panel` Class

✅ Class Declaration:

```
public class Panel extends Container implements Accessible
```

✅ Constructors:

Constructor	Description
`Panel()`	Creates a new blank panel.
`Panel(LayoutManager layout)`	Creates panel with specific layout.

✅ Methods:

Method	Description
`add(Component c)`	Adds a component to panel.
`setLayout(LayoutManager mgr)`	Sets layout (FlowLayout default).

✅ Example:

```
import java.awt.*;

public class PanelExample {
    public static void main(String[] args) {
        Frame f = new Frame("Panel Example");

        Panel p = new Panel();
        p.setBounds(50, 50, 200, 100);
        p.setBackground(Color.lightGray);

        Button b1 = new Button("OK");
        Button b2 = new Button("Cancel");

        p.add(b1);
        p.add(b2);

        f.add(p);
        f.setSize(400, 300);
        f.setLayout(null);
        f.setVisible(true);
    }
}
```

🔷 `Frame` Class

✅ Class Declaration:

```
public class Frame extends Window implements MenuContainer, Accessible
```

✅ Constructors:

Constructor	Description
`Frame()`	Creates a blank window.
`Frame(String title)`	Creates a window with a title.

✅ Important Methods:

Method	Description
`setSize(int width, int height)`	Sets the window size.
`setVisible(boolean)`	Shows/hides the window.
`add(Component c)`	Adds components.
`setLayout(LayoutManager)`	Sets layout.
`setTitle(String)`	Changes the window title.
`setBackground(Color)`	Sets background color.
`addWindowListener(WindowListener l)`	Closes frame properly.

✅ Example:

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

public class FrameExample {
    public static void main(String[] args) {
        Frame f = new Frame("Frame Example");
        f.setSize(400, 300);
```

```
        f.setLayout(null);

        f.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                f.dispose();
            }
        });

        f.setVisible(true);
    }
}
```