Experiment No.11
Aim: Basic programming constructs like branching and looping
Name: Aman Mehtar
Roll no: 32
Date of Performance: 25/9/24
Date of Submission: 9/8/24

Aim: Implement a program on Applet or AWT Controls

Objective:

To develop application like Calculator, Games, Animation using AWT Controls.

Theory:

Java AWT (Abstract Window Toolkit) is an API to develop Graphical User Interface (GUI) or windows-based applications in Java.

Java AWT components are platform-dependent i.e. components are displayed according to the view of operating system. AWT is heavy weight i.e. its components are using the resources of underlying operating system (OS).

The java.awt package provides classes for AWT API such as TextField, Label, TextArea, RadioButton, CheckBox, Choice, List etc.

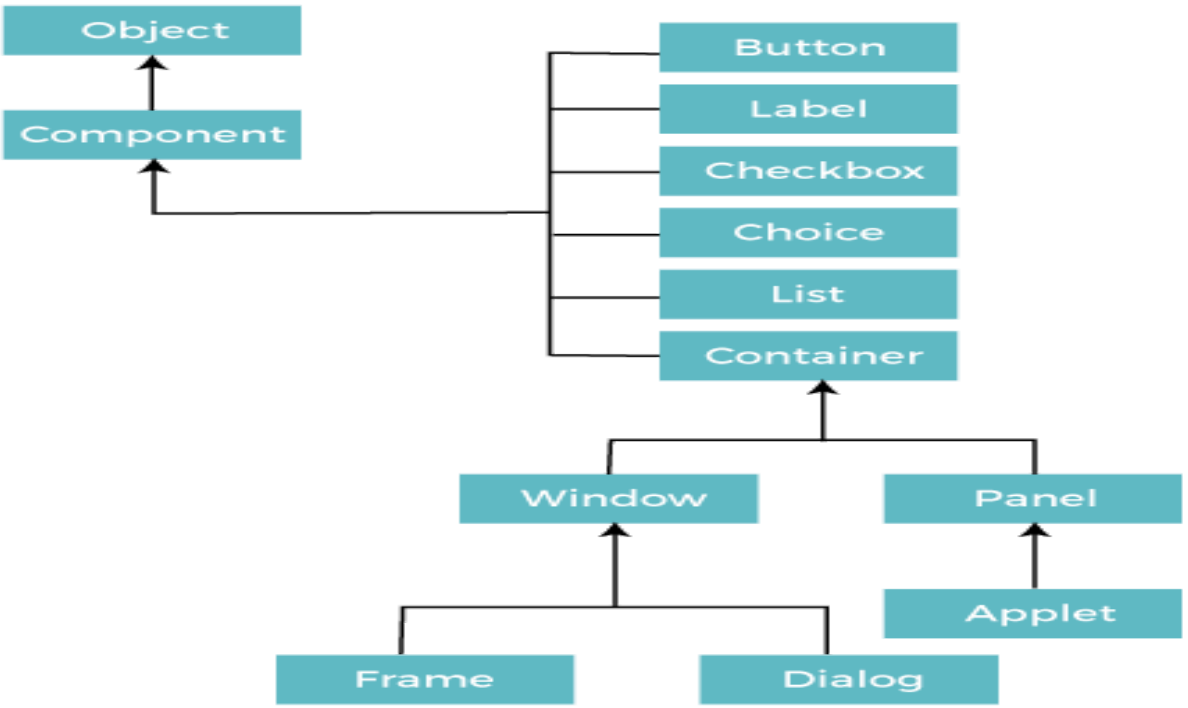
1. A general interface between Java and the native system, used for windowing, events and layout managers. This API is at the core of Java GUI programming and is also used by Swing and Java 2D. It contains the interface between the native windowing system and the Java application1.
2. A basic set of GUI widgets such as buttons, text boxes, and menus1. AWT also provides Graphics and imaging tools, such as shape, color, and font classes2. AWT also avails layout managers which helps in increasing the flexibility of the window layouts2

Java AWT calls the native platform calls the native platform (operating systems) subroutine for creating API components like TextField, ChechBox, button, etc.

For example, an AWT GUI with components like TextField, label and button will have different look and feel for the different platforms like Windows, MAC OS, and Unix. The reason for this is the platforms have different view for their native components and AWT directly calls the native subroutine that creates those components.

In simple words, an AWT application will look like a windows application in Windows OS whereas it will look like a Mac application in the MAC OS.

Java AWT Hierarchy



Code:

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

public class Calculator extends Frame implements ActionListener {
    TextField mainDisplay;
    Label historyDisplay; // For showing the previous input and operation
    String currentInput = "";
    String operator = "";
    double result = 0;

    public Calculator() {
        // Setting up the frame
        setLayout(new BorderLayout());
```

```
setTitle("Enhanced Calculator");
setSize(400, 600);
setResizable(false);

// History display (for showing the previous operations)
historyDisplay = new Label(" ", Label.RIGHT);
historyDisplay.setFont(new Font("Arial", Font.PLAIN, 18));
historyDisplay.setBackground(Color.LIGHT_GRAY);
add(historyDisplay, BorderLayout.NORTH);

// Main display for the current input
mainDisplay = new TextField();
mainDisplay.setFont(new Font("Arial", Font.BOLD, 24));
mainDisplay.setEditable(false);
add(mainDisplay, BorderLayout.CENTER);

// Panel for buttons
Panel buttonPanel = new Panel();
buttonPanel.setLayout(new GridLayout(4, 4, 10, 10));

// Adding buttons to the panel
String[] buttonLabels = {
    "7", "8", "9", "/",
    "4", "5", "6", "*",
    "1", "2", "3", "-",
    "0", "C", "=", "+"
};

for (String label : buttonLabels) {
    Button button = new Button(label);
    button.setFont(new Font("Arial", Font.PLAIN, 18));
    button.addActionListener(this);
    buttonPanel.add(button);
}
```

```
    add(buttonPanel, BorderLayout.SOUTH);

    setVisible(true);
}

@Override
public void actionPerformed(ActionEvent e) {
    String command = e.getActionCommand();

    if (command.charAt(0) >= '0' && command.charAt(0) <= '9') {
        // Number button pressed
        currentInput += command;
        mainDisplay.setText(currentInput);
    } else if (command.equals("C")) {
        // Clear button
        currentInput = "";
        operator = "";
        result = 0;
        historyDisplay.setText(" ");
        mainDisplay.setText("");
    } else if (command.equals("=")) {
        // Equals button
        calculate();
        operator = "";
        currentInput = String.valueOf(result);
        mainDisplay.setText(currentInput);
        historyDisplay.setText(" ");
    } else {
        // Operator button
        if (!currentInput.isEmpty()) {
            calculate();
            operator = command;
            historyDisplay.setText(result + " " + operator);
            currentInput = "";
        }
```

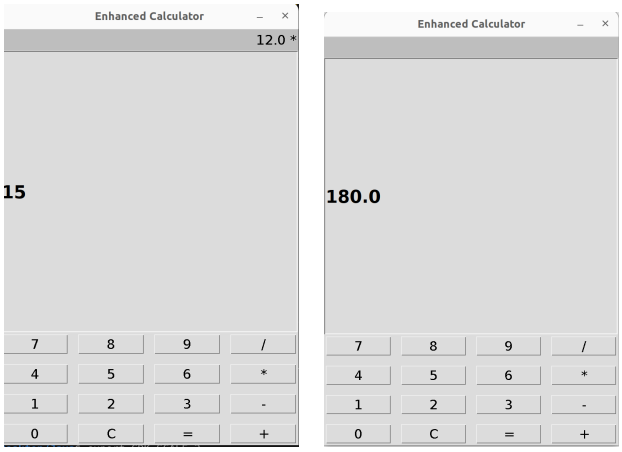
```
        }
    }

    private void calculate() {
        double input = currentInput.isEmpty() ? 0 :
Double.parseDouble(currentInput);

        switch (operator) {
            case "+":
                result += input;
                break;
            case "-":
                result -= input;
                break;
            case "*":
                result *= input;
                break;
            case "/":
                if (input != 0) {
                    result /= input;
                } else {
                    mainDisplay.setText("Error");
                }
                break;
            default:
                result = input;
        }
    }

    public static void main(String[] args) {
        Calculator calculator = new Calculator();
    }
}
```

Output:



Conclusion:

Using AWT controls to develop applications like this calculator demonstrates basic GUI handling in Java. AWT provides core widgets such as buttons and text fields, allowing for the creation of simple user interfaces. However, AWT components are platform-dependent, which means they may look different across various operating systems, making consistency in appearance a challenge. Additionally, AWT is considered "heavyweight," as it relies on the native system resources.