Experiment No. 11
Implement a program on Applet or AWT Controls
Date of Performance:
Date of Submission:

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.

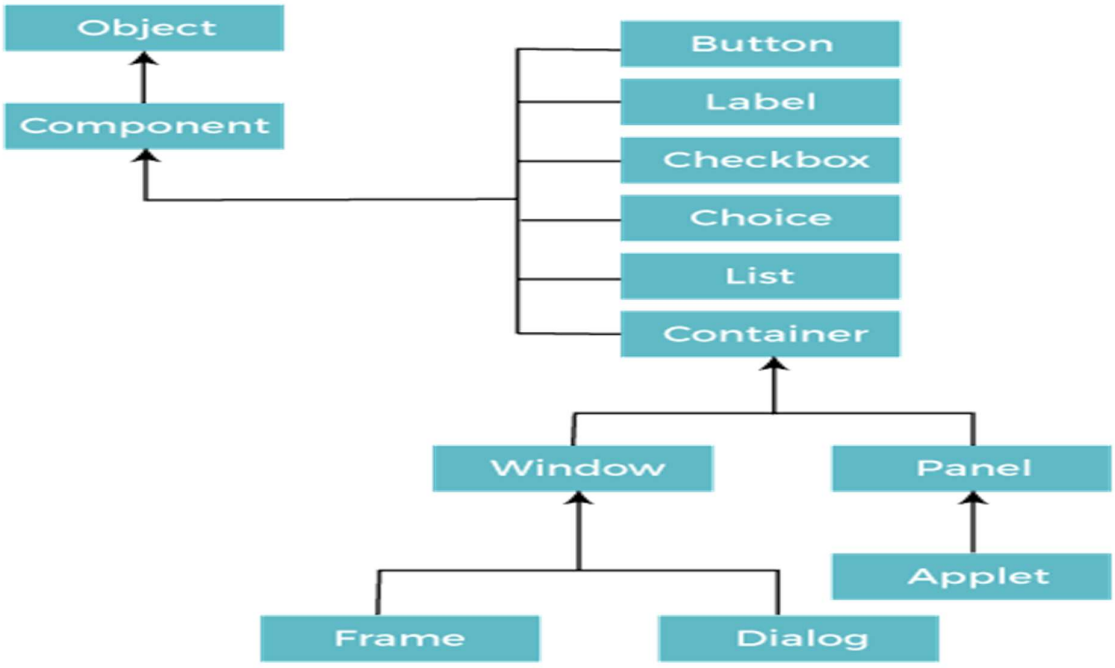
1. A general interface between Java and the native system, used for windowing, events and layout managers. This API is at the core of Java GUI programming and is also used by Swing and Java 2D. It contains the interface between the native windowing system and the Java application1.
2. A basic set of 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

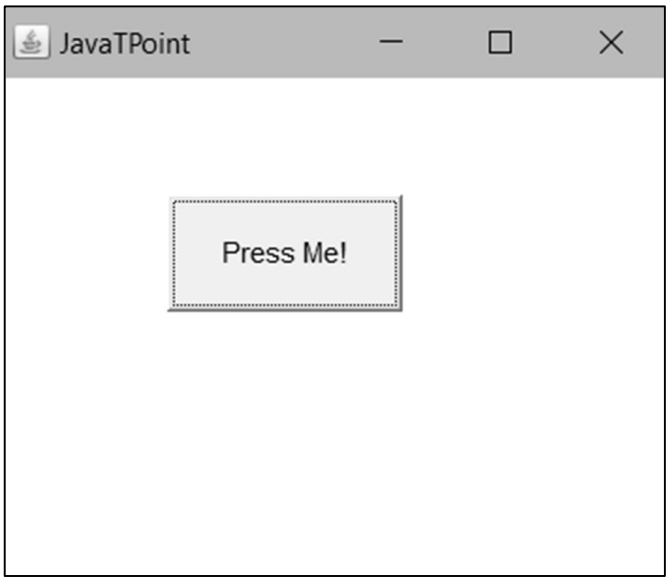


Code:

```
import java.awt.*;
public class AwtProgram {
public AwtProgram()
    {
Frame f = new Frame();
        Button btn=new Button("Press Me!");
        btn.setBounds(80, 80, 100, 50);
        f.add(btn);         //adding a new Button.
        f.setSize(300, 250);        //setting size.
        f.setTitle("JavaTPoint");  //setting title.
        f.setLayout(null);   //set default layout for frame.
        f.setVisible(true);           //set frame visibility true.
    }


public static void main(String[] args) {
// To-Do Auto-generated method stub

        AwtProgram awt = new AwtProgram();   //creating a frame.
    }
}
```

Output:

```
C:\Users\User.DESKTOP-VKOH6B7\Documents\Java Projects>javac AwtProgram.java

C:\Users\User.DESKTOP-VKOH6B7\Documents\Java Projects>java AwtProgram.java
```



Conclusion:

Comment on application development using AWT Controls.

Application development using AWT Controls is a way of creating graphical user interfaces (GUIs) in Java. AWT stands for Abstract Window Toolkit, which is a set of classes and interfaces that provide components, layouts, events, and graphics for GUI programming. AWT Controls are the basic elements of a GUI, such as buttons, labels, text fields, checkboxes, lists, menus, etc. AWT Controls can be used to create windows, dialogs, panels, and other containers that can hold and arrange other components. AWT Controls can also interact with the user through mouse clicks, keyboard inputs, and other events.

Some of the advantages of using AWT Controls for application development are:

- AWT Controls are platform-independent, meaning that they can run on any operating system that supports Java.
- AWT Controls are easy to use and implement, as they provide a simple and consistent API for creating and manipulating GUI components.
- AWT Controls are integrated with the Java language and the Java Virtual Machine (JVM), which means that they can use the features and benefits of Java, such as object-oriented programming, exception handling, multithreading, etc.

Some of the disadvantages of using AWT Controls for application development are:

- AWT Controls are platform-dependent in terms of their appearance and behavior, meaning that they may look and behave differently on different operating systems