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
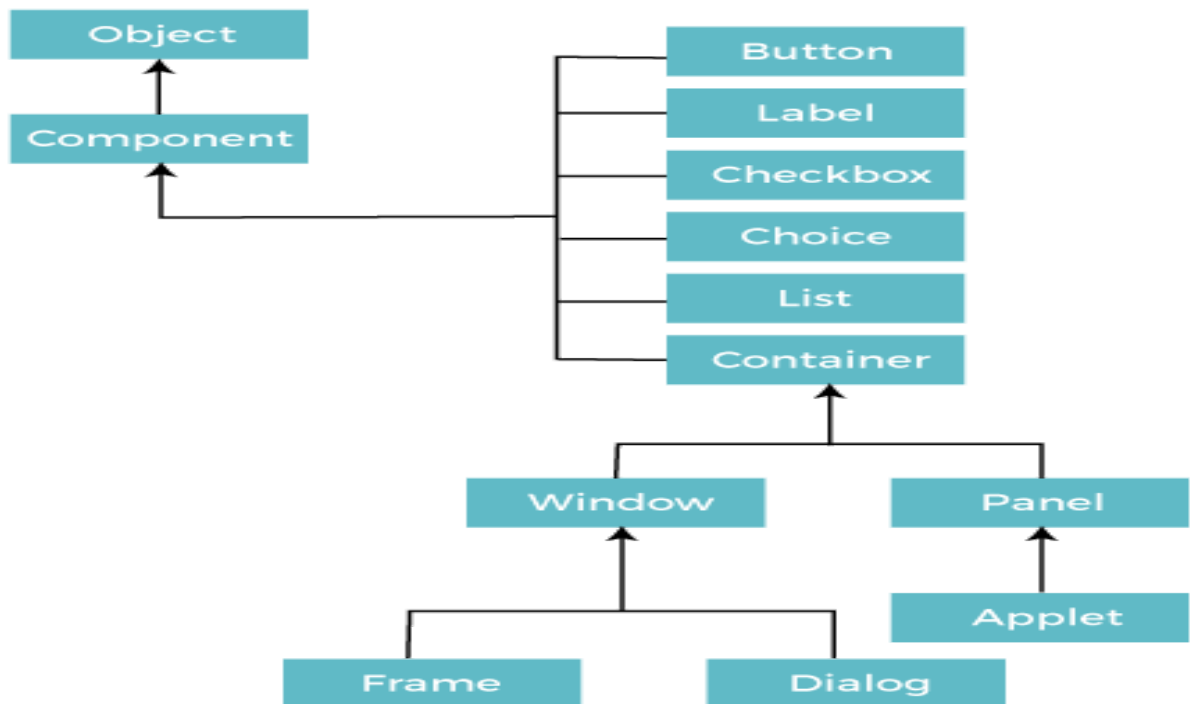
Code:
```
import java.awt.*;
public class AwtApp extends Frame
{
AwtApp()
{
```

```java
Label firstName = new Label("First Name");
firstName.setBounds(60, 70, 100, 40);
Label lastName = new Label("Last Name");
lastName.setBounds(40, 100, 100, 40);
Label dob = new Label("Date of Birth");
dob.setBounds(40, 130, 100, 30);
TextField firstNameTF = new TextField();
firstNameTF.setBounds(140, 70, 120, 40);
TextField lastNameTF = new TextField();
lastNameTF.setBounds(140, 100, 130, 40);
TextField dobTF = new TextField();
 dobTF.setBounds(140, 130, 120, 40);
Button sbmt = new Button("Submit");
sbmt.setBounds(40, 180, 120, 50);
Button reset = new Button("Reset");
reset.setBounds(140,180,120,50);
add(firstName);
add(lastName);
add(dob);
add(firstNameTF);
add(lastNameTF);
add(dobTF);
add(sbmt);
add(reset);
setSize(500,500);
setLayout(null);
setVisible(true);
}
public static void main(String[] args)
{
AwtApp awt = new AwtApp();
}
}
```

 Conclusion:

 Comment on application development using AWT Controls.

-Using AWT controls in object-oriented programming (OOP) for application development involves creating GUIs with component-based objects. AWT is event-driven, making it compatible with OOP principles like encapsulation and inheritance. You can customize components and use OOP to create custom layout managers. Implementing the MVC pattern with OOP helps maintain code structure. AWT provides cross-platform compatibility, but newer frameworks like Swing and JavaFX offer more

features and better performance. Understanding AWT can be valuable for legacy applications or as a foundation for more advanced GUI libraries.