Experiment No. 11
Implement a program on Applet or AWT Controls
Date of Performance: 9-10-2024
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.

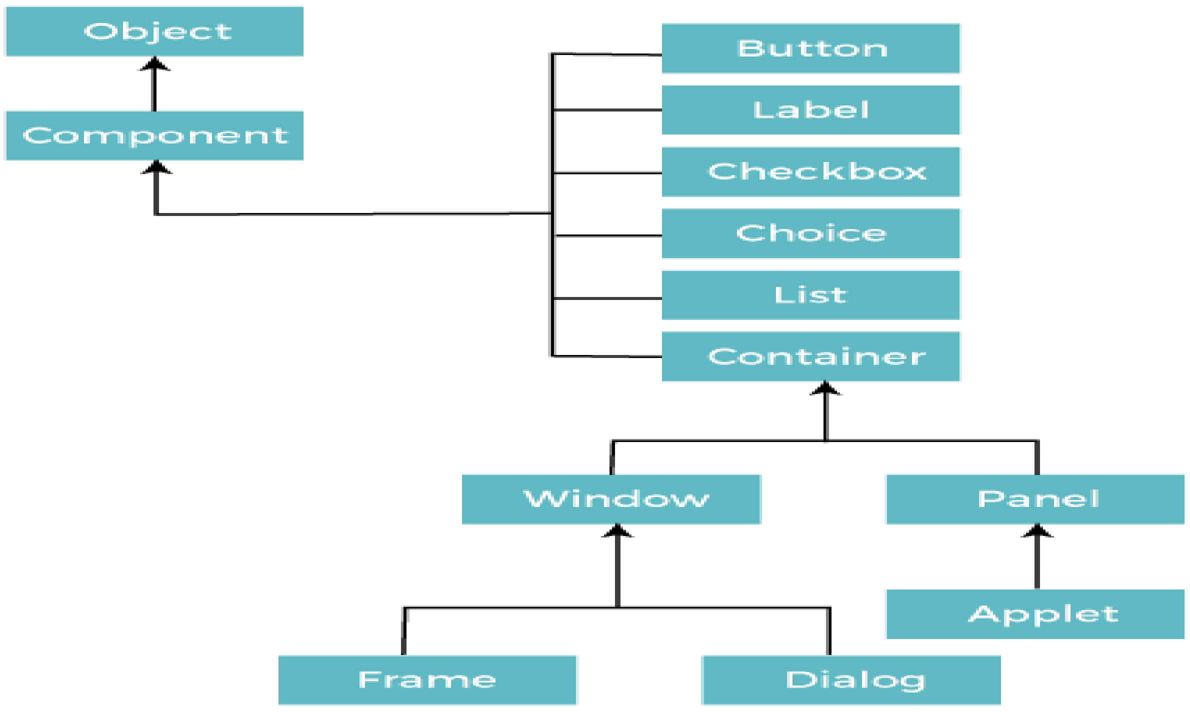
1. A general interface between Java and the native system, used for windowing, events and layout managers. This API is at the core of Java GUI programming and is also used by Swing and Java 2D. It contains the interface between the native windowing system and the Java application1.
2. A basic set of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

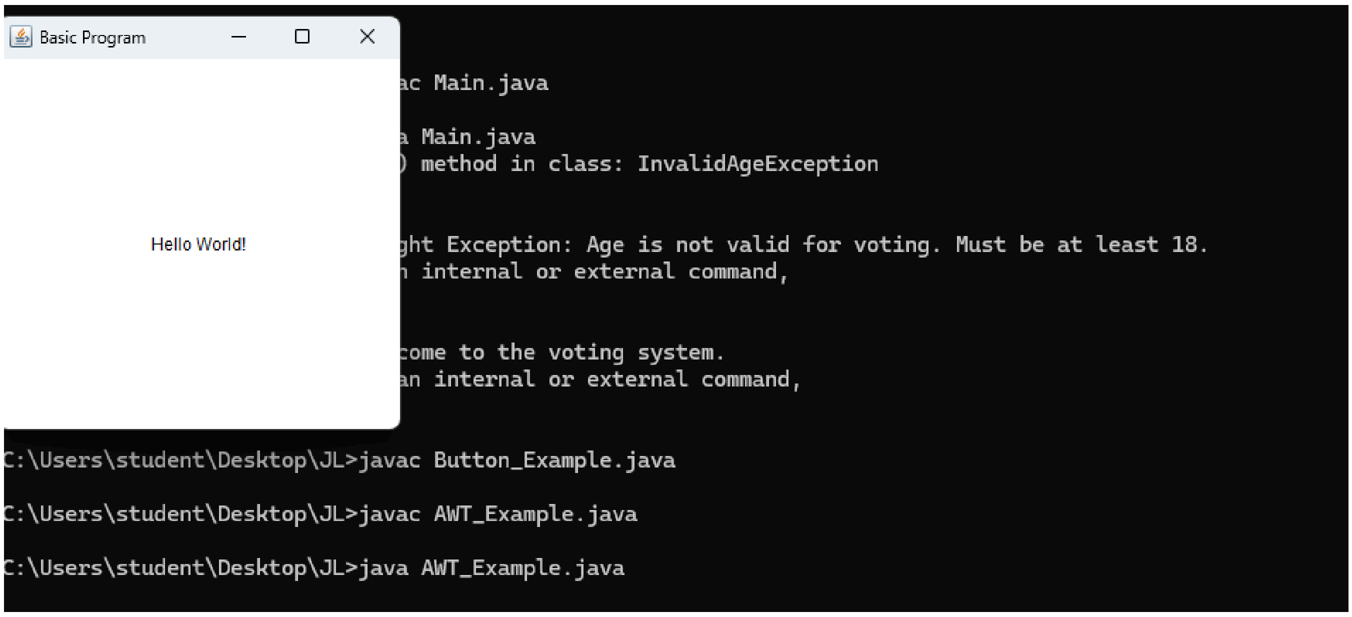


Code:

1. **Hello World AWT Program**

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

// Driver Class
public class AWT_Example {
    // main function
    public static void main(String[] args) {
        // Declaring a Frame and Label
        Frame frame = new Frame("Basic Program");
        Label label = new Label("Hello World!");

        // Aligning the label to CENTER
        label.setAlignment(Label.CENTER);

        // Adding Label and Setting the Size of the Frame
        frame.add(label);
        frame.setSize(300, 300);

        // Making the Frame visible
        frame.setVisible(true);

        // Using WindowListener for closing the window
        frame.addWindowListener(new WindowAdapter() {
            @Override
            public void windowClosing(WindowEvent e) {
                System.exit(0);
            }
        });
    }
}
```

Output:



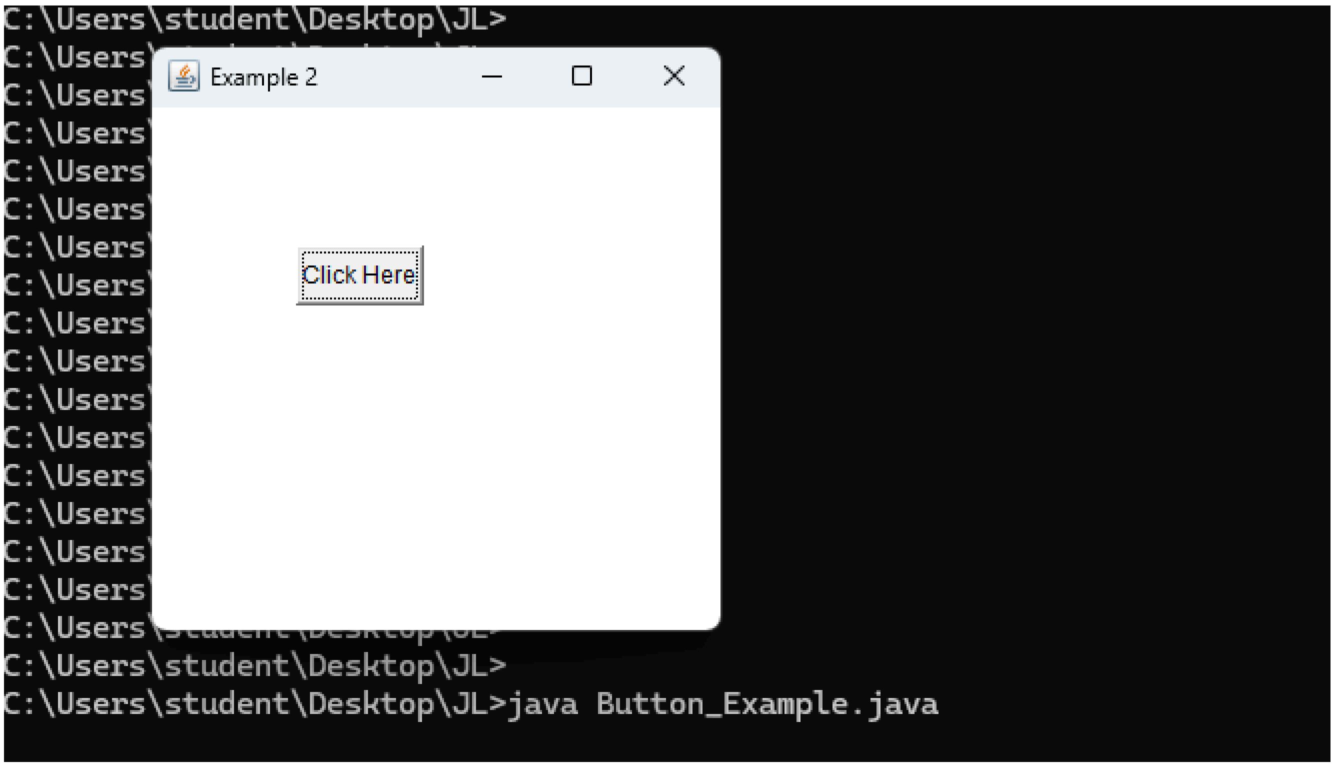
```
ac Main.java

a Main.java
) method in class: InvalidAgeException

ght Exception: Age is not valid for voting. Must be at least 18.
n internal or external command,

come to the voting system.
an internal or external command,

C:\Users\student\Desktop\JL>javac Button_Example.java

C:\Users\student\Desktop\JL>javac AWT_Example.java

C:\Users\student\Desktop\JL>java AWT_Example.java
```

Code :

2. Button AWT Program

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

// Driver Class
public class Button_Example {
    // main function
    public static void main(String[] args) {
        // Creating instance of frame with the label
        Frame frame = new Frame("Example 2");

        // Creating instance of button with label
        Button button = new Button("Click Here");
        // Setting the position for the button in frame
        button.setBounds(80, 100, 64, 30);

        // Adding button to the frame
        frame.add(button);

        // setting size, layout, and visibility of frame
        frame.setSize(300, 300);
        frame.setLayout(null);
        frame.setVisible(true);

        // Using WindowListener for closing the window
        frame.addWindowListener(new WindowAdapter() {
            @Override
            public void windowClosing(WindowEvent e) {
                System.exit(0);
            }
        });
    }
}
```

Output:



```
C:\Users\student\Desktop\JL>
C:\Users\student\Desktop\JL>
C:\Users\student\Desktop\JL>java Button_Example.java
```

Conclusion:

Comment on application development using AWT Controls.

→

Application Development Using AWT Controls

AWT (Abstract Window Toolkit) is a set of APIs provided by Java for building graphical user interfaces (GUIs) in Java applications. AWT controls are essential components that allow developers to create interactive interfaces, facilitating user interaction with applications. Here are some key points regarding the use of AWT controls in application development:

1. Rich User Interface Elements:

- AWT provides a variety of controls (also known as components), such as buttons, labels, text fields, checkboxes, radio buttons, lists, and more. These controls are the building blocks for creating user-friendly interfaces, enhancing user engagement.

2. Event Handling:

- AWT supports event-driven programming through its event-handling model. Developers can create listeners that respond to user actions like clicks, key presses, or window events. This allows for dynamic interaction, making applications more responsive to user inputs.

3. Cross-Platform Compatibility:

- Being part of Java, AWT applications are platform-independent. Code written using AWT can run on any operating system that supports Java, ensuring a consistent user experience across different environments.

4. Lightweight Components:

- AWT components are considered "heavyweight" because they rely on the native GUI of the operating system. This can lead to some inconsistencies in appearance across different platforms. However, they offer the advantage of being able to leverage platform-specific features.

5. Integration with Other Java Technologies:

- AWT can be easily integrated with other Java technologies, such as Swing (which provides more sophisticated and flexible components) and JavaFX (which allows for more modern UI development). This interoperability enables developers to use AWT where it's suitable and transition to other frameworks as needed.

6. Layout Management:

- AWT provides various layout managers (like FlowLayout, BorderLayout, GridLayout, etc.) to control the arrangement of components within a container. This helps in creating organized and visually appealing layouts without needing to manually calculate component positions.

7. Performance:

- Since AWT components are rendered using the native system's GUI components, they can perform better for simple applications compared to more complex GUI frameworks that use custom painting. However, for highly interactive applications, Swing or JavaFX may be preferred.