| **Experiment No. 15** |
| --- |
| Graphical user interfaces and handling user interactions |
| Date of Performance:03/10/24 |
| Date of Submission:10/10/24 |

**Aim:-**  To create graphical user interfaces and handle user interactions in Java applications using Java AWT and Event Handling

**Objective:-** Develop a Java application that creates a simple graphical user interface (GUI) using Java AWT. The GUI should include basic components such as buttons, text fields, and labels. Implement event handling to respond to user interactions, such as button clicks, by displaying a message or updating the text field content. This program should demonstrate the use of AWT for building GUIs and handling events to make the application interactive.

**Theory:-**

The Graphics class is the abstract base class for all graphics contexts that allow an application to draw onto components that are realized on various devices, as well as onto off-screen images.

Java provides a readymade package named awt(abstract windowing toolkit) which contains various classes to support GUI designing.

What is an Event?

Change in the state of an object is known as event i.e. event describes the change in state of source. Events are generated as result of user interaction with the graphical user interface components. For example, clicking on a button, moving the mouse, entering a character through keyboard, selecting an item from list, scrolling the page are the activities that causes an event to happen.

What is Event Handling?

Event Handling is the mechanism that controls the event and decides what should happen if an event occurs. This mechanism have the code which is known as event handler that is executed when an event occurs. Java Uses the Delegation Event Model to handle the events. This model defines the standard mechanism to generate and handle the events.Let's have a brief introduction to this model.

**Code :**

import java.awt.\*;

import java.awt.event.\*;

public class SimpleAWTGUI extends Frame implements ActionListener {

Label label;

TextField textField;

Button button;

SimpleAWTGUI() {

setTitle("College Application");

setLayout(new FlowLayout());

label = new Label("Enter your College name:");

textField = new TextField(20);

button = new Button("Submit");

add(label);

add(textField);

add(button);

button.addActionListener(this);

setSize(300, 150);

addWindowListener(new WindowAdapter() {

public void windowClosing(WindowEvent windowEvent) {

System.exit(0);

}

});

}

public void actionPerformed(ActionEvent e) {

String name = textField.getText();

if (!name.isEmpty()) {

Dialog dialog = new Dialog(this, "Message", true);

dialog.setLayout(new FlowLayout());

dialog.add(new Label("You're Studying in " + name + "."));

Button okButton = new Button("OK");

dialog.add(okButton);

dialog.setSize(200, 100);

dialog.setVisible(true);

okButton.addActionListener(ae -> dialog.setVisible(false));

}

}

public static void main(String[] args) {

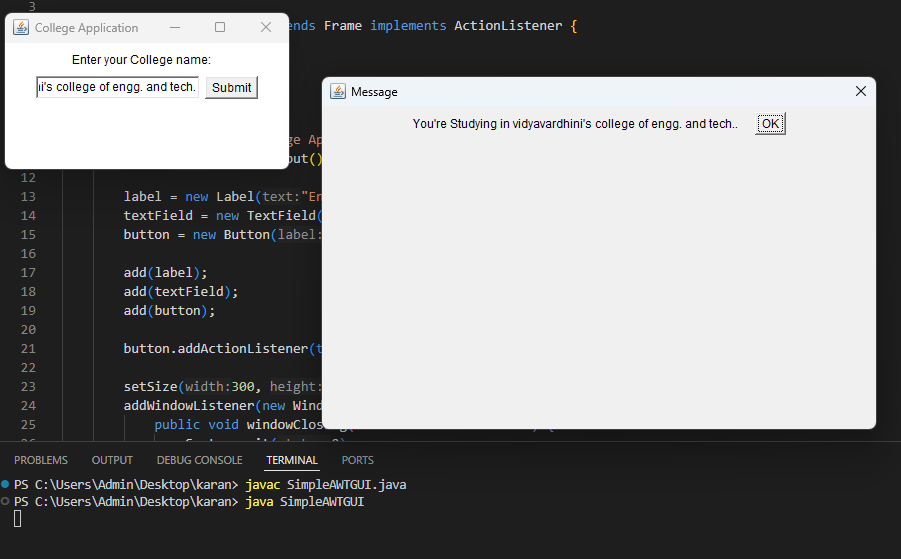
SimpleAWTGUI myFrame = new SimpleAWTGUI();

myFrame.setVisible(true);

}

}

**Output:**

****

**Conclusion:** Recent developments in Java GUI, particularly with JavaFX, emphasize modern design and enhanced functionality. JavaFX provides improved CSS styling, responsive layouts, and rich media support, making it easier to create visually appealing applications. Additionally, integration with web technologies and improved performance further strengthens its position in GUI development.