NARULA INSTITUTE OF TECHNOLOGY

PROJECT REPORT

Develop an AWT GUI application

Name: Ayan Bhattacherjee

Class roll :35

University roll :431221010035

Department: CSBS

ACKNOWLEDGEMENT

I would like to express my profound gratitude to Mr. Jayanta Paul, of CSE department, of Narula Institute of Technology university for the contributions to the completion of my project titled “Develop an AWT GUI application”.

I would like to express my special thanks to our mentor Mr. Jayanta Paul for his time and efforts he provided throughout the year. Your useful advice and suggestions were really helpful to me during the project’s completion. In this aspect, I am eternally grateful to you.
I would like to acknowledge that this project was completed entirely by me and not by someone else.

INTRODUCTION

This project involves developing a graphical user interface (GUI) application using Java's Abstract Window Toolkit (AWT) that allows users to vote for their favourite fruit. The application will present users with options to vote for three fruits: Apple, Banana, and Orange. After voting, the application will display the current vote counts for each fruit, providing immediate feedback on the popularity of each option.

Objectives:

1. Create a GUI Application: Utilize Java AWT to build a user-friendly interface where users can interact with voting options.
2. Voting Mechanism: Implement functionality that allows users to vote for one of the available fruits.
3. Vote Counting: Track and display the number of votes each fruit receives in real-time.
4. User Feedback: Update the vote counts dynamically as votes are cast, ensuring users can see the impact of their votes immediately.

Features:

- Voting Buttons: Three buttons, each labelled with one of the fruit options (Apple, Banana, Orange).
- Vote Count Display: Labels showing the number of votes for each fruit, updated in real-time as votes are cast.
- Window Management: Proper handling of the application window including closing the application gracefully.

Implementation Steps:

5. Set Up the Development Environment: Ensure Java and an IDE are installed for writing and executing the application.
6. Design the GUI: Create the layout using AWT components such as `Button`, `Label`, and `Frame`.
7. Implement Voting Logic: Define the logic to count votes and update the GUI components accordingly.
8. Handle User Interaction: Add event listeners to buttons to capture user actions and update vote counts.
9. Testing: Verify that the application works as expected, including correct vote counting and GUI updates.

This project will provide hands-on experience with Java AWT for GUI development and basic event handling, and it serves as a practical introduction to building interactive desktop applications.

IMPLEMENTATION

Develop an AWT GUI application where users can vote for their favourite fruit (e.g., Apple, Banana, Orange). Display the vote counts for each option.

```
import java.awt.*;
import java.awt.event.*;
public class FruitVotingApp extends Frame implements ActionListener {
    private int appleVotes = 0;
    private int bananaVotes = 0;
    private int orangeVotes = 0;
    private Label appleVoteLabel;
    private Label bananaVoteLabel;
    private Label orangeVoteLabel;
    public FruitVotingApp() {
        setTitle("Fruit Voting App");
        setLayout(new GridLayout(4, 2));
        Button appleButton = new Button("Vote for Apple");
        Button bananaButton = new Button("Vote for Banana");
        Button orangeButton = new Button("Vote for Orange");
        appleButton.addActionListener(this);
        bananaButton.addActionListener(this);
        orangeButton.addActionListener(this);
        add(appleButton);
        add(bananaButton);
        add(orangeButton);
        appleVoteLabel = new Label("Apple Votes: 0");
        bananaVoteLabel = new Label("Banana Votes: 0");
        orangeVoteLabel = new Label ("Orange Votes: 0");
```

```
        add(appleVoteLabel);
        add(bananaVoteLabel);
        add(orangeVoteLabel);

        setSize(300, 200);
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent windowEvent) {
                System.exit(0);
            }
        });
    }
    public void actionPerformed(ActionEvent e) {
        String action = e.getActionCommand();
        if (action.equals("Vote for Apple")) {
            appleVotes++;
            appleVoteLabel.setText("Apple Votes: " + appleVotes);
        } else if (action.equals("Vote for Banana")) {
            bananaVotes++;
            bananaVoteLabel.setText("Banana Votes: " + bananaVotes);
        } else if (action.equals("Vote for Orange")) {
            orangeVotes++;
            orangeVoteLabel.setText("Orange Votes: " + orangeVotes);
        }
    }
    public static void main(String[] args) {
        FruitVotingApp app = new FruitVotingApp();
        app.setVisible(true);
    }
}
```

CONCLUSION

The development of the Fruit Voting AWT GUI Application has successfully met its objectives of providing an interactive and user-friendly platform for voting on favorite fruits. Below is a summary of the project's achievements and outcomes:

Summary of Achievements:

10. Functional Voting Interface:
    - Created a GUI application using Java's Abstract Window Toolkit (AWT) that includes buttons for voting and labels for displaying vote counts.
    - Implemented a straightforward layout with `Button` and `Label` components to facilitate user interaction and vote tracking.
11. Real-time Vote Tracking:
    - Incorporated functionality to count votes for each fruit and update the vote counts in real-time as users make their selections.
    - Ensured that the application provides immediate feedback to users by updating vote counts dynamically on the interface.
12. User Interaction:
    - Added action listeners to the voting buttons to handle user interactions and process votes.
    - Provided a clear and intuitive user experience with responsive and easily accessible voting options.
13. Robust Design:
    - Designed the application to handle user actions effectively, ensuring smooth operation and accurate vote counting.
    - Managed the application window lifecycle, including handling window closure events gracefully to ensure proper termination.

Lessons Learned:

- AWT Component Usage: Gained practical experience in utilizing AWT components and layout managers to design and build a functional GUI application.
- Event Handling: Developed skills in handling user interactions and updating GUI elements in response to events.
- Real-time Updates: Implemented techniques for real-time data display and user feedback, crucial for interactive applications.

Future Improvements:

- Enhanced User Interface: Consider incorporating additional features such as graphical representations of vote counts (e.g., pie charts) to improve visual appeal.

- Extended Functionality: Add more voting options or features, such as user authentication or vote limits, to expand the application's capabilities.
- Cross-Platform Compatibility: Explore the use of Swing or JavaFX for a more modern and flexible GUI framework with additional features.

This project demonstrates the fundamental concepts of AWT and event-driven programming, providing a solid foundation for more complex GUI applications in Java.