Calculator Documentation

Introduction

Welcome to the Calculator Application Documentation! This comprehensive guide provides detailed insights into the calculator project, offering a deeper understanding of its core functionalities, technical aspects, and potential future improvements.

The calculator application serves as an intuitive and user-friendly tool for performing basic mathematical operations. Through the application of Object-Oriented Programming (OOP) principles, graphical user interface (GUI) design, and event-driven programming, the calculator ensures a seamless user experience.

Key Functionalities

Core Functionalities

Basic Arithmetic Operations: Addition, subtraction, multiplication, and division.

Decimal Point: Capability to input decimal numbers.

Clear and Delete: Options to clear the entire input or delete the last character.

Negative Numbers: Ability to change the sign of the current number.

Error Handling: Graceful handling of invalid inputs.

User Interactions and Expected Outcomes

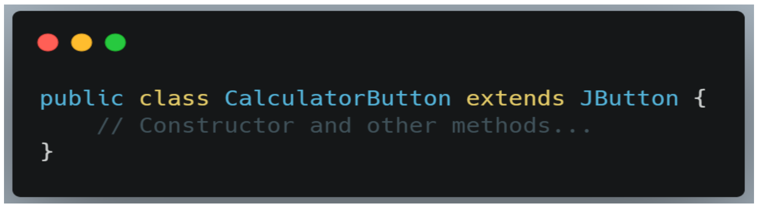
- **Button Clicks**: Clicking numeric and operation buttons updates the input display.
- **Equals (=) Button**: Computes and displays the result of the entered expression.
- **Clear (Clr) Button**: Clears the input field.
- **Delete (Del) Button**: Removes the last entered character.
- **Negative (-) Button**: Changes the sign of the current input.

Methodology / Technical Part

-Object-Oriented Design

Abstraction

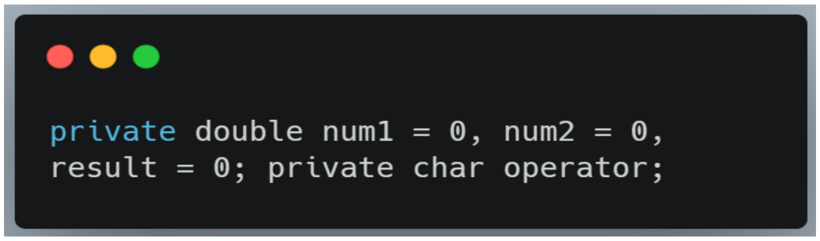
Abstraction is implemented in the **CalculatorButton** class, representing abstracted calculator buttons:



```
public class CalculatorButton extends JButton {
    // Constructor and other methods...
}
```

Encapsulation

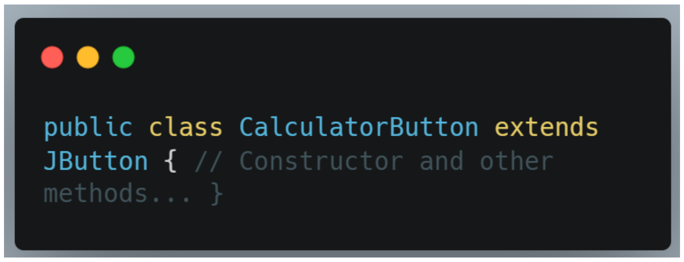
Encapsulation is applied with private variables encapsulating the internal state within the **CalculatorNew** class:



```
private double num1 = 0, num2 = 0,
result = 0; private char operator;
```

Inheritance

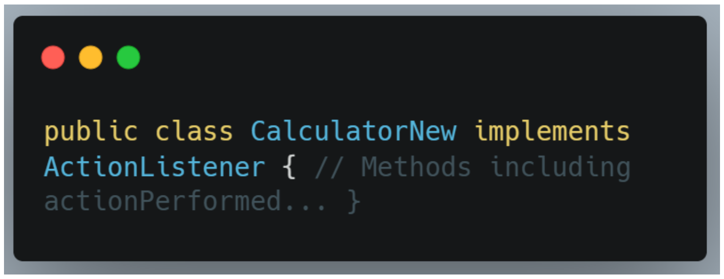
Inheritance is demonstrated through the extension of **JButton** in the **CalculatorButton** class:



```
public class CalculatorButton extends
JButton { // Constructor and other
methods... }
```

Polymorphism

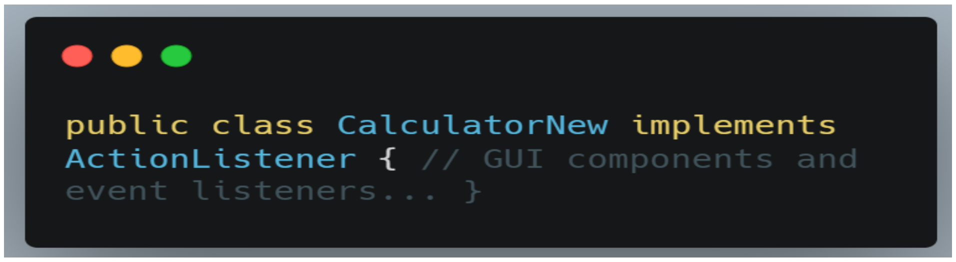
Polymorphism is showcased in the **CalculatorNew** class, implementing the **ActionListener** interface:



```
public class CalculatorNew implements
ActionListener { // Methods including
actionPerformed... }
```

GUI and Event-Driven Programming

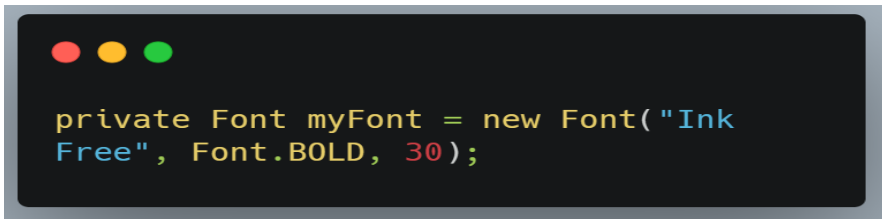
For GUI and event-driven programming, the Swing library is used for GUI components, and event listeners capture and respond to user interactions:



```
public class CalculatorNew implements
ActionListener { // GUI components and
event listeners... }
```

Technical Aspects

The technical aspects include font customization, dynamic button creation, and exception handling:



```
private Font myFont = new Font("Ink
Free", Font.BOLD, 30);
```

Summary

The Calculator Application successfully integrates Object-Oriented Programming (OOP) principles, GUI design, and event-driven programming to deliver a functional and user-friendly tool for basic mathematical operations. Core functionalities include basic arithmetic operations, decimal point input, clear/delete options, and error handling.

Future Enhancements

Looking ahead, our goal is for the calculator to have a use for much more difficult mathematical operations, including use in a scientific calculator and that may not be the only use developed