Advanced Calculator

Objective:

The objective of this project is to develop a simple graphical user interface (GUI) based calculator that can perform basic arithmetic operations — addition, subtraction, multiplication, and division and some advanced operations like square root and square — using Java Swing components.

Software Requirements:

Language: Java.

IDE: Eclipse / IntelliJ IDEA / NetBeans (or any text editor with Java).

Java Version: Java SE 8 or higher.

Tools and Technologies used:

Java Swing (for GUI)

Event Handling (ActionListener)

Exception Handling (for input errors)

Project Description:

The Advanced Calculator is a Java application that allows users to input two numbers and perform basic arithmetic operations and some advanced operations like square root and square.

The user interface includes:

- Two input fields for numbers
- Buttons for Add, Subtract, Multiply, Divide, and Clear
- A result field to display the output.
- Functions like square (x^2) and square root (√) can be applied.
- The = button performs calculation.
- Input can be cleared using the C button.
- Proper validations for non-numeric input and division by zero

The program uses ActionListener to perform the respective operations when a button is clicked. The result is displayed dynamically without restarting the program.

Features:

- Perform addition, subtraction, multiplication, and division

- Clear the input and output fields
- User-friendly graphical interface
- Error handling for invalid input and division by zero

CODE AND OUTPUT

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class AdvancedCalculator extends JFrame implements ActionListener {

    JTextField result;
    JButton[] numberButtons = new JButton[10];
    JButton addButton, subButton, mulButton, divButton, equalButton, clearButton,
sqrtButton, squareButton;
    JButton decimalButton; // Added for decimal point
    String currentInput = "";
    double firstNumber = 0;
    String operation = "";
    boolean secondOperand = false; // Flag to indicate if we are entering the second operand

    public AdvancedCalculator() {
        // Frame settings
        setTitle("Advanced Calculator");
        setSize(400, 500); // Increased size to fit more buttons
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setResizable(false); // Prevent resizing for a consistent look

        // Result text field
```

```
        result = new JTextField(15);
        result.setHorizontalAlignment(JTextField.RIGHT);
        result.setEditable(false);
        result.setFont(new Font("Arial", Font.PLAIN, 24)); // Increased font size

        // Number buttons
        for (int i = 0; i < 10; i++) {
            numberButtons[i] = new JButton(String.valueOf(i));
            numberButtons[i].addActionListener(this);
            numberButtons[i].setFont(new Font("Arial", Font.PLAIN, 20)); // Set font for number buttons
        }

        // Operation buttons
        addButton = new JButton("+");
        subButton = new JButton("-");
        mulButton = new JButton("*");
        divButton = new JButton("/");
        equalButton = new JButton("=");
        clearButton = new JButton("C");
        sqrtButton = new JButton("√");
        squareButton = new JButton("x²");
        decimalButton = new JButton("."); // Button for decimal point

        addButton.addActionListener(this);
        subButton.addActionListener(this);
        mulButton.addActionListener(this);
        divButton.addActionListener(this);
        equalButton.addActionListener(this);
        clearButton.addActionListener(this);
        sqrtButton.addActionListener(this);
```

```
squareButton.addActionListener(this);
decimalButton.addActionListener(this); // Add action listener for decimal button

// Set font for operation buttons
addButton.setFont(new Font("Arial", Font.PLAIN, 20));
subButton.setFont(new Font("Arial", Font.PLAIN, 20));
mulButton.setFont(new Font("Arial", Font.PLAIN, 20));
divButton.setFont(new Font("Arial", Font.PLAIN, 20));
equalButton.setFont(new Font("Arial", Font.PLAIN, 20));
clearButton.setFont(new Font("Arial", Font.PLAIN, 20));
sqrtButton.setFont(new Font("Arial", Font.PLAIN, 20));
squareButton.setFont(new Font("Arial", Font.PLAIN, 20));
decimalButton.setFont(new Font("Arial", Font.PLAIN, 20));

// Layout using GridBagLayout for more control
setLayout(new GridBagLayout());
GridBagConstraints gbc = new GridBagConstraints();
gbc.fill = GridBagConstraints.BOTH;
gbc.insets = new Insets(5, 5, 5, 5); // Add some padding

// Result field
gbc.gridx = 0;
gbc.gridy = 0;
gbc.gridwidth = 4;
gbc.weightx = 1; // Make it expand horizontally
gbc.weighty = 0;
add(result, gbc);

// Number buttons (using a loop)
gbc.gridwidth = 1;
```

```
gbc.weightx = 0.5; // Give buttons some weight
gbc.weighty = 0.5;
for (int i = 1; i <= 9; i++) {
    gbc.gridx = (i - 1) % 3; // 0, 1, 2
    gbc.gridy = 4 - (i - 1) / 3; // 3, 2, 1
    add(numberButtons[i], gbc);
}

// 0 button
gbc.gridx = 0;
gbc.gridy = 5;
add(numberButtons[0], gbc);

// Decimal button
gbc.gridx = 1;
gbc.gridy = 5;
add(decimalButton, gbc);

// Clear button
gbc.gridx = 2;
gbc.gridy = 5;
add(clearButton, gbc);

// Operator buttons
gbc.gridx = 3;
gbc.gridy = 1;
add(divButton, gbc);

gbc.gridx = 3;
gbc.gridy = 2;
```

```
    add(mulButton, gbc);
    gbc.gridx = 3;
    gbc.gridy = 3;
    add(subButton, gbc);
    gbc.gridx = 3;
    gbc.gridy = 4;
    add(addButton, gbc);
    gbc.gridx = 3;
    gbc.gridy = 5;
    add(equalButton, gbc);
    // sqrt button
    gbc.gridx = 0;
    gbc.gridy = 1;
    add(sqrtButton, gbc);
    // square button
    gbc.gridx = 1;
    gbc.gridy = 1;
    add(squareButton, gbc);
    setVisible(true);
}
public void actionPerformed(ActionEvent e) {
    String command = e.getActionCommand();

    if (command.equals("C")) {
        currentInput = "";
        result.setText("");
        firstNumber = 0;
        operation = "";
        secondOperand = false;
    } else if (command.equals("=")) {
```

```
if (operation.isEmpty() || currentInput.isEmpty()) return; // Nothing to calculate
try {
    double secondNumber = Double.parseDouble(currentInput);
    double finalResult = 0;
    switch (operation) {
        case "+":
            finalResult = firstNumber + secondNumber;
            break;
        case "-":
            finalResult = firstNumber - secondNumber;
            break;
        case "*":
            finalResult = firstNumber * secondNumber;
            break;
        case "/":
            if (secondNumber == 0) {
                JOptionPane.showMessageDialog(this, "Cannot divide by zero!");
                return;
            }
            finalResult = firstNumber / secondNumber;
            break;
    }
    result.setText(String.valueOf(finalResult));
    currentInput = String.valueOf(finalResult); // Store for chaining
    operation = ""; // Clear for new calculations
    secondOperand = false;
} catch (NumberFormatException ex) {
    JOptionPane.showMessageDialog(this, "Invalid input!");
    currentInput = "";
    result.setText("");
```

```
            secondOperand = false;
        }
    } else if (command.equals("+") || command.equals("-") || command.equals("*") ||
command.equals("/")) {
        if (!currentInput.isEmpty() && !secondOperand) {
            try {
                firstNumber = Double.parseDouble(currentInput);
                operation = command;
                currentInput = ""; // Clear for the next number
                secondOperand = true;
            } catch (NumberFormatException ex) {
                JOptionPane.showMessageDialog(this, "Invalid input!");
                currentInput = "";
                result.setText("");
                secondOperand = false;
            }
        } else if (secondOperand && !currentInput.isEmpty()) { //handle consecutive
operations
            try{
                double secondNumber = Double.parseDouble(currentInput);
                double tempResult = 0;
                 switch (operation) {
                    case "+":
                        tempResult = firstNumber + secondNumber;
                        break;
                    case "-":
                        tempResult = firstNumber - secondNumber;
                        break;
                    case "*":
                        tempResult = firstNumber * secondNumber;
                        break;
```

```
                case "/":
                    if (secondNumber == 0) {
                        JOptionPane.showMessageDialog(this, "Cannot divide by zero!");
                        return;
                    }
                    tempResult = firstNumber / secondNumber;
                    break;
                }
                firstNumber = tempResult;
                operation = command;
                currentInput = "";
                result.setText(String.valueOf(firstNumber));
            }catch(NumberFormatException ex){
                JOptionPane.showMessageDialog(this, "Invalid Input");
                currentInput = "";
                result.setText("");
                secondOperand = false;
                operation = "";
            }
        }
        else {
            operation = command;
        }
    } else if (command.equals("√")) {
        if (!currentInput.isEmpty()) {
            try {
                double number = Double.parseDouble(currentInput);
                if (number >= 0) {
                    result.setText(String.valueOf(Math.sqrt(number)));
                    currentInput = String.valueOf(Math.sqrt(number));
```

```
                } else {
                    JOptionPane.showMessageDialog(this, "Cannot take square root of a negative
number!");
                    currentInput = "";
                    result.setText("");
                }
            } catch (NumberFormatException ex) {
                JOptionPane.showMessageDialog(this, "Invalid input!");
                currentInput = "";
                result.setText("");
            }
        }
    } else if (command.equals("x²")) {
        if (!currentInput.isEmpty()) {
            try {
                double number = Double.parseDouble(currentInput);
                result.setText(String.valueOf(number * number));
                currentInput = String.valueOf(number * number);
            } catch (NumberFormatException ex) {
                JOptionPane.showMessageDialog(this, "Invalid input!");
  currentInput = "";
                result.setText("");
            }
        }
    } else if (command.equals(".")) { // Handle decimal point
        if (!currentInput.contains(".")) { // Only allow one decimal point
            currentInput += ".";
            result.setText(currentInput);
        }
    } else { // Number buttons
        currentInput += command;
```

```
            result.setText(currentInput);
        }
    }
    public static void main(String[] args) {
        new AdvancedCalculator();
    }
}
```

OUTPUT

Conclusion:

The Advanced Calculator project successfully demonstrates the use of Java Swing for building a basic GUI application along with event handling and exception handling. This project helped in understanding the basics of GUI development in Java.