Department of Computer Science and Engineering Varendra University

Assignment

Course Code: CSE 2101

Course Title: Object Oriented Design and Design Patterns

|  |  |  |
| --- | --- | --- |
| Assignment no | : | 1 |
| Topic Name | : | Write a Java program to create a Calculator Application with a Graphical User Interface (GUI). |

|  |  |
| --- | --- |
| Student Information | Course Teachers |
| Name: MD SOLAHIN MOLLA SUMON | 1. Name:MD NOUR NOBY |
| ID: 232311142 | Designation: Lecturer  Varendra University |
| Section: D |  |
| Semester: 3rd |  |
| Batch: 33rd |  |

|  |  |
| --- | --- |
| Signature of the Course Teacher | Obtained Grade/Marks |

**INTRODUCTION:**

This is a simple implementation of a calculator using Java Swing. It performs addition, subtraction, multiplication, division, modulus, and square root. The application provides a GUI to the user for input of numbers and operations, showing the result on the screen. The application also includes features like clearing the present input or going one step back, making it easy to use for simple calculations.

**SOURCE CODE:**

package Calculator;

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

public class Calculator implements ActionListener {

JFrame frame;

JTextField tfDisplay;

JButton[] numButtons = new JButton[10];

JButton addButton, subButton, mulButton, divButton, eqButton, clrButton, backButton, pointButton, sqrtButton, negateButton;

double num1 = 0, num2 = 0, result = 0;

String equation = "";

char operator = '\0';

public void createApp() {

frame = new JFrame("Calculator");

frame.setBounds(350, 150, 300, 520);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setResizable(false);

frame.setLayout(null);

frame.getContentPane().setBackground(new Color(0, 0, 0));

tfDisplay = new JTextField();

tfDisplay.setBounds(10, 10, 260, 50);

tfDisplay.setEditable(false);

tfDisplay.setFont(new Font("Arial", Font.BOLD, 20));

tfDisplay.setHorizontalAlignment(JTextField.RIGHT);

tfDisplay.setBackground(new Color(255, 255, 255));

tfDisplay.setForeground(new Color(0, 0, 0));

frame.add(tfDisplay);

Color buttonColor = new Color(255, 255, 255);

Color operatorButtonColor = new Color(255, 255, 255);

Color eqButtonColor = new Color(255, 255, 250);

Color clrButtonColor = new Color(255, 255, 250);

clrButton = new JButton("Clear");

backButton = new JButton("->Back");

clrButton.setBounds(10, 65, 100, 40);

backButton.setBounds(180, 65, 100, 40);

clrButton.addActionListener(this);

backButton.addActionListener(this);

clrButton.setBackground(clrButtonColor);

backButton.setBackground(operatorButtonColor);

clrButton.setForeground(Color.BLACK);

backButton.setForeground(Color.BLACK);

frame.add(clrButton);

frame.add(backButton);

for (int i = 0; i < 10; i++) {

numButtons[i] = new JButton(String.valueOf(i));

numButtons[i].addActionListener(this);

numButtons[i].setBackground(buttonColor);

numButtons[i].setForeground(Color.BLACK);

}

numButtons[7].setBounds(10, 130, 60, 50);

numButtons[8].setBounds(80, 130, 60, 50);

numButtons[9].setBounds(150, 130, 60, 50);

numButtons[4].setBounds(10, 190, 60, 50);

numButtons[5].setBounds(80, 190, 60, 50);

numButtons[6].setBounds(150, 190, 60, 50);

numButtons[1].setBounds(10, 250, 60, 50);

numButtons[2].setBounds(80, 250, 60, 50);

numButtons[3].setBounds(150, 250, 60, 50);

numButtons[0].setBounds(10, 310, 60, 50);

addButton = new JButton("+");

addButton.setBounds(220, 130, 60, 50);

subButton = new JButton("-");

subButton.setBounds(220, 190, 60, 50);

mulButton = new JButton("x");

mulButton.setBounds(220, 250, 60, 50);

divButton = new JButton("/");

divButton.setBounds(220, 310, 60, 50);

eqButton = new JButton("=");

eqButton.setBounds(150, 310, 60, 50);

pointButton = new JButton(".");

pointButton.setBounds(80, 310, 60, 50);

sqrtButton = new JButton("√");

sqrtButton.setBounds(150, 370, 130, 50);

negateButton = new JButton("+/-");

negateButton.setBounds(10, 370, 130, 50);

negateButton.addActionListener(this);

JButton[] operatorButtons = {addButton, subButton, mulButton, divButton, eqButton, pointButton, sqrtButton, negateButton};

for (JButton button : operatorButtons) {

button.addActionListener(this);

button.setBackground(operatorButtonColor);

button.setForeground(Color.BLACK);

}

for (JButton button : numButtons) {

frame.add(button);

}

frame.add(addButton);

frame.add(subButton);

frame.add(mulButton);

frame.add(divButton);

frame.add(eqButton);

frame.add(pointButton);

frame.add(sqrtButton);

frame.add(negateButton);

frame.setVisible(true);

}

@Override

public void actionPerformed(ActionEvent e) {

for (int i = 0; i < 10; i++) {

if (e.getSource() == numButtons[i]) {

equation += i;

tfDisplay.setText(equation);

}

}

if (e.getSource() == addButton) {

setOperator('+');

} else if (e.getSource() == subButton) {

setOperator('-');

} else if (e.getSource() == mulButton) {

setOperator('\*');

} else if (e.getSource() == divButton) {

setOperator('/');

} else if (e.getSource() == eqButton) {

calculateResult();

} else if (e.getSource() == clrButton) {

clearCalculator();

} else if (e.getSource() == backButton) {

if (!equation.isEmpty()) {

equation = equation.substring(0, equation.length() - 1);

tfDisplay.setText(equation);

}

} else if (e.getSource() == pointButton) {

if (!equation.contains(".")) {

equation += ".";

tfDisplay.setText(equation);

}

} else if (e.getSource() == sqrtButton) {

if (!equation.isEmpty()) {

double value = Double.parseDouble(equation);

if (value < 0) {

JOptionPane.showMessageDialog(frame, "Invalid Input for Square Root");

} else {

result = Math.sqrt(value);

tfDisplay.setText(String.valueOf(result));

equation = String.valueOf(result);

}

}

} else if (e.getSource() == negateButton) {

if (!equation.isEmpty()) {

try {

double value = Double.parseDouble(equation);

value = -value;

equation = String.valueOf(value);

tfDisplay.setText(equation);

} catch (NumberFormatException ex) {

JOptionPane.showMessageDialog(frame, "Invalid Number Format");

}

}

}

}

private void setOperator(char op) {

num1 = Double.parseDouble(equation);

operator = op;

equation = "";

}

private void calculateResult() {

num2 = Double.parseDouble(equation);

switch (operator) {

case '+' -> result = num1 + num2;

case '-' -> result = num1 - num2;

case '\*' -> result = num1 \* num2;

case '/' -> {

if (num2 == 0) {

JOptionPane.showMessageDialog(frame, "Division by Zero is Not Allowed");

return;

}

result = num1 / num2;

}

}

tfDisplay.setText(String.valueOf(result));

equation = String.valueOf(result);

num1 = result;

}

private void clearCalculator() {

tfDisplay.setText("");

equation = "";

num1 = num2 = result = 0;

operator = '\0';

}

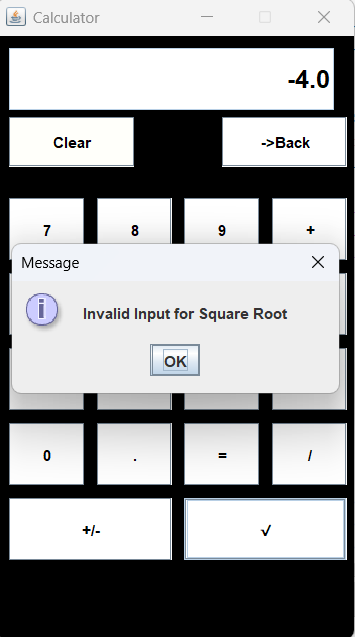
public static void main(String[] args) {

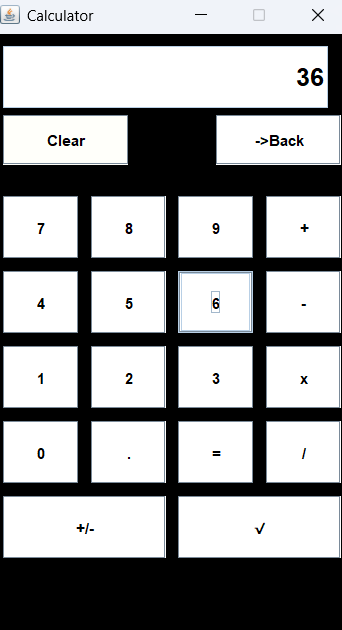
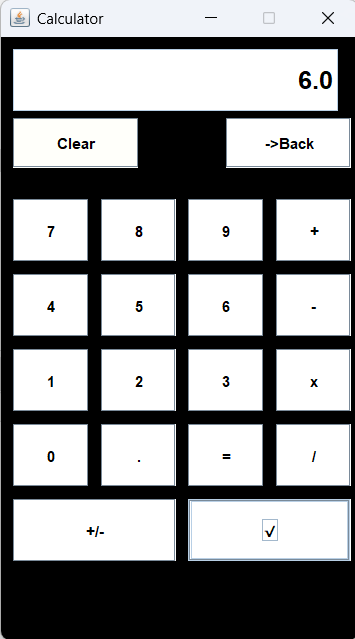
Calculator calculator = new Calculator();

calculator.createApp();

}

}

**OUTPUT:**

**using root button**

**LIMITATION OF THIS APPLICATION:**

1. Limited Mathematical Functions: The calculator supports only basic arithmetic operations (addition, subtraction, multiplication, division, modulus) and square root. More advanced functions like exponentiation, logarithms, and trigonometric functions are not available.
2. Single Calculation: This calculator only allows for one calculation at a time. Once a result is displayed, the user must clear the screen to perform another calculation.
3. No Input Validation for Complex Expressions: The application does not handle complex mathematical expressions or operator precedence (e.g., parentheses are not supported).
4. No Memory Functions: The calculator lacks memory functions like storing or recalling previous results.
5. No Decimal Precision Control: The application doesn't provide controls to set the precision of decimal points for floating-point operations.
6. No Scientific or Graphing Capabilities: This calculator is a basic model and does not include features like scientific calculations or graphing.
7. Error Handling: If users input invalid data or operations (e.g., division by zero or square root of negative numbers), the application shows a pop-up message, but does not have sophisticated error recovery.
8. This basic implementation is designed to serve as a simple tool for everyday arithmetic needs and can be extended in the future for more advanced use cases.