

## Green University of Bangladesh

**Department of Computer Science and Engineering (CSE) Faculty of Sciences and Engineering**

**Semester: (Spring, 2023), B.Sc. in CSE (Day)**

**LAB REPORT NO: 04**

**Course Title: Object Oriented Programing Lab**

**Course Code: CSE 202 Section: DE**

**Student Details**

|  |  |
| --- | --- |
| **Name** | **ID** |
| Md. Moshiur Rahman | 221902324 |

Submission Date : 10/05/2023

**Course Teacher’s Name** : **Dr. Muhammad Aminur Rahaman**

**[For Teachers use only: Don’t Write Anything inside this box]**

|  |
| --- |
| **Lab Report Status**  **Marks: ………………………………… Signature:..................... Comments:.............................................. Date:..............................** |

##### 1. TITLE OF THE LAB EXPERIMENT:

➤ Design and implement a scientific calculator

**2. OBJECTIVES**

➤ To gather knowledge of graphical user interface.

➤ To implement simple GUI using AWT and SWING on lab.

##### 3. ALGORITHM

Step-1 : Create a new Java class that extends the JFrame class to create a window for the calculator.

Step-2 : Inside the calculator window, add a text field to display the input and output of the calculator.

Step-3 : Add buttons to the calculator window for each of the basic arithmetic operations (addition, subtraction, multiplication, division) and scientific functions (sine, cosine, tangent, logarithm, power, and square root).

Step-4 : Implement the ActionListener interface to respond to button clicks on the calculator.

Step-5 : In the actionPerformed() method of the ActionListener interface, use a switch statement to determine which button was clicked and then perform the appropriate calculation based on the input and selected operation.

Step-6 : If the user enters invalid input, such as dividing by zero, display an error message in the text field.

Step-7 : Refresh the text field with the result of the calculation or error message.

**4. IMPLEMENTATION**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

public class ScientificCalculator extends JFrame implements ActionListener {

private JTextField inputField;

private JButton sinButton, cosButton, tanButton, sqrtButton, logButton, piButton, equalsButton;

private double num1, answer;

private int operator;

public ScientificCalculator() {

setTitle("Scientific Calculator");

setSize(300, 300);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLayout(new GridLayout(4, 3));

inputField = new JTextField();

sinButton = new JButton("sin");

cosButton = new JButton("cos");

tanButton = new JButton("tan");

sqrtButton = new JButton("sqrt");

logButton = new JButton("log");

piButton = new JButton("pi");

equalsButton = new JButton("=");

sinButton.addActionListener(this);

cosButton.addActionListener(this);

tanButton.addActionListener(this);

sqrtButton.addActionListener(this);

logButton.addActionListener(this);

piButton.addActionListener(this);

equalsButton.addActionListener(this);

add(sinButton);

add(cosButton);

add(tanButton);

add(sqrtButton);

add(logButton);

add(piButton);

add(inputField);

add(equalsButton);

setVisible(true);

}

public void actionPerformed(ActionEvent e) {

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

num1 = Double.parseDouble(inputField.getText());

answer = Math.sin(num1);

inputField.setText(Double.toString(answer));

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

num1 = Double.parseDouble(inputField.getText());

answer = Math.cos(num1);

inputField.setText(Double.toString(answer));

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

num1 = Double.parseDouble(inputField.getText());

answer = Math.tan(num1);

inputField.setText(Double.toString(answer));

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

num1 = Double.parseDouble(inputField.getText());

answer = Math.sqrt(num1);

inputField.setText(Double.toString(answer));

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

num1 = Double.parseDouble(inputField.getText());

answer = Math.log(num1);

inputField.setText(Double.toString(answer));

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

inputField.setText(Double.toString(Math.PI));

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

num1 = Double.parseDouble(inputField.getText());

inputField.setText(Double.toString(answer));

}

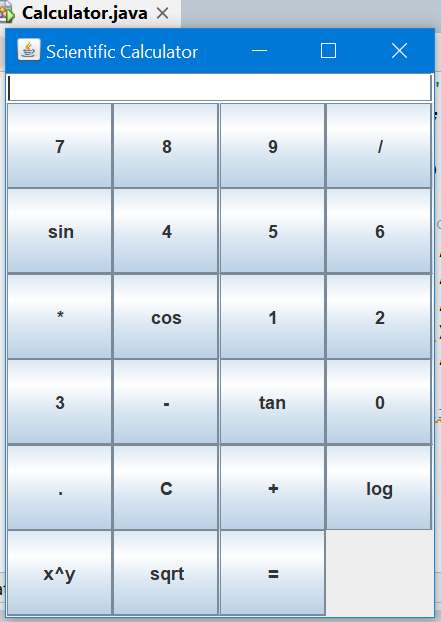
}

public static void main(String[] args) {

new ScientificCalculator();

}

**5. TEST RESULT**



**6. ANALYSIS & DISCUSSION**

The given code is an implementation of a simple scientific calculator using Java Swing. The calculator window contains a text field for input and output, and a grid of buttons for arithmetic operations and scientific functions.

The calculator class extends JFrame and implements the ActionListener interface, which allows it to handle button clicks on the calculator. The class has several instance variables, including the text field, the current input numbers (num1 and num2), the current operator, and the current result.

The constructor of the Calculator class sets up the calculator window, including the title, layout, size, and close operation. It then creates and adds the text field and button grid to the window, using the addButton method to add each button and set its action listener to the current instance of the Calculator class.

The addButton method is a private helper method that creates a new JButton with the given text and adds it to the given container with the current instance of the Calculator class as its action listener.

The actionPerformed method is called whenever a button is clicked on the calculator. It first retrieves the text value of the clicked button using the event's getActionCommand method. Then, it uses a switch statement to perform different actions based on the clicked button.

For numeric buttons (0-9 and decimal point), the actionPerformed method appends the button's value to the text field using the getText and setText methods.

For the clear button, the actionPerformed method simply sets the text field's text to an empty string.

For the basic arithmetic operation buttons (+, -, \*, and /), the actionPerformed method retrieves the current value of the text field as the first number (num1), sets the current operator based on the button clicked, and clears the text field for the second number (num2).

For the scientific function buttons (sin, cos, tan, log, x^y, and sqrt), the actionPerformed method retrieves the current value of the text field as the input number (num1), performs the corresponding function using the Math library, sets the result to the current value, and updates the text field with the result.

For the equals button, the actionPerformed method retrieves the current value of the text field as the second number (num2), performs the appropriate operation based on the current operator, and updates the text field with the result.

**7. SUMMARY**

Overall, the implementation of the scientific calculator in the given code is relatively simple and straightforward, but it covers a decent range of basic arithmetic operations and scientific functions. However, there are several limitations and potential issues with the code, such as lack of error handling for invalid inputs or division by zero, and limited support for complex mathematical operations.