

CALCULATOR



A PROJECT REPORT

Submitted by

MUTHU MEENA K (2303811724322073)

in partial fulfilment of requirements for the award of the course

CGB1201 – JAVA PROGRAMMING

in

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by
AICTE, New Delhi)

SAMAYAPURAM – 621 112

DECEMBER, 2024

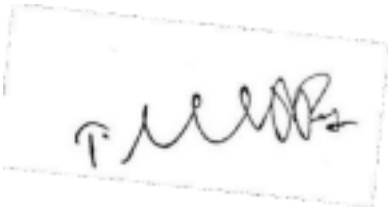
K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY (AUTONOMOUS)

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report on “**CALCULATOR**” is the bonafide work of

***Submitted by* MUTHU MEENA K (2303811724322073)** who carried out the project work during the academic year 2024 - 2025 under my supervision.



Signature

Dr. T. AVIDAIAPPAN M.E.,Ph.D.,

HEAD OF THE DEPARTMENT,

Department of Artificial Intelligence,

K. Ramakrishnan College of Technology,

Samayapuram, Trichy -621 112.



Signature

Mrs. S. GEETHA M.E.,

SUPERVISOR,

Department of Artificial Intelligence,

K. Ramakrishnan College of Technology,

Samayapuram, Trichy -621 112.

Submitted for the viva-voce examination held on 3.12.24



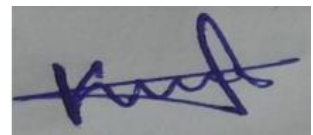
INTERNAL EXAMINER



EXTERNAL EXAMINER

DECLARATION

I declare that the project report on “**CALCULATOR**” is the result of original work done by me and best of our knowledge, similar work has not been submitted to “**ANNA UNIVERSITY CHENNAI**” for the requirement of Degree of **BACHELOR OF TECHNOLOGY**. This project report is submitted on the partial fulfillment of the requirement of the award of the **CGB1201 – JAVA PROGRAMMING**.



Signature

MUTHU MEENA K

Place: Samayapuram

Date: 3/12/2024

ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and indebtedness to our institution, **“K. Ramakrishnan College of Technology (Autonomous)”**, for providing us with the opportunity to do this project.

I extend our sincere acknowledgement and appreciation to the esteemed and honourable Chairman, **Dr. K. RAMAKRISHNAN, B.E.**, for having provided the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director, **Dr. S. KUPPUSAMY, MBA, Ph.D.**, for forwarding our project and offering an adequate duration to complete it.

I would like to thank **Dr. N. VASUDEVAN, M.TECH., Ph.D.**, Principal, who gave the opportunity to frame the project to full satisfaction.

I thank **Dr.T.AVUDAIAPPAN, M.E.,Ph.D.**, Head of the Department of **ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**, for providing her encouragement in pursuing this project.

I wish to convey our profound and heartfelt gratitude to our esteemed project guide **Mrs.S.GEETHA M.E.**, Department of **ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**, for her incalculable suggestions, creativity, assistance and patience, which motivated us to carry out this project.

I render our sincere thanks to the Course Coordinator and other staff members for providing valuable information during the course.

I wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

VISION OF THE INSTITUTION

To serve the society by offering top-notch technical education on par with global standards.

MISSION OF THE INSTITUTION

- Be a centre of excellence for technical education in emerging technologies by exceeding the needs of industry and society.
- Be an institute with world class research facilities.
- Be an institute nurturing talent and enhancing competency of students to transform them as all- round personalities respecting moral and ethical values.

VISION AND MISSION OF THE DEPARTMENT

To excel in education, innovation and research in Artificial Intelligence and Data Science to fulfill industrial demands and societal expectations.

Mission 1: To educate future engineers with solid fundamentals, continually improving teaching methods using modern tools.

Mission 2: To collaborate with industry and offer top-notch facilities in a conducive learning environment.

Mission 3: To foster skilled engineers and ethical innovation in AI and Data Science for global recognition and impactful research.

Mission 4: To tackle the societal challenge of producing capable professionals by instilling employability skills and human values.

PROGRAM EDUCATIONAL OBJECTIVES (PEOS)

PEO 1: Compete on a global scale for a professional career in Artificial Intelligence and Data Science.

PEO 2: Provide industry-specific solutions for the society with effective communication and ethics.

PEO 3: Hone their professional skills through research and lifelong learning initiatives.

PROGRAM OUTCOMES

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSOs)

- **PSO 1:** Capable of working on data-related methodologies and providing industry-focussed solutions.
- **PSO2:** Capable of analysing and providing a solution to a given real-world problem by designing an effective program.

ABSTRACT

This Java program provides a robust and user-friendly **Scientific Calculator** implemented using the AWT framework, designed to handle both **basic** (arithmetic operations, square root, square, reciprocal) and **scientific** (trigonometric functions like sin, cos, tan, logarithmic functions like log and ln, exponential, and constants such as π) calculations. The application features a **dynamic interface**, where the button layout updates seamlessly based on the selected mode—**Basic Mode** or **Scientific Mode**—accessible via a menu bar. It includes a **TextField** for displaying inputs and results, a grid-layout button panel for user interaction, and event-driven programming using the `ActionListener` interface to process button clicks.

Mathematical computations are powered by the **Math** library, ensuring precision for advanced functions. Thoughtful error handling is integrated to manage invalid inputs or operations such as division by zero. With its intuitive design, mode-switching capability, and comprehensive functionality, the calculator serves as a versatile tool for both everyday arithmetic and complex scientific computations.

TABLE OF CONTENTS

CHAPTER No.	TITLE	PAGE No.
	ABSTRACT	viii
1	INTRODUCTION	1
	1.1 INTRODUCTION	1
	1.2 OBJECTIVE	1
2	PROJECT METHODOLOGY	2
	2.1 PROPOSED WORK	2
	2.2 BLOCK DIAGRAM	2
3	JAVA PROGRAMMING CONCEPTS	3
	3.1 AWT (ABSTRACT WINDOW TOOLKIT)	3
	3.2 EVENT HANDLING IN AWT	4
4	MODULE DESCRIPTION	5
	4.1 BASIC OPERATIONS HANDLING MODULE	5
	4.2 SCIENTIFIC OPERATIONS HANDLING MODULE	5
5	CONCLUSION	6
	REFERENCES	7
	APPENDICES	8
	Appendix A – Source code	8
	Appendix B – Screen shots	15

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

The **Scientific Calculator** program is a graphical application developed in Java using the **AWT (Abstract Window Toolkit)** framework. It is designed to facilitate both **basic arithmetic operations** (such as addition, subtraction, multiplication, and division) and **scientific computations** (such as trigonometric functions, logarithms, exponentials, and constants like π). The application features an intuitive interface with dynamic functionality, allowing users to toggle between Basic and Scientific modes. By leveraging Java's event-driven programming model and the **Math** library, the program ensures precise and efficient calculations. It caters to diverse user needs, from simple daily arithmetic to complex scientific calculations.

1.2 OBJECTIVE

The primary objective of this program is to create a versatile and user-friendly calculator application that combines basic and advanced mathematical operations within a single platform. The program aims to demonstrate the practical implementation of Java's graphical and event-handling capabilities while providing users with accurate computational tools. The dynamic interface ensures accessibility for users of varying skill levels, with seamless mode-switching to suit different computational requirements. Additionally, the program emphasizes code efficiency, error handling, and extensibility, serving as both a functional tool and an educational example of graphical application development in Java.

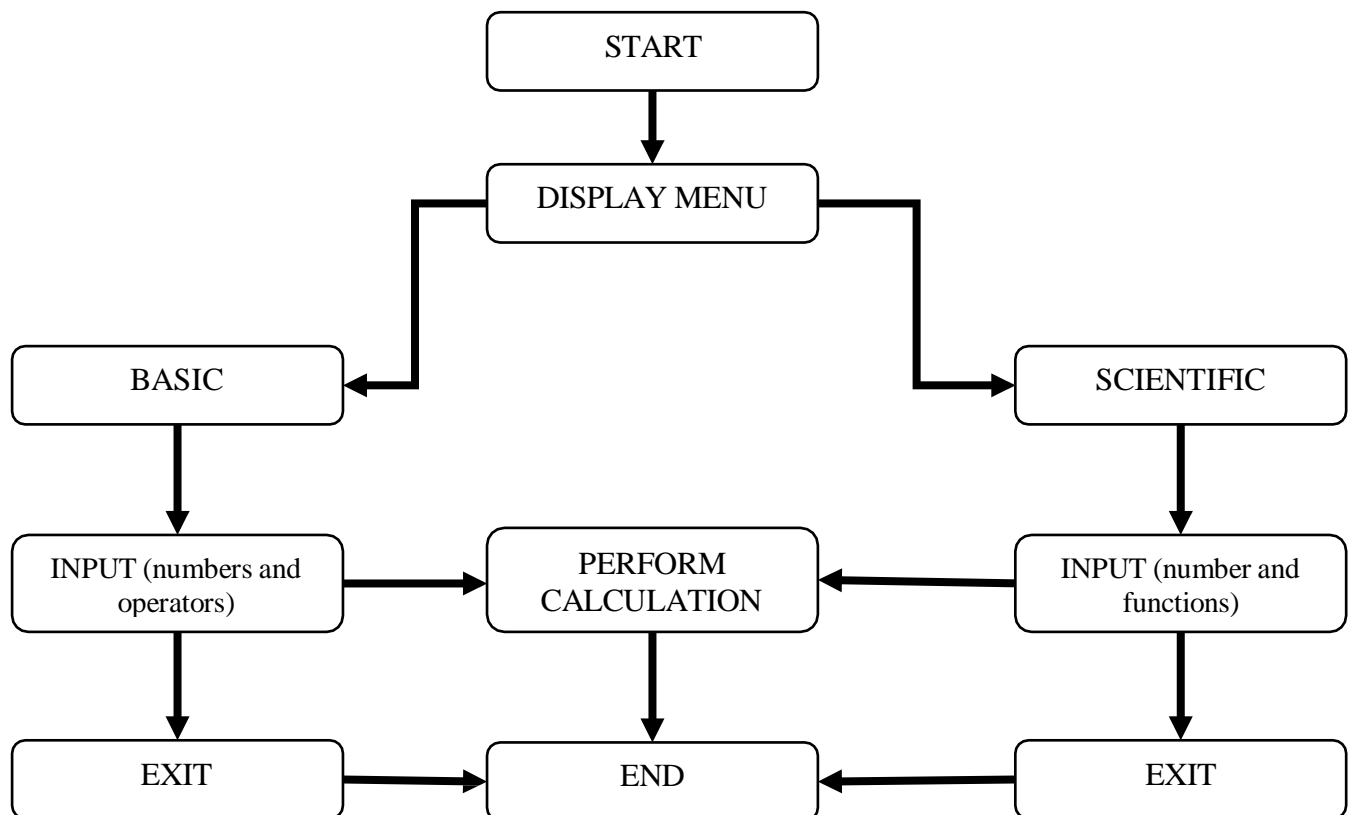
CHAPTER 2

PROJECT METHODOLOGY

2.1 PROPOSED WORK

The proposed work for the **Scientific Calculator** program involves the development of a robust, interactive, and feature-rich application that combines **basic arithmetic operations** with **advanced scientific computations** in a single platform. The program leverages the **Java AWT framework** to create a graphical user interface (GUI) that is intuitive and accessible. Users can toggle between **Basic Mode** and **Scientific Mode** through a menu bar, with the button layout dynamically updating to display the respective functionalities.

2.2 BLOCK DIAGRAM



CHAPTER 3

JAVA PROGRAMMING CONCEPTS

3.1 AWT (ABSTRACT WINDOW TOOLKIT)

The **Abstract Window Toolkit (AWT)** is one of the foundational frameworks in Java for building **Graphical User Interfaces (GUIs)**. It is part of the **Java Foundation Classes (JFC)** and provides a set of classes for creating windows, buttons, menus, and other graphical components. AWT is platform-dependent, meaning it uses the native GUI components of the underlying operating system to ensure a consistent look and feel.

1. Top-Level Containers:

- **Frame:** A top-level window with a title and border. It serves as the main window for the calculator.
- **Dialog:** A pop-up window, typically used for messages (not used in this program but available in AWT).

2. Basic UI Components:

- **Button:** Represents clickable buttons (e.g., calculator buttons like +, sin).
- **TextField:** A single-line text box used for input or displaying output.
- **Label:** Displays non-editable text (not used in this program but common in GUIs).

3. Panels and Layout Managers:

- **Panel:** A container for organizing components. The calculator uses a Panel with a GridLayout to arrange buttons.
- **Layout Managers:** Used to control the arrangement of components. Commonly used managers include:

- **BorderLayout**: Divides the container into five regions (North, South, East, West, Center).
- **GridLayout**: Arranges components in a grid structure.

4. Menu Components:

- **MenuBar**: A bar at the top of the frame to hold menus.
- **Menu**: A drop-down list of options (e.g., Basic Mode, Scientific Mode).
- **MenuItem**: Individual selectable options within a menu.

3.2 EVENT HANDLING IN AWT:

AWT follows the **event delegation model**, where events are captured and handled by **listeners**.

1. Event Classes:

- Represent various user interactions, such as:
 - **ActionEvent**: For button clicks or menu selections.
 - **WindowEvent**: For window actions like closing or minimizing.

2. Listener Interfaces:

- Interfaces to handle events:
 - **ActionListener**: Handles button clicks and menu selections.
 - **WindowListener** or **WindowAdapter**: Handles window-related events (e.g., closing the application).

3. **Example**: In the calculator program, the ActionListener interface is implemented to handle button and menu actions. The WindowAdapter class is used to manage the close action for the frame.

CHAPTER 4

MODULE DESCRIPTION

4.1 BASIC OPERATIONS HANDLING MODULE

- The Basic Operations Module is responsible for performing fundamental arithmetic calculations such as addition, subtraction, multiplication, and division.
- It allows the user to perform simple mathematical operations using two numbers and an operator.
- The module also handles error conditions like division by zero and invalid operator input.
- The **Basic Operations Handling Module** serves as the core of the calculator's Basic Mode, ensuring users can perform common mathematical tasks with ease. Its seamless integration with the GUI and error resilience make it a reliable foundation for the program.

4.2 SCIENTIFIC OPERATIONS HANDLING MODULE

- The Scientific Operations Module allows the user to perform more advanced mathematical calculations beyond basic arithmetic.
- This includes trigonometric functions (sin, cos, tan), exponential functions (exp), logarithms (log), and square roots (sqrt).
- The module takes a single number as input and applies the requested function.
- It also provides error handling for invalid inputs (e.g., non-positive values for logarithms or negative values for square roots).
- The **Scientific Operations Handling Module** significantly enhances the calculator's functionality, enabling it to serve as a practical tool for professionals, students, and researchers who require advanced mathematical computations.

CHAPTER 5

CONCLUSION

- The Scientific Calculator program is a comprehensive and user-friendly application that effectively combines basic arithmetic and advanced scientific computations in a single interface. By leveraging Java's **Abstract Window Toolkit (AWT)** and **Math library**, the program demonstrates a robust implementation of GUI-based event-driven programming. It provides users with essential features like addition, subtraction, and square root calculations in **Basic Mode** while offering advanced trigonometric, logarithmic, and exponential operations in **Scientific Mode**.
- The program emphasizes modularity, error handling, and dynamic UI updates, ensuring reliability and adaptability for various user requirements. Its ability to switch between basic and scientific functionalities highlights its versatility. This project not only showcases the power of Java for creating functional and intuitive desktop applications but also serves as a foundational framework for more complex computational tools in the future.
- Overall, this project not only demonstrates a strong grasp of Java programming concepts like GUI design, event handling, and the use of the Math library but also provides a practical tool for students, professionals, and anyone requiring quick and accurate mathematical computations. It is a testament to Java's capability for building reliable, interactive, and functional desktop applications.

REFERENCES:

1.Java Documentation – Oracle.

- The official Java documentation provides comprehensive information about the AWT library, Math class, and Java event handling mechanisms used in the program.
- URL: <https://docs.oracle.com/javase/8/docs/api/>

2.Oracle AWT Tutorial – Oracle.

- Oracle's tutorial on the Abstract Window Toolkit (AWT), which covers the components used in the program such as Frame, Panel, TextField, and Button, as well as event handling.
- URL: <https://docs.oracle.com/javase/tutorial/uiswing/>

3.Java Math Class Documentation – Oracle.

- The Math class in Java provides methods for performing basic numeric operations such as square root, logarithms, and trigonometric functions, all of which were utilized in this calculator program.
- URL: <https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html>

APPENDICES

APPENDIX A – SOURCE CODE

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

public class Calculator extends Frame implements ActionListener {
    private TextField display;
    private Panel buttonPanel;
    private boolean isScientificMode = false;

    private double operand1 = 0;
    private String operator = "";

    public Calculator() {
        // Frame setup
        setTitle("Scientific Calculator");
        setSize(400, 500);
        setLayout(new BorderLayout());
        setResizable(false);

        // Display setup
        display = new TextField();
        display.setFont(new Font("Arial", Font.BOLD, 20));
        display.setEditable(false);
        add(display, BorderLayout.NORTH);

        // Buttons panel
        buttonPanel = new Panel();
```

```

buttonPanel.setLayout(new GridLayout(5, 4, 5, 5));
add(buttonPanel, BorderLayout.CENTER);

// Add basic buttons initially
addBasicButtons();

// Menu setup for switching modes
MenuBar menuBar = new MenuBar();
Menu menu = new Menu("Mode");
MenuItem basicMode = new MenuItem("Basic Mode");
MenuItem scientificMode = new MenuItem("Scientific Mode");

basicMode.addActionListener(e -> {
    isScientificMode = false;
    updateButtons();
});

scientificMode.addActionListener(e -> {
    isScientificMode = true;
    updateButtons();
});

menu.add(basicMode);
menu.add(scientificMode);
menuBar.add(menu);
setMenuBar(menuBar);

// Close action
addWindowListener(new WindowAdapter() {

```

```

        public void windowClosing(WindowEvent e) {
            dispose();
        }
    });
}

```

```

private void addBasicButtons() {

```

```

    buttonPanel.removeAll();

```

```

    String[] basicButtons = {

```

```

        "7", "8", "9", "/",

```

```

        "4", "5", "6", "*",

```

```

        "1", "2", "3", "-",

```

```

        "0", ".", "=", "+",

```

```

        "C", "√", "x²", "1/x"

```

```

    };

```

```

    for (String text : basicButtons) {

```

```

        Button button = new Button(text);

```

```

        button.setFont(new Font("Arial", Font.BOLD, 16));

```

```

        button.addActionListener(this);

```

```

        buttonPanel.add(button);

```

```

    }

```

```

    buttonPanel.revalidate();

```

```

    buttonPanel.repaint();

```

```

}

```

```

private void addScientificButtons() {

```

```

    buttonPanel.removeAll();

```

```

    String[] scientificButtons = {

```

```

        "sin", "cos", "tan", "log",
        "exp", "ln", "π", "C",
        "7", "8", "9", "/",
        "4", "5", "6", "*",
        "1", "2", "3", "-",
        "0", ".", "=", "+"
    };

    for (String text : scientificButtons) {
        Button button = new Button(text);
        button.setFont(new Font("Arial", Font.BOLD, 16));
        button.addActionListener(this);
        buttonPanel.add(button);
    }
    buttonPanel.revalidate();
    buttonPanel.repaint();
}

private void updateButtons() {
    if (isScientificMode) {
        addScientificButtons();
    } else {
        addBasicButtons();
    }
}

@Override
public void actionPerformed(ActionEvent e) {
    String command = e.getActionCommand();

```

```

try {
    if ("C".equals(command)) {
        display.setText("");
        operand1 = 0;
        operator = "";
    } else if ("=".equals(command)) {
        calculate();
    } else if ("+".equals(command) || "-".equals(command) ||
        "*".equals(command) || "/".equals(command)) {
        operator = command;
        operand1 = Double.parseDouble(display.getText());
        display.setText("");
    } else if ("√".equals(command)) {
        double num = Double.parseDouble(display.getText());
        display.setText(String.valueOf(Math.sqrt(num)));
    } else if ("x²".equals(command)) {
        double num = Double.parseDouble(display.getText());
        display.setText(String.valueOf(Math.pow(num, 2)));
    } else if ("1/x".equals(command)) {
        double num = Double.parseDouble(display.getText());
        display.setText(String.valueOf(1 / num));
    } else if ("sin".equals(command)) {
        double num = Double.parseDouble(display.getText());
        display.setText(String.valueOf(Math.sin(Math.toRadians(num))));
    } else if ("cos".equals(command)) {
        double num = Double.parseDouble(display.getText());
        display.setText(String.valueOf(Math.cos(Math.toRadians(num))));
    } else if ("tan".equals(command)) {
        double num = Double.parseDouble(display.getText());

```

```

        display.setText(String.valueOf(Math.tan(Math.toRadians(num))));
    } else if ("log".equals(command)) {
        double num = Double.parseDouble(display.getText());
        display.setText(String.valueOf(Math.log10(num)));
    } else if ("exp".equals(command)) {
        double num = Double.parseDouble(display.getText());
        display.setText(String.valueOf(Math.exp(num)));
    } else if ("ln".equals(command)) {
        double num = Double.parseDouble(display.getText());
        display.setText(String.valueOf(Math.log(num)));
    } else if ("π".equals(command)) {
        display.setText(String.valueOf(Math.PI));
    } else {
        display.setText(display.getText() + command);
    }
} catch (Exception ex) {
    display.setText("Error");
}
}

```

```

private void calculate() {
    try {
        double operand2 = Double.parseDouble(display.getText());
        double result = 0;

        switch (operator) {
            case "+":
                result = operand1 + operand2;
                break;

```

```

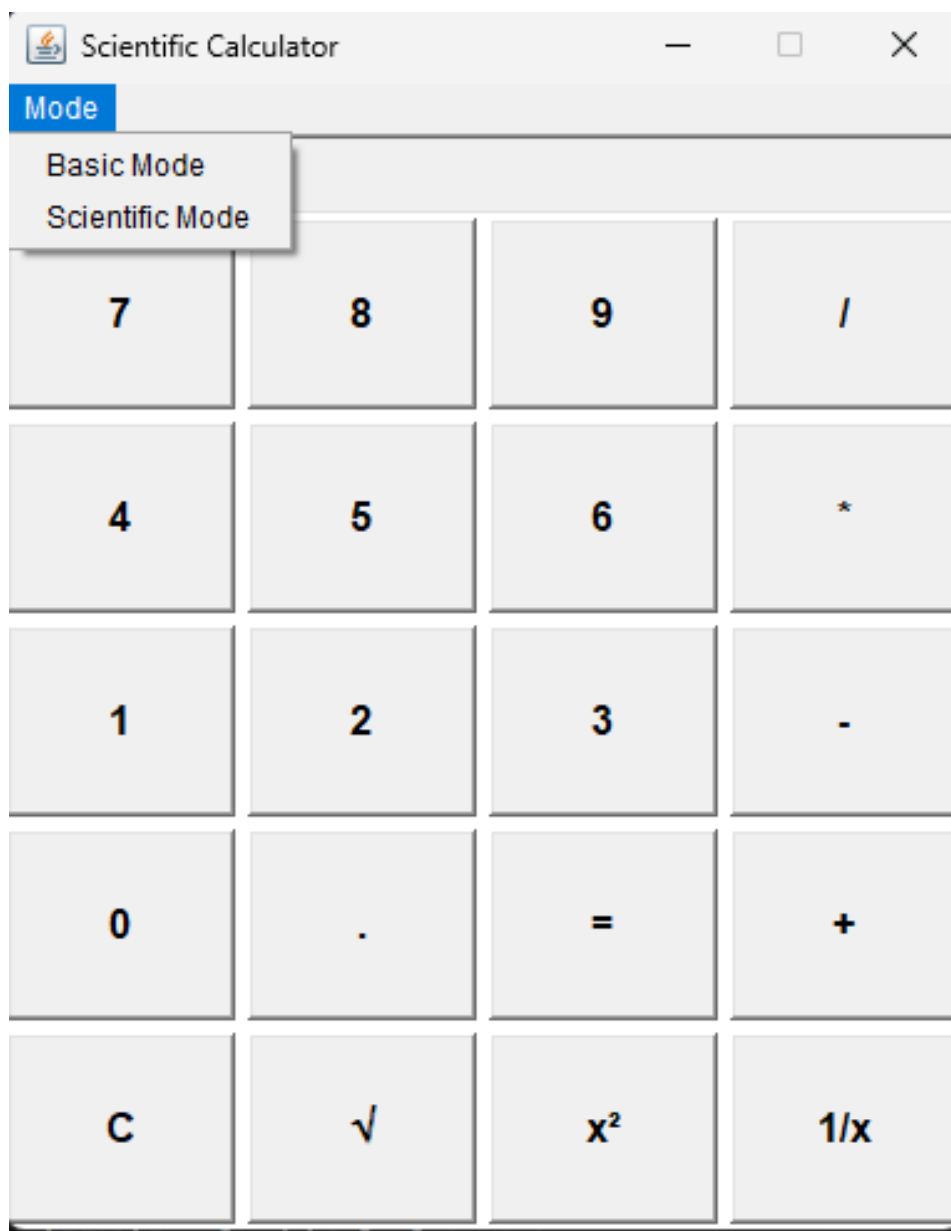
        case "-":
            result = operand1 - operand2;
            break;
        case "*":
            result = operand1 * operand2;
            break;
        case "/":
            if (operand2 == 0) {
                throw new ArithmeticException("Division by zero");
            }
            result = operand1 / operand2;
            break;
        default:
            display.setText("Error");
            return;
    }

    display.setText(String.valueOf(result));
    operator = "";
    operand1 = result;
} catch (Exception ex) {
    display.setText("Error");
}
}

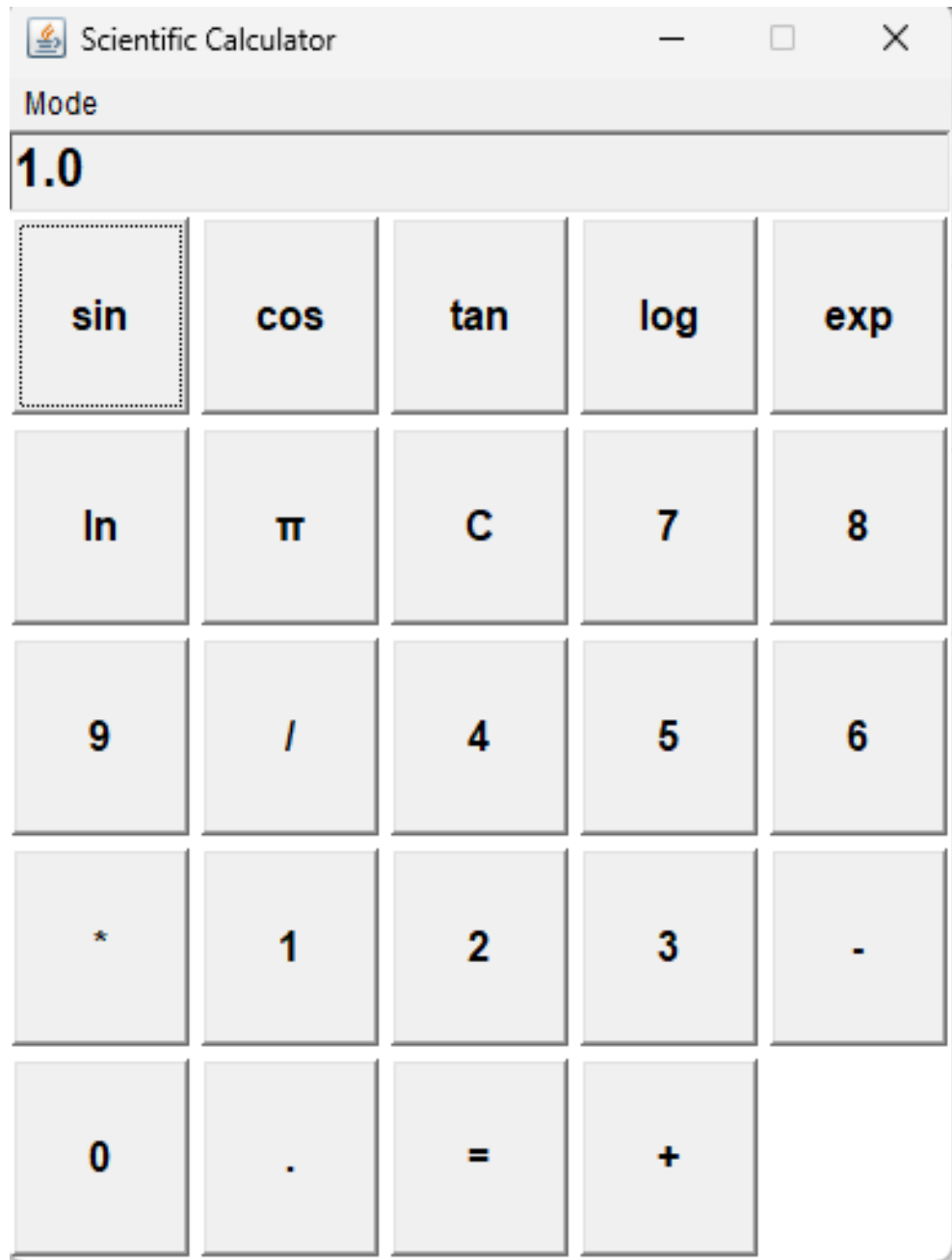
public static void main(String[] args) {
    Calculator calculator = new Calculator();
    calculator.setVisible(true);
}
}

```

APPENDIX B - SCREENSHOTS



Scientific Mode



Basic Mode

