



Langat Singh College

(Run under BRABU Bihar University)

A Mini Project Report on

Calculator

For Java (401 and 405)

Name: **Aditya Srivastava**

Roll: **63**

University Roll: **226060**

Session: **2022-25**

Reg No: **133201119406322/22**

Guided By: **Mr. Ravikant Sir**

Submitted To:

Langat Singh College, Muzaffarpur, Bihar

CONTENTS

<u>S.no.</u>	<u>Content</u>	<u>Page</u>
1.	Acknowledgement	1
2.	Declaration	2
3.	Project Report	3
4.	Hardware and Software Requirements	4
5.	Project Scope	5
6.	Future Enhancement	6
7.	Method	7
8.	User Guide	8
9.	Source Code	9 - 20
10.	Output	21
11.	Testing	22 - 28
12.	Maintenance	29
13.	Bibliography	30

ACKNOWLEDGMENT

I would like to express my sincere gratitude to my Principal Dr. O. P. Roy Sir and my teacher Ravikant Sir and other official of BCA department of Langat Singh College for their invaluable contributions and support throughout the completion of this mini project.

Their guidance, encouragement, and expertise have been instrumental in achieving our objectives.

Finally, I am grateful to my family, friends and colleagues for their patience, understanding, and encouragement during this endeavor.

DECLARATION

I, Aditya Srivastava, hereby declare that the mini project entitled "Calculator" submitted in partial fulfillment of the requirements for BCA at Langat Singh College, Muzaffarpur is entirely my own work, conducted under the guidance of Ravikant Sir.

I affirm that:

1. The contents of this project report are original and have not been submitted, in part or in whole, for any other degree or qualification.
2. Any sources of information used in this project report have been properly acknowledged and referenced.
3. The experiments, analyses, and findings presented in this report are genuine and have not been manipulated or falsified in any manner.
4. Any assistance received from individuals or institutions during the course of this project has been duly acknowledged.

I understand that any act of plagiarism or academic dishonesty in this project would constitute a breach of academic integrity and could lead to disciplinary action.

Aditya Srivastava

Date:

PROJECT REPORT:

JAVA CALCULATOR APPLICATION

Introduction:

The Java Calculator Application is a simple desktop-based calculator developed using Java programming language. The purpose of this application is to perform basic arithmetic calculations such as addition, subtraction, multiplication, and division, along with additional functionalities like calculating square, square root, cube, and cube root of a number. The application provides a graphical user interface (GUI) for ease of use.

Features:

1. Basic Arithmetic Operations: Addition, Subtraction, Multiplication, and Division.
2. Additional Functions: Square, Square Root, Cube, and Cube Root.
3. Error Handling: Proper error handling for invalid input expressions.
4. Graphical User Interface: Utilizes AWT (Abstract Window Toolkit) for GUI components.

HARDWARE AND SOFTWARE REQUIREMENTS

1. Hardware Requirements:

- a. **Processor:** Any modern processor capable of running Java applications, such as Intel Core series or AMD Ryzen series.
- b. **Memory (RAM):** At least 512 MB of RAM. However, having 1 GB or more is recommended for optimal performance.
- c. **Storage:** Minimal storage space required for the application files. A few megabytes should suffice.

2. Software Requirements:

- a. **Operating System:** The Java Calculator Application is platform-independent and can run on any operating system that supports Java, including:
 - i) Windows (Windows 7 or later)
 - ii) MacOS (OS X 10.7 or later)
 - iii) Linux/Unix-based systems
- b. **Java Runtime Environment (JRE):**
 - i) The application requires a Java Runtime Environment (JRE) installed on the system to execute Java programs.
 - ii) **Recommended JRE version:** Java SE 8 or later. However, compatibility may extend to earlier versions depending on the features used in the application.

PROJECT SCOPE

The Java Calculator Application serves as a versatile tool suitable for a wide range of users, including students, professionals, and enthusiasts, who require a reliable solution for performing mathematical computations. Whether it's for simple arithmetic tasks or more complex mathematical analyses, this application offers the functionality and convenience to meet diverse computational needs.

FUTURE ENHANCEMENT

While the current version of the Java Calculator Application provides a solid foundation for basic mathematical operations, future iterations may explore additional features and improvements. Potential enhancements could include:

Scientific Calculator Functionality: Introducing scientific calculator features such as trigonometric functions, logarithms, and exponential calculations to support more advanced mathematical tasks.

User Customization Options: Allowing users to customize the calculator interface, including themes, button layouts, and input preferences, to tailor the application to their individual preferences and requirements.

Memory and History Features: Implementing memory storage capabilities for storing and recalling previous calculations, as well as a history log to track past computations for reference.

By continually refining and expanding the functionality of the Java Calculator Application, we aim to provide users with an indispensable tool for mathematical computation and analysis, empowering them to tackle a wide range of mathematical challenges with ease and efficiency.

METHOD

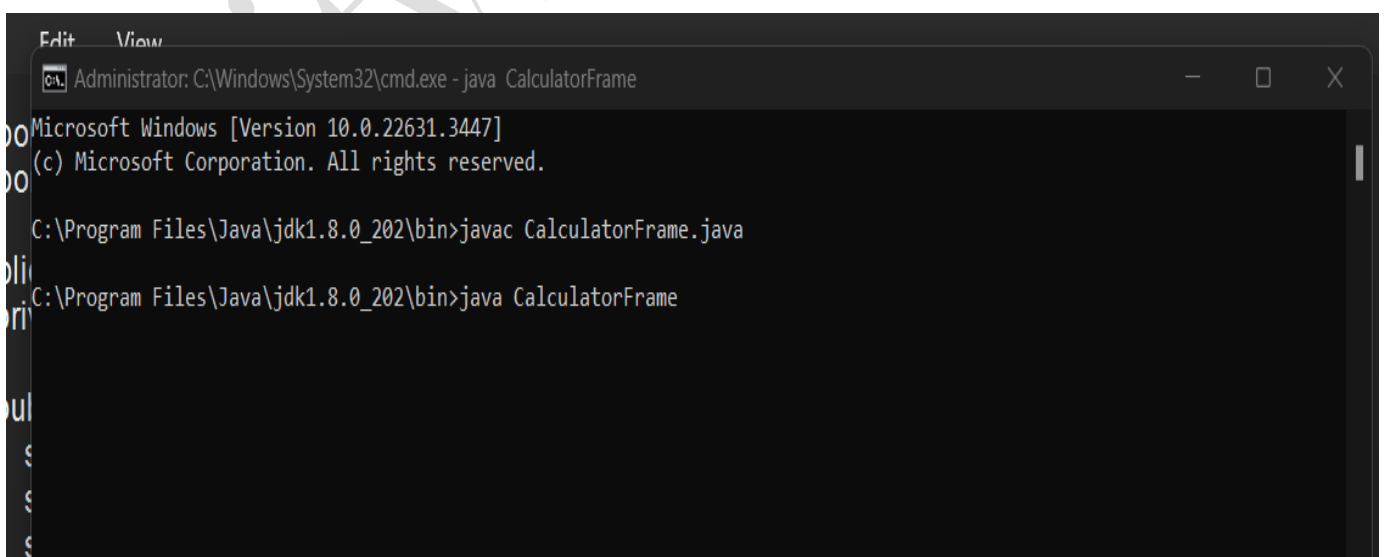
The application is implemented in Java and consists of a single class **CalculatorFrame** which extends the **Frame** class. The graphical user interface is created using AWT components such as **TextField**, **Button**, and **Panel**. The layout is managed using **BorderLayout** and **GridLayout** to arrange the components.

The main functionalities of the calculator are implemented within the **ButtonClickListener** class, which handles button click events. Each button corresponds to a specific action such as arithmetic operations, clearing the display, backspacing, and additional mathematical functions.

The mathematical expressions are evaluated using a custom **evaluate** method, which parses the input expression and calculates the result based on the order of operations (BODMAS/BIDMAS).

USER GUIDE

1. Launch the application by running the `CalculatorFrame` class.
2. The calculator interface will appear with a display area at the top and buttons for various operations below.
3. Enter numerical values and perform operations using the buttons.
4. Additional functions such as square, square root, cube, and cube root can be accessed by clicking the corresponding buttons.
5. To clear the display, press the "C" button.
6. To delete the last entered character, press the "<-" button.
7. Closing the application window will exit the calculator.



```
Administrator: C:\Windows\System32\cmd.exe - java CalculatorFrame
Microsoft Windows [Version 10.0.22631.3447]
(c) Microsoft Corporation. All rights reserved.

C:\Program Files\Java\jdk1.8.0_202\bin>javac CalculatorFrame.java
C:\Program Files\Java\jdk1.8.0_202\bin>java CalculatorFrame
```

SOURCE CODE

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

public class CalculatorFrame extends Frame {
    private TextField display;

    public CalculatorFrame() {
        setTitle("Calculator");
        setSize(500, 700);
        setBackground(new Color(0, 20, 40)); // Set background
color to blackish blue
        setResizable(false);

        setLayout(new BorderLayout());

        display = new TextField();
        display.setEditable(false);
        display.setFont(new Font("Arial", Font.PLAIN, 20)); //
Increase font size
```

```
display.setPreferredSize(new Dimension(10, 50)); //
Increase TextField size
```

```
display.setBackground(new Color(0, 20, 40)); // Set
background color to blackish blue
```

```
display.setForeground(Color.WHITE); // Set text color
to white
```

```
add(display, BorderLayout.NORTH);
```

```
Panel buttonPanel = new Panel(new GridLayout(6, 4));
```

```
buttonPanel.setBackground(new Color(0, 20, 40)); // Set
background color to blackish blue
```

```
String[] buttonLabels = {
```

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

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

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

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

```
    "(", ")", "C", "<-", // Added backspace button
```

```
    "x^2", "sqrt", "x^3", "cbrt" // Added square, square root,
cube, and cube root buttons
```

```
};
```

```
for (String label : buttonLabels) {  
    Button button = new Button(label);  
    button.addActionListener(new ButtonClickListener());  
    button.setFont(new Font("Arial", Font.PLAIN, 18)); //  
Increase button font size  
    button.setBackground(new Color(0, 20, 40)); // Set  
background color to blackish blue  
    button.setForeground(Color.WHITE); // Set text  
color to white  
    buttonPanel.add(button);  
}  
  
add(buttonPanel, BorderLayout.CENTER);  
  
addWindowListener(new WindowAdapter() {  
    public void windowClosing(WindowEvent  
windowEvent) {  
        System.exit(0);  
    }  
});  
}
```

```
private class ButtonClickListener implements
ActionListener {

    public void actionPerformed(ActionEvent e) {
        String command = e.getActionCommand();
        if (command.equals("=")) {
            calculate();
        } else if (command.equals("C")) {
            clearAll();
        } else if (command.equals("<-")) {
            backspace();
        } else if (command.equals("x^2")) {
            square();
        } else if (command.equals("sqrt")) {
            squareRoot();
        } else if (command.equals("x^3")) {
            cube();
        } else if (command.equals("cbrt")) {
            cubeRoot();
        } else if (command.equals("(") || command.equals(")"))
{
            display.setText(display.getText() + command);
        } else {
```

```
        display.setText(display.getText() + command);
    }
}
```

```
private void calculate() {
    String expression = display.getText();
    try {
        double result = evaluate(expression);
        display.setText(Double.toString(result));
    } catch (Exception ex) {
        display.setText("Error");
    }
}
```

```
private void clearAll() {
    display.setText("");
}
```

```
private void backspace() {
    String text = display.getText();
    if (!text.isEmpty()) {
```

```
        display.setText(text.substring(0, text.length() - 1));  
    }  
}
```

```
private void square() {  
    String text = display.getText();  
    if (!text.isEmpty()) {  
        double value = Double.parseDouble(text);  
        display.setText(Double.toString(value * value));  
    }  
}
```

```
private void squareRoot() {  
    String text = display.getText();  
    if (!text.isEmpty()) {  
        double value = Double.parseDouble(text);  
        display.setText(Double.toString(Math.sqrt(value)));  
    }  
}
```

```
private void cube() {
```



```
String text = display.getText();
if (!text.isEmpty()) {
    double value = Double.parseDouble(text);
    display.setText(Double.toString(value * value *
value));
}
}

private void cubeRoot() {
    String text = display.getText();
    if (!text.isEmpty()) {
        double value = Double.parseDouble(text);
        display.setText(Double.toString(Math.cbrt(value)));
    }
}

private double evaluate(String expression) {
    return new Object() {
        int pos = -1, ch;

        void nextChar() {
```

```
        ch = (++pos < expression.length()) ?  
expression.charAt(pos) : -1;  
    }
```

```
    boolean eat(int charToEat) {  
        while (ch == ' ') nextChar();  
        if (ch == charToEat) {  
            nextChar();  
            return true;  
        }  
        return false;  
    }
```

```
    double parse() {  
        nextChar();  
        double x = parseExpression();  
        if (pos < expression.length()) throw new  
RuntimeException("Unexpected: " + (char) ch);  
        return x;  
    }
```

```
    double parseExpression() {
```

```
double x = parseTerm();  
for (;;) {  
    if (eat('+')) x += parseTerm();  
    else if (eat('-')) x -= parseTerm();  
    else return x;  
}  
}
```

```
double parseTerm() {  
    double x = parseFactor();  
    for (;;) {  
        if (eat('*')) x *= parseFactor();  
        else if (eat('/')) x /= parseFactor();  
        else return x;  
    }  
}
```

```
double parseFactor() {  
    if (eat('+')) return parseFactor();  
    if (eat('-')) return -parseFactor();
```

```
double x;
int startPos = this.pos;
if (eat('(')) {
    x = parseExpression();
    eat(')');
} else if ((ch >= '0' && ch <= '9') || ch == '.') {
    while ((ch >= '0' && ch <= '9') || ch == '.')
nextChar();

    x =
Double.parseDouble(expression.substring(startPos, this.pos));
} else {
    throw new RuntimeException("Unexpected: "
+ (char) ch);
}

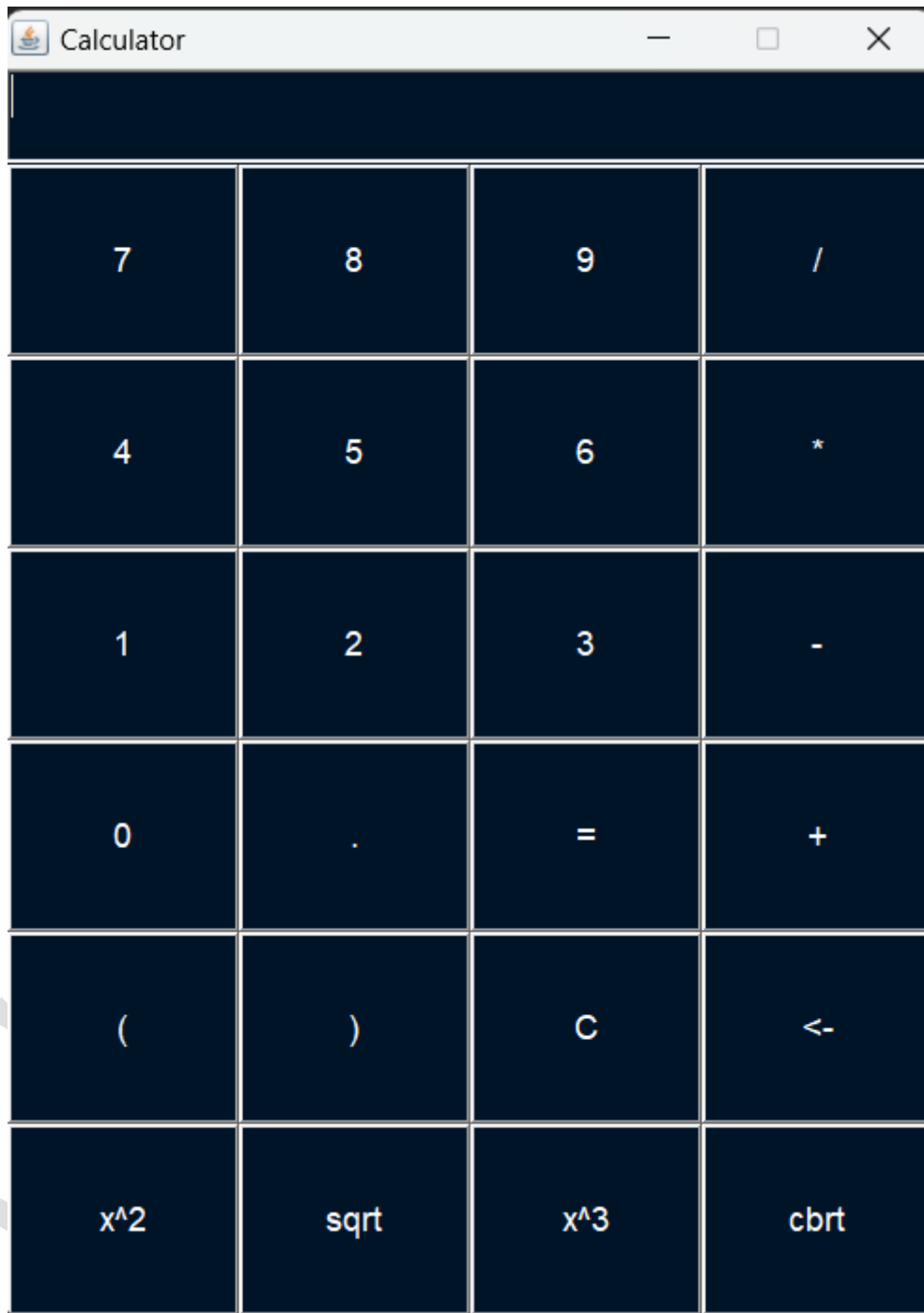
if (eat('^')) x = Math.pow(x, parseFactor());

return x;
}

}.parse();
}
}
```

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

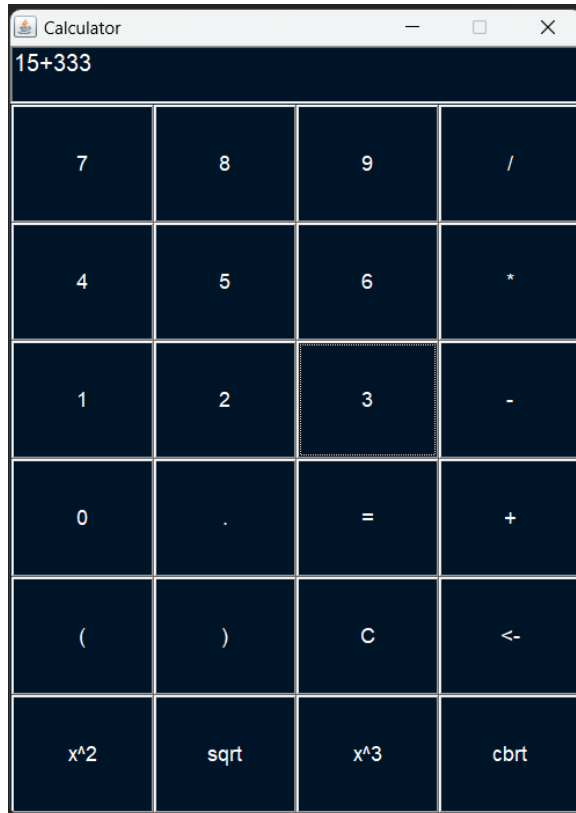
OUTPUT



TESTING

1. Unit Testing:

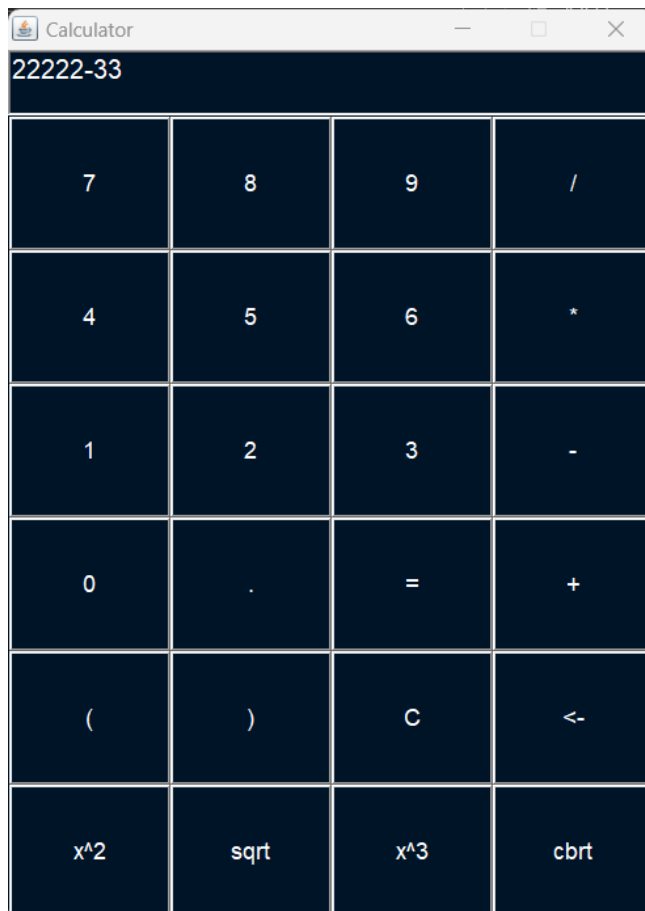
a. Addition:



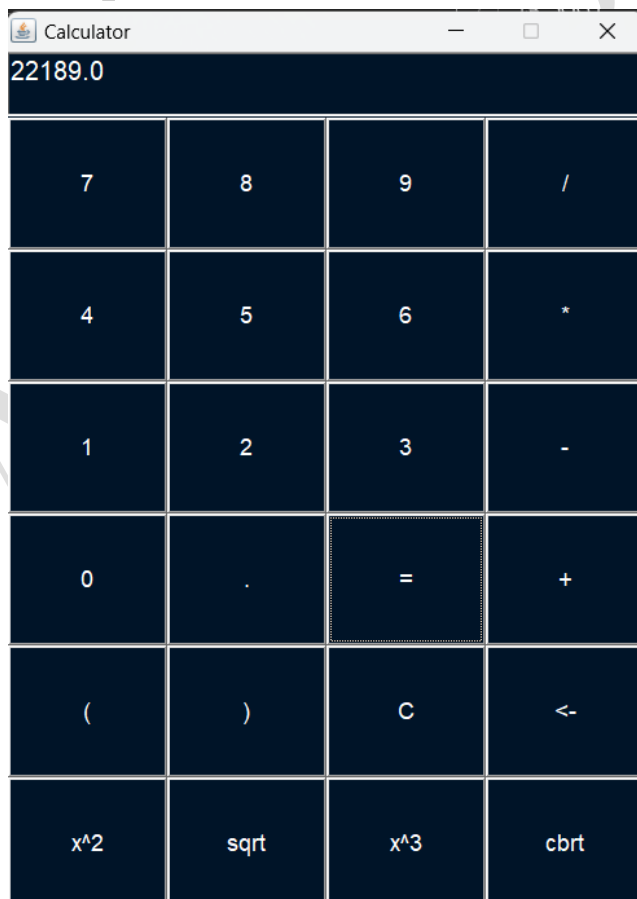
Output:



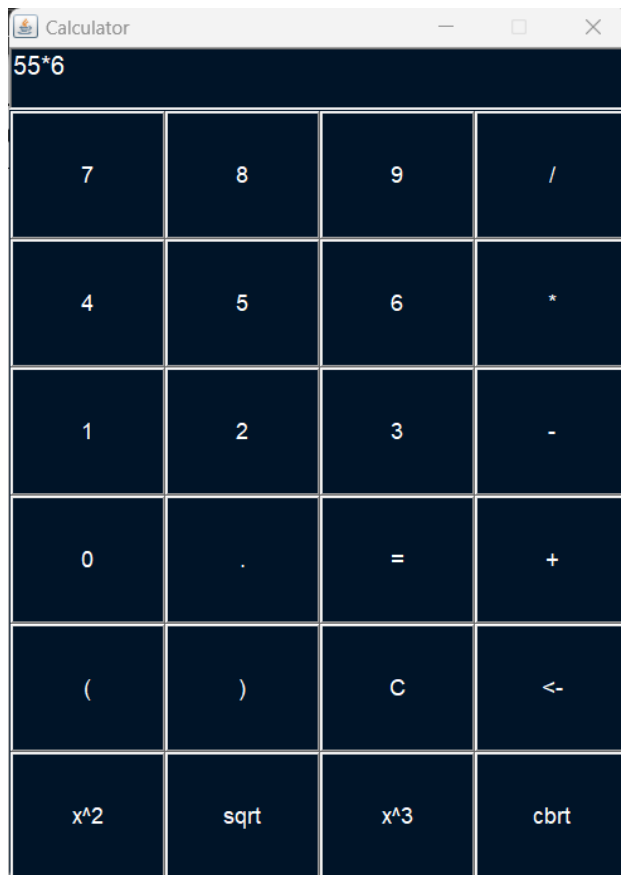
b. Substruction:



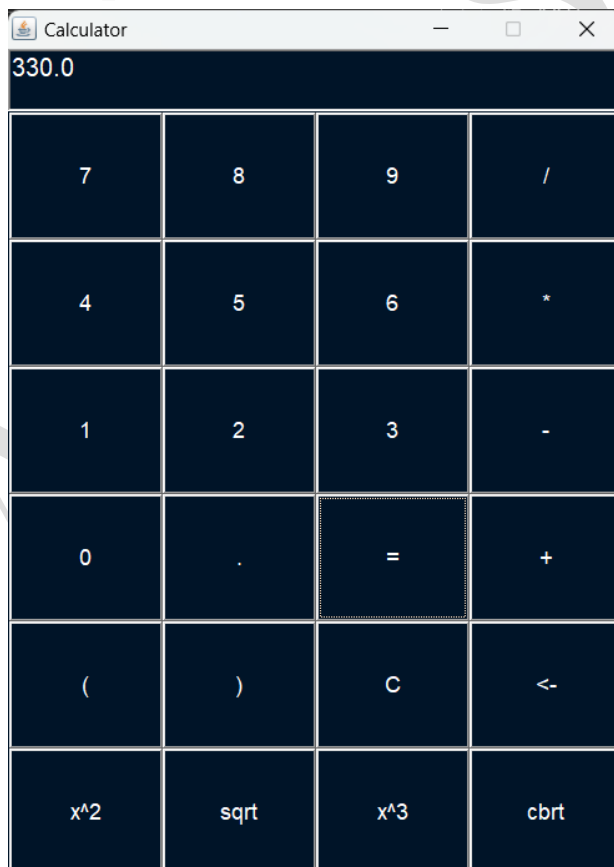
Output:



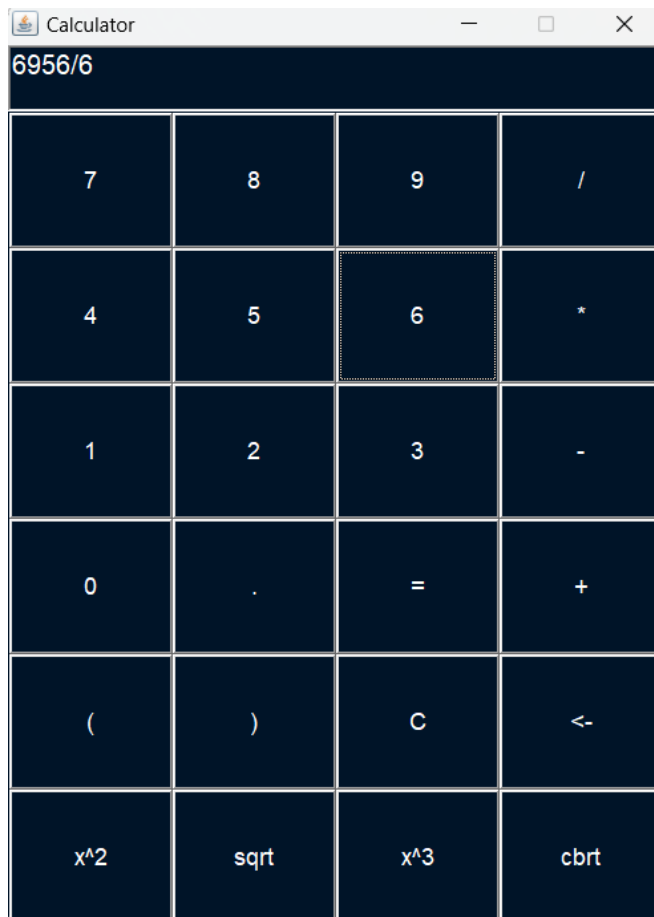
c. Multiplication:



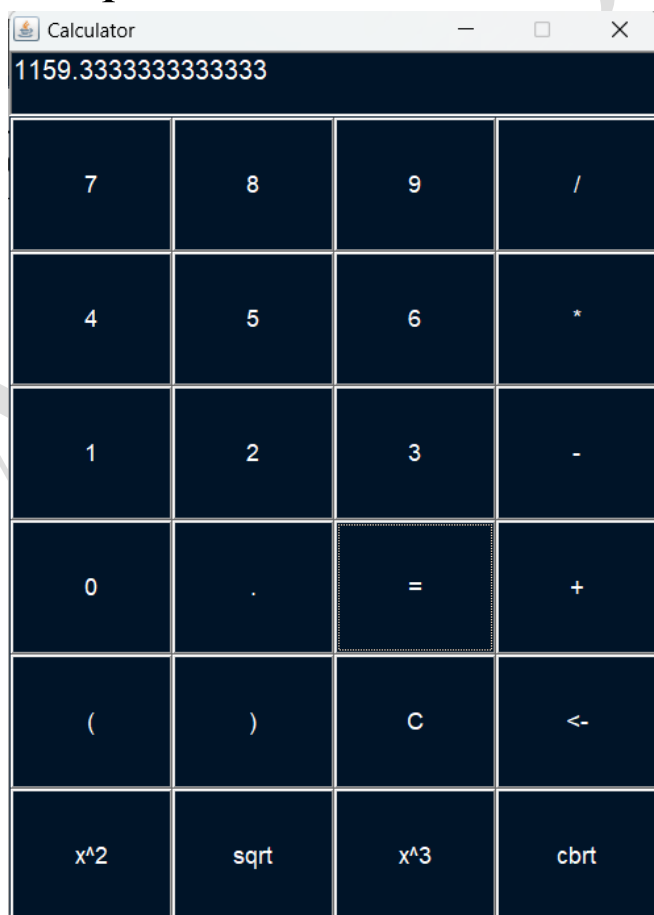
Output:



d. Division:

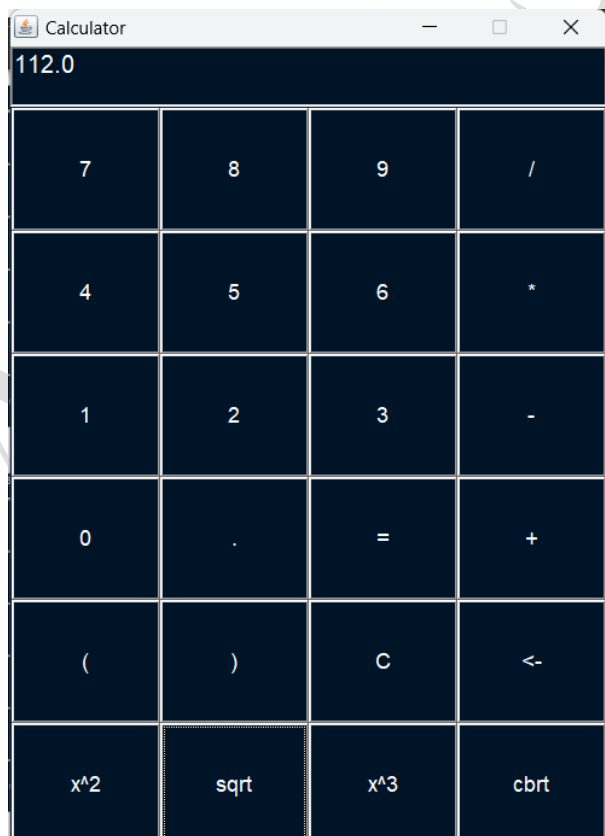
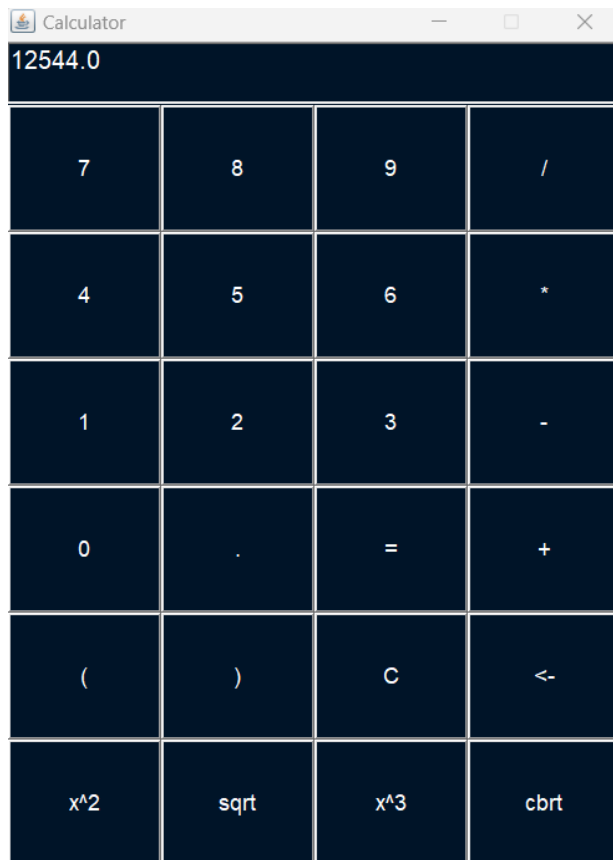


Output:



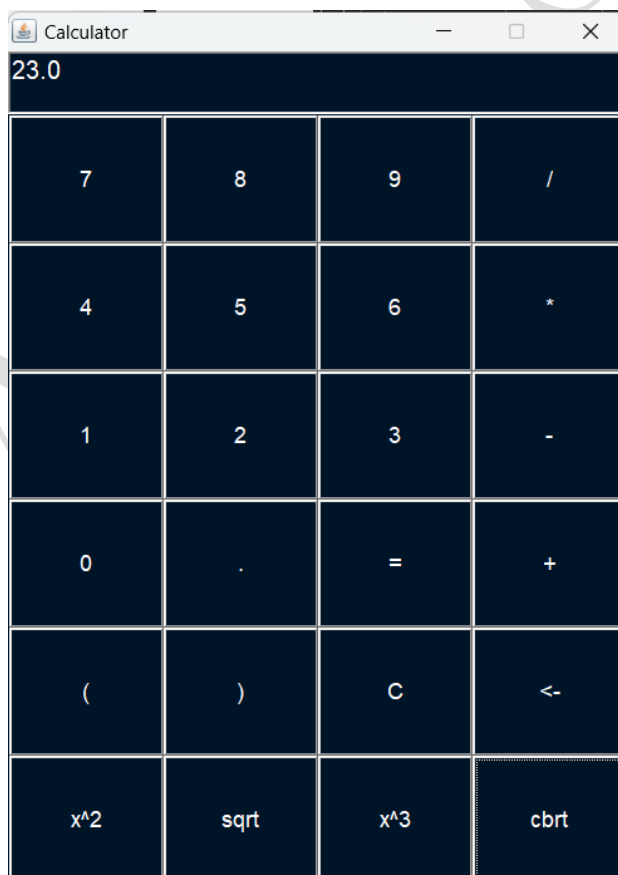
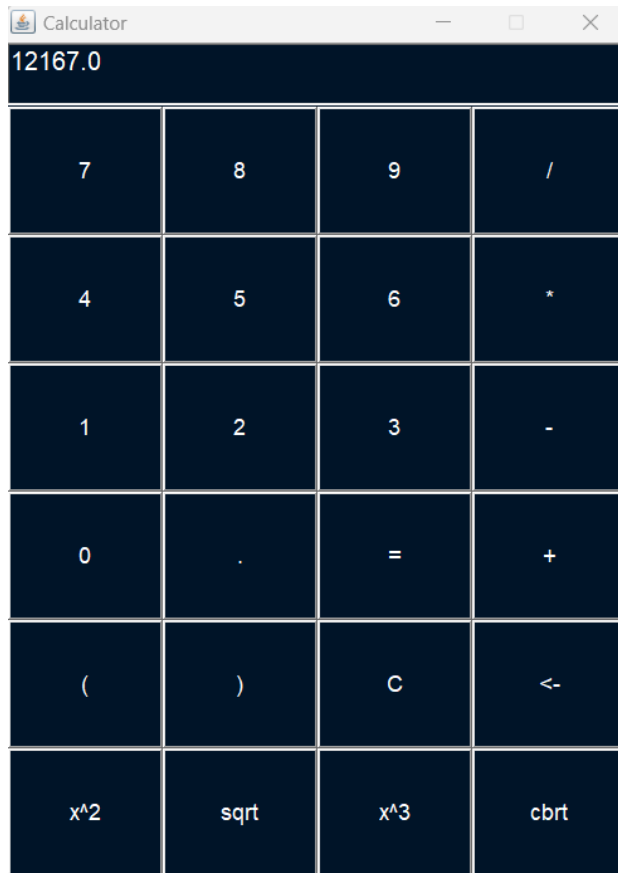
e. Square and Square Root:

For example: Number is 112

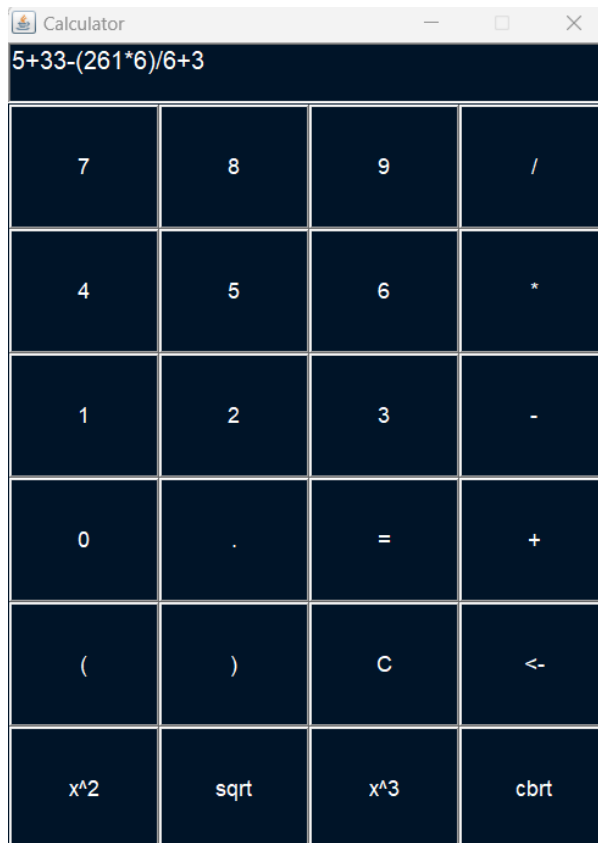


f. Cube and Cube Root:

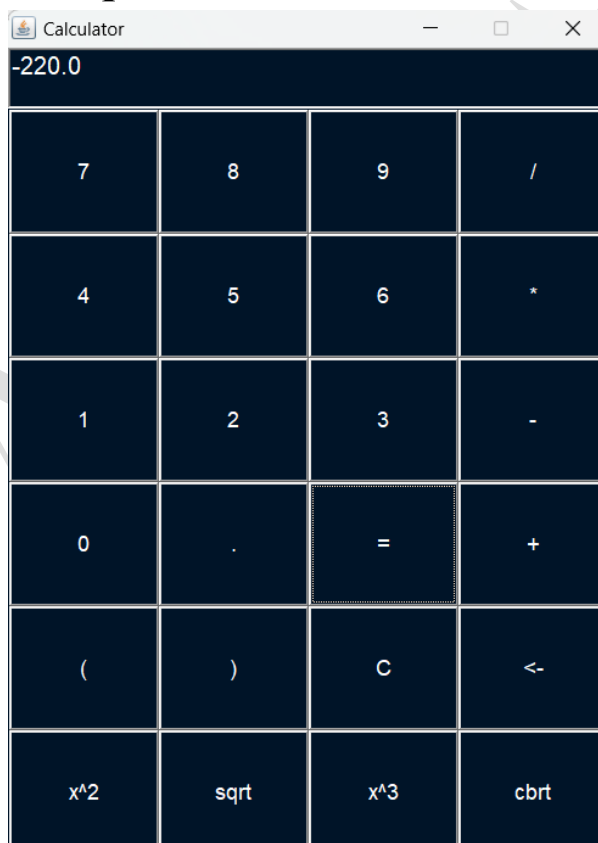
For example: Number is 23



2. System Test:



Output:



MAINTENANCE

1. Bug Fixes and Issue Resolution:

- Addressed reported bugs and issues, prioritizing those affecting core functionalities and user experience.

2. Performance Optimization:

- Conducted performance monitoring and analysis to identify performance bottlenecks and areas for optimization.

3. Compatibility Updates:

- Stayed up-to-date with Java platform updates and compatibility requirements.

4. Security Enhancements:

- Reviewed and updated security measures to address potential vulnerabilities and security threats.

5. Feature Updates and Enhancements:

- Solicited user feedback and feature requests to identify opportunities for improving existing functionality and adding new features.

BIBLIOGRAPHY

1. <https://www.javatpoint.com/calculator-in-java>
2. <https://github.com/topics/java-calculator>
3. <https://chatgpt.com/c/465700bd>