Group Work

Edulink international college,nairobi.

Console based calculator

Table of Contents

[CERTIFICATE OF COMPLETION 2](#_Toc173319597)

[SYNOPSIS 2](#_Toc173319598)

[ANALYSIS 3](#_Toc173319599)

[Functionality: 3](#_Toc173319600)

[User Interface: 3](#_Toc173319601)

[Design Considerations: 3](#_Toc173319602)

[Use Cases: 3](#_Toc173319603)

[Limitations: 3](#_Toc173319604)

[DESIGN. 4](#_Toc173319605)

[SCREEN SHOTS 5](#_Toc173319606)

[SOURCE CODE 6](#_Toc173319607)

[USER/DEVELOPERS GUIDE 13](#_Toc173319608)

[Instructional Manual 13](#_Toc173319609)

## CERTIFICATE OF COMPLETION

This is to certify that:

Student1485831 BONFACE ONCHIEKU KEROSI

Student1488107 VINCENT OMUSUNGU OMUSEE

Student1485985 NELLY WANJIRU KNIGHT

Student1485874 MARK GITHAE NGURARI

has successfully designed, developed, and implemented a Console-Based Calculator as part of our programming coursework.

The project includes the following features:

* Adding, subtracting, Multiplication, Division.
* Power, Square, Cube, Square root.
* Round, ceiling, floor, Min Value, Max Value.
* Trigonometric functions such as sin, cos, asin, acos, atan.
* Exponential function.
* Palindrome.
* Armstrong number.
* Prime number.
* Average.
* GCD of the two number Enter ed by the user.
* LCM of the two number Enter ed by the user.
* Exit option (to terminate the program).

This certificate is awarded in recognition of the dedication, effort, and technical skills demonstrated in completing the project.

Project Supervisor: Mr. Kenneth Mathiu

Completion Date: 17 July 2024

Institution: Edulink International College

## SYNOPSIS

A console-based calculator is a simple, text-based application that allows users to perform basic arithmetic operations such as addition, subtraction, multiplication, and division. The user interacts with the calculator through a command-line interface (CLI), entering mathematical expressions, and receiving the computed results as output.

The calculator typically includes the following features:

1. **Basic Operations**: Supports addition (+), subtraction (-), multiplication (\*), and division (/).
2. **Input Handling**: Accepts user input in the form of mathematical expressions.
3. **Output Display**: Displays the result of the computation.
4. **Error Handling**: Manages common errors such as division by zero and invalid input formats.

## ANALYSIS

### Functionality:

* The primary functionality of a console-based calculator is to interpret and evaluate mathematical expressions provided by the user. This involves parsing the input, performing the necessary computations, and returning the result.
* Advanced versions may include support for additional operations such as exponentiation, square roots, and trigonometric functions, as well as the ability to handle parentheses for operation precedence.

### User Interface:

* The user interface is minimalistic, relying solely on text input and output. This simplicity makes the application lightweight and easy to use on any system with a command-line interface.
* The calculator should provide clear instructions for use and error messages.

### Design Considerations:

* **Input Parsing**: The calculator must accurately parse user input to identify numbers and operators. This may involve tokenizing the input string and applying the appropriate arithmetic operations.
* **Error Handling**: Robust error handling is essential to manage invalid input, such as non-numeric characters or invalid operations (e.g., division by zero). The application should provide informative error messages to guide the user.
* **Modularity**: The design should be modular, separating the input handling, computation, and output display into distinct components. This promotes maintainability and allows for easy extension of functionality.
* **Efficiency**: Given the typically simple nature of the calculations, the application should be efficient and responsive, providing instant results for user inputs.

### Use Cases:

1. **Educational Tool**: It can serve as an educational tool for learning basic arithmetic and understanding how mathematical expressions are evaluated.
2. **Quick Calculations**: It provides a quick and convenient way to perform simple calculations without the need for a graphical user interface (GUI).
3. **Development Exercise**: Implementing a console-based calculator is a common exercise for learning programming concepts such as input/output handling, control flow, and basic algorithms.

### Limitations:

* The lack of a graphical interface may limit its appeal for users accustomed to more visual tools.
* Advanced mathematical functions and complex input expressions may not be supported in the most basic versions.
* Limited to environments where a command-line interface is available and accessible.

Conclusion: A console-based calculator is a fundamental tool that demonstrates core programming concepts while providing practical utility for basic arithmetic calculations. Its simplicity and ease of use make it a suitable project for beginner programmers and a useful utility for quick mathematical tasks. With thoughtful design and implementation, it can serve as a robust and extensible application, offering opportunities for further development and enhancement.

## DESIGN.

## SCREEN SHOTS

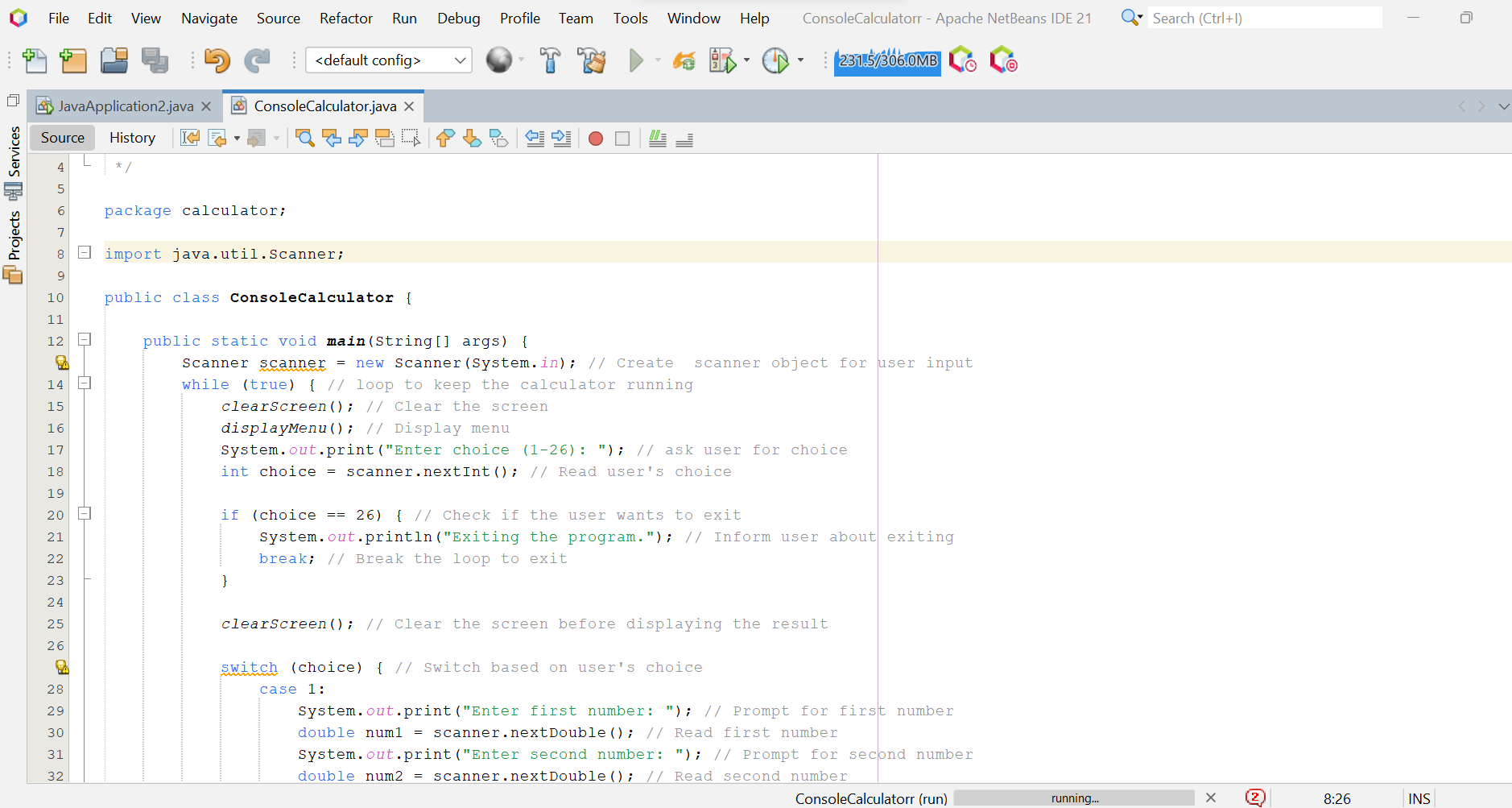


Figure 1: code snippet

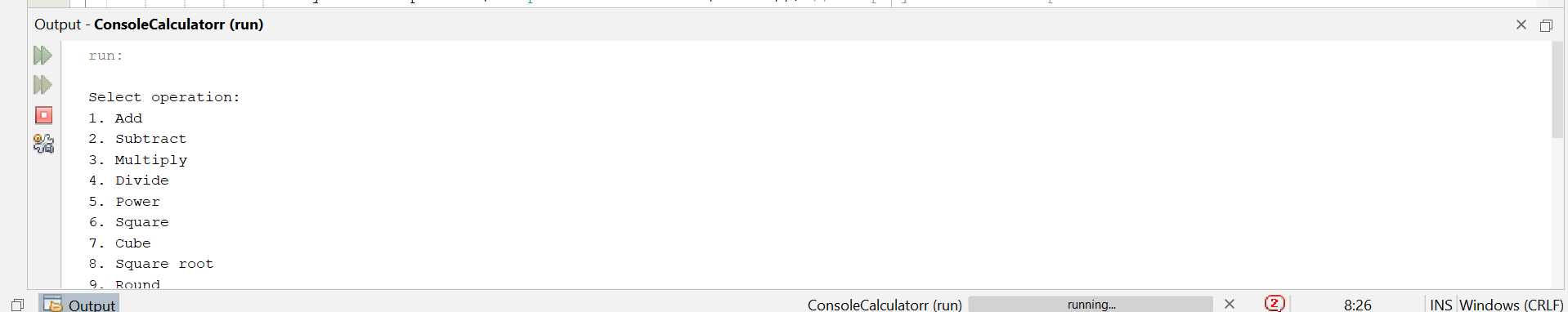


Figure 2: operation window



Figure 3: Output window

## SOURCE CODE

Below is the code snippet of our console calculator program and the comments for each piece of code to help other programmers navigate and understand our code.

import java.util.Scanner;

public class ConsoleCalculator {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in); // Create scanner object for user input

while (true) { // loop to keep the calculator running

clearScreen(); // Clear the screen

displayMenu(); // Display menu

System.out.print("Enter choice (1-26): "); // ask user for choice

int choice = scanner.nextInt(); // Read user's choice

if (choice == 26) { // Check if the user wants to exit

System.out.println("Exiting the program."); // Inform user about exiting

break; // Break the loop to exit

}

clearScreen(); // Clear the screen before displaying the result

switch (choice) { // Switch based on user's choice

case 1:

System.out.print("Enter first number: "); // Prompt for first number

double num1 = scanner.nextDouble(); // Read first number

System.out.print("Enter second number: "); // Prompt for second number

double num2 = scanner.nextDouble(); // Read second number

System.out.println("Result: " + add(num1, num2)); // Display result of addition

break;

case 2:

System.out.print("Enter first number: "); // Prompt for first number

num1 = scanner.nextDouble(); // Read first number

System.out.print("Enter second number: "); // Prompt for second number

num2 = scanner.nextDouble(); // Read second number

System.out.println("Result: " + subtract(num1, num2)); // Display result of subtraction

break;

case 3:

System.out.print("Enter first number: "); // Prompt for first number

num1 = scanner.nextDouble(); // Read first number

System.out.print("Enter second number: "); // Prompt for second number

num2 = scanner.nextDouble(); // Read second number

System.out.println("Result: " + multiply(num1, num2)); // Display result of multiplication

break;

case 4:

System.out.print("Enter first number: "); // Prompt for first number

num1 = scanner.nextDouble(); // Read first number

System.out.print("Enter second number: "); // Prompt for second number

num2 = scanner.nextDouble(); // Read second number

if (num2 != 0) { // Check if second number is not zero

System.out.println("Result: " + divide(num1, num2)); // Display result of division

} else {

System.out.println("Error! Division by zero."); // Display error message

}

break;

case 5:

System.out.print("Enter base: "); // Prompt for base

num1 = scanner.nextDouble(); // Read base

System.out.print("Enter exponent: "); // Prompt for exponent

num2 = scanner.nextDouble(); // Read exponent

System.out.println("Result: " + Math.pow(num1, num2)); // Display result of power

break;

case 6:

System.out.print("Enter number: "); // Prompt for number

num1 = scanner.nextDouble(); // Read number

System.out.println("Result: " + square(num1)); // Display result of square

break;

case 7:

System.out.print("Enter number: "); // Prompt for number

num1 = scanner.nextDouble(); // Read number

System.out.println("Result: " + cube(num1)); // Display result of cube

break;

case 8:

System.out.print("Enter number: "); // Prompt for number

num1 = scanner.nextDouble(); // Read number

System.out.println("Result: " + Math.sqrt(num1)); // Display result of square root

break;

case 9:

System.out.print("Enter number: "); // Prompt for number

num1 = scanner.nextDouble(); // Read number

System.out.println("Result: " + Math.round(num1)); // Display result of round

break;

case 10:

System.out.print("Enter number: "); // Prompt for number

num1 = scanner.nextDouble(); // Read number

System.out.println("Result: " + Math.ceil(num1)); // Display result of ceiling

break;

case 11:

System.out.print("Enter number: "); // Prompt for number

num1 = scanner.nextDouble(); // Read number

System.out.println("Result: " + Math.floor(num1)); // Display result of floor

break;

case 12:

System.out.print("Enter first number: "); // Prompt for first number

num1 = scanner.nextDouble(); // Read first number

System.out.print("Enter second number: "); // Prompt for second number

num2 = scanner.nextDouble(); // Read second number

System.out.println("Result: " + Math.min(num1, num2)); // Display minimum value

break;

case 13:

System.out.print("Enter first number: "); // Prompt for first number

num1 = scanner.nextDouble(); // Read first number

System.out.print("Enter second number: "); // Prompt for second number

num2 = scanner.nextDouble(); // Read second number

System.out.println("Result: " + Math.max(num1, num2)); // Display maximum value

break;

case 14:

System.out.print("Enter angle in degrees: "); // Prompt for angle in degrees

num1 = scanner.nextDouble(); // Read angle

System.out.println("Result: " + Math.sin(Math.toRadians(num1))); // Display sine value

break;

case 15:

System.out.print("Enter angle in degrees: "); // Prompt for angle in degrees

num1 = scanner.nextDouble(); // Read angle

System.out.println("Result: " + Math.cos(Math.toRadians(num1))); // Display cosine value

break;

case 16:

System.out.print("Enter value (-1 to 1): "); // Prompt for value

num1 = scanner.nextDouble(); // Read value

System.out.println("Result: " + Math.asin(num1)); // Display arc sine value

break;

case 17:

System.out.print("Enter value (-1 to 1): "); // Prompt for value

num1 = scanner.nextDouble(); // Read value

System.out.println("Result: " + Math.acos(num1)); // Display arc cosine value

break;

case 18:

System.out.print("Enter value: "); // Prompt for value

num1 = scanner.nextDouble(); // Read value

System.out.println("Result: " + Math.atan(num1)); // Display arc tangent value

break;

case 19:

System.out.print("Enter number: "); // Prompt for number

num1 = scanner.nextDouble(); // Read number

System.out.println("Result: " + Math.exp(num1)); // Display exponential value

break;

case 20:

System.out.print("Enter a number: "); // Prompt for number

int intNum = scanner.nextInt(); // Read number

System.out.println("Is palindrome: " + isPalindrome(intNum)); // Display if number is palindrome

break;

case 21:

System.out.print("Enter a number: "); // Prompt for number

intNum = scanner.nextInt(); // Read number

System.out.println("Is Armstrong: " + isArmstrong(intNum)); // Display if number is Armstrong number

break;

case 22:

System.out.print("Enter a number: "); // Prompt for number

intNum = scanner.nextInt(); // Read number

System.out.println("Is prime: " + isPrime(intNum)); // Display if number is prime

break;

case 23:

System.out.print("Enter the number of entries: "); // Prompt for number of entries

int count = scanner.nextInt(); // Read number of entries

double[] numbers = new double[count]; // Create array to store numbers

for (int i = 0; i < count; i++) { // Loop to read each number

System.out.print("Enter number " + (i + 1) + ": "); // Prompt for each number

numbers[i] = scanner.nextDouble(); // Read each number

}

System.out.println("Average: " + calculateAverage(numbers)); // Display average

break;

case 24:

System.out.print("Enter first number: "); // Prompt for first number

num1 = scanner.nextDouble(); // Read first number

System.out.print("Enter second number: "); // Prompt for second number

num2 = scanner.nextDouble(); // Read second number

System.out.println("GCD: " + gcd(num1, num2)); // Display GCD

break;

case 25:

System.out.print("Enter first number: "); // Prompt for first number

num1 = scanner.nextDouble(); // Read first number

System.out.print("Enter second number: "); // Prompt for second number

num2 = scanner.nextDouble(); // Read second number

System.out.println("LCM: " + lcm(num1, num2)); // Display LCM

break;

default:

System.out.println("Invalid choice! Please try again."); // Display error for invalid choice

break;

}

System.out.print("\nPress Enter to continue..."); // Prompt to press Enter to continue

scanner.nextLine(); // Consume newline left-over

scanner.nextLine(); // Wait for user to press Enter

}

scanner.close(); // Close the scanner

}

public static void displayMenu() {

System.out.println("\nSelect operation:"); // Display menu options

System.out.println("1. Add");

System.out.println("2. Subtract");

System.out.println("3. Multiply");

System.out.println("4. Divide");

System.out.println("5. Power");

System.out.println("6. Square");

System.out.println("7. Cube");

System.out.println("8. Square root");

System.out.println("9. Round");

System.out.println("10. Ceiling");

System.out.println("11. Floor");

System.out.println("12. Min Value");

System.out.println("13. Max Value");

System.out.println("14. Sine");

System.out.println("15. Cosine");

System.out.println("16. Arc Sine");

System.out.println("17. Arc Cosine");

System.out.println("18. Arc Tangent");

System.out.println("19. Exponential");

System.out.println("20. Check Palindrome");

System.out.println("21. Check Armstrong Number");

System.out.println("22. Check Prime Number");

System.out.println("23. Calculate Average");

System.out.println("24. GCD");

System.out.println("25. LCM");

System.out.println("26. Exit");

}

public static void clearScreen() {

System.out.print("\033[H\033[2J"); // ANSI escape codes to clear the console

System.out.flush(); // Flush the output stream

}

public static double add(double a, double b) {

return a + b; // Return sum of two numbers

}

public static double subtract(double a, double b) {

return a - b; // Return difference of two numbers

}

public static double multiply(double a, double b) {

return a \* b; // Return product of two numbers

}

public static double divide(double a, double b) {

return a / b; // Return quotient of two numbers

}

public static double square(double a) {

return a \* a; // Return square of a number

}

public static double cube(double a) {

return a \* a \* a; // Return cube of a number

}

public static boolean isPalindrome(int number) {

int reversed = 0, original = number, remainder; // Initialize variables

while (number != 0) { // Loop to reverse the number

remainder = number % 10; // Get last digit

reversed = reversed \* 10 + remainder; // Append digit to reversed number

number /= 10; // Remove last digit from number

}

return original == reversed; // Check if original number is equal to reversed number

}

public static boolean isArmstrong(int number) {

int original = number, remainder, result = 0, n = 0; // Initialize variables

while (original != 0) { // Loop to count digits

original /= 10; // Remove last digit from original number

++n; // Increment digit count

}

original = number; // Reset original to number

while (original != 0) { // Loop to calculate sum of powers of digits

remainder = original % 10; // Get last digit

result += Math.pow(remainder, n); // Add power of digit to result

original /= 10; // Remove last digit from original number

}

return result == number; // Check if result is equal to original number

}

public static boolean isPrime(int number) {

if (number <= 1) { // Check if number is less than or equal to 1

return false; // Return false for non-prime numbers

}

for (int i = 2; i <= Math.sqrt(number); i++) { // Loop to check divisibility

if (number % i == 0) { // Check if number is divisible by i

return false; // Return false if number is divisible by i

}

}

return true; // Return true for prime numbers

}

public static double calculateAverage(double[] numbers) {

double sum = 0; // Initialize sum

for (double number : numbers) { // Loop through numbers

sum += number; // Add each number to sum

}

return sum / numbers.length; // Return average

}

public static double gcd(double a, double b) {

while (b != 0) { // Loop until b is zero

double temp = b; // Store b in temp

b = a % b; // Set b to remainder of a divided by b

a = temp; // Set a to temp

}

return a; // Return GCD

}

public static double lcm(double a, double b) {

return (a \* b) / gcd(a, b); // Return LCM

}

}

## USER GUIDE

### Instructional Manual

This manual will guide you on how to use the calculator to perform various arithmetic and mathematical operations. Follow the steps below to use the calculator effectively.

#### 1. Introduction

The Console-Based Calculator allows you to perform various operations including addition, subtraction, multiplication, division, power calculations, trigonometric functions, number theory operations, and more on your compiler. The calculator runs in a loop, allowing you to perform an operation then afterwards displays the results and prompts you to click enter to resume the loop which then gives you the menu of the supported arithmetic operations.

#### **2.** Getting Started

##### Requirements:

Java Development Kit (JDK) installed on your computer.

Basic knowledge of navigating the terminal or command prompt.

##### Steps to Run the Calculator

Save the provided source code in a file named ConsoleCalculator.java.

Open Terminal/Command Prompt

Navigate to the directory where you saved the ConsoleCalculator.java file.

Compile the Program: javac ConsoleCalculator.java

Run the Program: java ConsoleCalculator

Use the Calculator

Follow the on-screen prompts to select operations and Enter numbers.

#### 3. List of Supported Operations

The program supports the following arithmetic operations:

Addition, Subtraction, Multiplication, Division, Power, Square, Cube, Square Root, Round, Ceiling, Floor, Min Value, Max Value, Sin, Cos, Asin, Acos, Atan, Exponential, Palindrome, Armstrong, Number, Prime Number, Average, GCD, LCM, Exit.

#### 4. Detailed Instructions for Each Operation

##### 4.1 Basic Arithmetic Operations

###### Addition (Option 1)

Use case: Adds two numbers.

User inputs two numbers.

Example: Enter 1 for addition, then two values, 10 and 20 to get 30.

###### Subtraction (Option 2)

Use case: Subtracts the second number from the first.

User inputs two numbers.

Example: Enter 2, then two values 20 and 10 to get 10.

##### Multiplication (Option 3)

Use case: Multiplies two numbers.

User inputs two numbers.

Example: On the command line option 3 for multiplication, then two values 10 and 20 to get 200.

##### Division (Option 4)

Description: Divides the first number by the second. Handles division by zero.

Input: Two numbers.

Example: Enter 4 for division, then 20 and 10 to get 2.

##### 4.2 Power and Roots

###### Power (Option 5)

Description: Raises the first number to the power of the second.

Input: Two numbers (base and exponent).

Example: Enter 5 for power, then 2 and 3 to get 8.

###### Square (Option 6)

Description: Squares a number.

Input: One number.

Example: Enter 6 for square, then 4 to get 16.

###### Cube (Option 7)

Description: Cubes a number.

Input: One number.

Example: Enter 7 for cube, then 3 to get 27.

###### Square Root (Option 8)

Description: Computes the square root of a number.

Input: One number.

Example: Enter 8 for square root, then 16 to get 4.

##### 4.3 Rounding Operations

###### Round (Option 9)

Description: Rounds a number to the nearest integer.

Input: One number.

Example: Enter 9 for round, then 4.5 to get 5.

###### Ceiling (Option 10)

Description: Finds the smallest integer greater than or equal to the number.

Input: One number.

Example: Enter 10 for ceiling, then 4.3 to get 5.

###### Floor (Option 11)

Description: Finds the largest integer less than or equal to the number.

Input: One number.

Example: Enter 11 for floor, then 4.7 to get 4.

##### 4.4 Min and Max

###### Min Value (Option 12)

Description: Finds the minimum of two numbers.

Input: Two numbers.

Example: Enter 12 for min value, then 3 and 5 to get 3.

###### Max Value (Option 13)

Description: Finds the maximum of two numbers.

Input: Two numbers.

Example: Enter 13 for max value, then 3 and 5 to get 5.

##### 4.5 Trigonometric Functions

###### Sin (Option 14)

Description: Computes the sine of an angle in degrees.

Input: One number (angle in degrees).

Example: Enter 14 for sin, then 30 to get 0.5.

###### Cos (Option 15)

Description: Computes the cosine of an angle in degrees.

Input: One number (angle in degrees).

Example: Enter 15 for cos, then 60 to get 0.5.

###### Asin (Option 16)

Description: Computes the arcsine in degrees.

Input: One number.

Example: Enter 16 for asin, then 0.5 to get 30.

###### Acos (Option 17)

Description: Computes the arccosine in degrees.

Input: One number.

Example: Enter 17 for acos, then 0.5 to get 60.

###### Atan (Option 18)

Description: Computes the arctangent in degrees.

Input: One number.

Example: Enter 18 for atan, then 1 to get 45.

##### 4.6 Exponential and Logarithmic Functions

###### Exponential (Option 19)

Description: Computes the exponential of a number.

Input: One number.

Example: Enter 19 for exponential, then 2 to get 7.38905609893065.

##### 4.7 Number Theory Operations

###### Palindrome (Option 20)

Description: Checks if a number is a palindrome.

Input: One number.

Example: Enter 20 for palindrome, then 121 to get true.

###### Armstrong Number (Option 21)

Description: Checks if a number is an Armstrong number.

Input: One number.

Example: Enter 21 for Armstrong, then 153 to get true.

###### Prime Number (Option 22)

Description: Checks if a number is a prime number.

Input: One number.

Example: Enter 22 for prime, then 17 to get true.

##### 4.8 Statistical Functions

###### Average (Option 23)

Description: Computes the average of a list of numbers.

Input: List of numbers.

Example: Enter 23 for average, then 3 (number of entries), followed by 10, 20, and 30 to get 20.

##### 4.9 Mathematical Functions

###### GCD (Option 24)

Description: Computes the greatest common divisor of two numbers.

Input: Two numbers.

Example: Enter 24 for GCD, then 8 and 12 to get 4.

###### LCM (Option 25)

Description: Computes the least common multiple of two numbers.

Input: Two numbers.

Example: Enter 25 for LCM, then 4 and 5 to get 20.

##### 4.10 Exiting the Calculator

###### Exit (Option 26)

Description: Exits the calculator program.

Example: Enter 26 to exit the program.

##### 5. Exiting the Calculator

To exit the calculator, simply choose the "Exit" option by Enter ing 26. The program will terminate and display a goodbye message.

##### 6. Troubleshooting

Common Issues and Solutions

###### **Invalid Option Selected:**

Issue: Typing a number not listed in the menu.

Solution: Re-enter a valid option from the menu.

###### **Division by Zero:**

Issue: Attempting to divide by zero.

Solution: The program will display an error message and prompt for valid input.

###### Non-Numeric Input:

Issue: Typing non-numeric characters when a number is expected.

Solution: Ensure to enter valid numbers as prompted.

## DEVELOPERS GUIDE

### Introduction

This guide provides detailed instructions for developers who want to understand, modify, or extend the Console-Based Java Calculator. The calculator is designed to be simple, modular, and easy to maintain.

### Prerequisites

* **Java Development Kit (JDK):** Ensure that the JDK is installed and the JAVA\_HOME environment variable is properly configured.
* **Java IDE/Text Editor:** Any IDE or text editor like IntelliJ IDEA, Eclipse, or VS Code can be used.
* **Basic Java Knowledge:** Familiarity with Java syntax, OOP concepts, and basic file handling.

#### Project Structure

Assuming the following project structure:

/CalculatorProject

│

├── src

│ ├── Calculator.java

│ └── Operations.java

│

└── README.md

* **Calculator.java:** The main class responsible for handling user input, displaying the menu, and invoking the appropriate operations.
* **Operations.java (Optional):** A separate class where mathematical operations are implemented.

## Setting Up the Development Environment

1. **Open the Project in Your IDE:** Import the project into your preferred IDE.
2. **Compile and Run:** Before making any changes, compile and run the existing code to ensure everything works as expected.

## Code Walkthrough

Refer to the source code on page 8 for the detailed code breakdown and the comments after each piece of code for better understanding of the code.

### Code Explanation

* **Scanner Object:** Used for capturing user input.
* **While Loop:** The program runs continuously until the user selects the exit option.
* **Switch Case:** Handles user selection and invokes the corresponding mathematical operation.
* **Error Handling:** Includes basic error checks like division by zero.

### Refactoring and Optimization

* **Modularization:** Consider moving operations into separate classes or packages if the codebase grows.
* **Input Validation:** Add more robust input validation to handle non-numeric input or other edge cases.
* **Exception Handling:** Expand exception handling to manage unexpected errors gracefully.

### Testing

* **Unit Tests:** Write unit tests for each method in Operations.java to ensure they work correctly.
* **Integration Tests:** Test the calculator as a whole to verify that all components interact as expected.

## Conclusion

This developer's guide provides a comprehensive overview of how to work with and maintain the Console-Based Java Calculator. The codebase is simple but flexible enough to be understood and maintained.

# Module Description

## Overview

The Console-Based Calculator module provides a simple and interactive command-line interface for performing basic arithmetic and scientific calculations. This module is designed to handle common mathematical operations, including addition, subtraction, multiplication, division, exponentiation, and various trigonometric and logarithmic functions.

## Key Features

* **Basic Arithmetic Operations:** Perform addition, subtraction, multiplication, and division with real numbers.
* **Exponentiation:** Calculate the power of numbers.
* **Square Root Calculation:** Find the square root of a number.
* **Trigonometric Functions:** Compute sine, cosine, and tangent values for angles given in degrees.
* **Logarithmic Functions:** Calculate the logarithm of a number in both base 10 and natural logarithm (ln).
* **User-Friendly Interface:** A text-based menu guides users through the selection of operations and input of data.

## Structure

* **Main Class (Calculator.java):** This is the entry point of the module. It initializes the calculator, displays the menu, and handles user input. The main class also manages the flow of the program, calling appropriate methods based on the user’s choices.
* **Operations Class (Optional):** If your design includes an operations class, this class encapsulates the core functionality for each mathematical operation, providing clean, modular methods that can be called from the main class.

## Usage

Users interact with the calculator by selecting options from a text-based menu. After selecting an operation, the calculator prompts the user for necessary inputs, processes the data, and then displays the result. The program continues to prompt the user until they choose to exit.

## Error Handling

The module includes basic error handling for common issues, such as division by zero and invalid numeric inputs. It prompts users to correct their input in case of errors, ensuring a smooth and continuous experience.

## Dependencies

This module has no external dependencies and runs on any system with a Java runtime environment.

## Extensibility

The design of the Console-Based Calculator is modular, making it easy to extend with additional mathematical operations or features. Future enhancements might include support for more advanced functions, complex numbers, or a graphical user interface.

## Conclusion

The Console-Based Calculator module offers a straightforward and reliable solution for performing a variety of mathematical calculations through a command-line interface. Its simple design and easy-to-use interface make it suitable for educational purposes, quick calculations, or as a base for more advanced projects.