**Building a Simple Calculator in Java**

**K. Naga Vihari**

Guest Faculty, Department of Statistics, Sri Venkateswara University, Tirupati, AP, India.

**Abstract:** Creating a simple calculator in Java is a great way to learn about basic programming principles like user input, arithmetic operations, and control flow. This article walks readers through the steps of creating a Java application that executes basic arithmetic operations (addition, subtraction, multiplication, and division) based on user input. The software may perform user-defined operations efficiently while managing failures such as division by zero by using conditional statements and switch-case logic. The article not only includes a step-by-step code implementation, but it also recommends potential improvements, such as adding more operations, continuous computations, and a graphic user interface. This project is suitable for novices who want to strengthen their knowledge of Java programming essentials.

**Keywords:**Simple Calculator, Arithmetic Operations, Java Programming.

**Introduction**

Calculators are one of the most simple yet important instruments in our daily lives, allowing us to execute arithmetic operations quickly and correctly. In this post, we'll look at how to build a simple Java calculator that can do fundamental arithmetic operations such as addition, subtraction, multiplication and division.This software is an excellent beginning place for newcomers to Java and object-oriented programming ideas. It covers fundamental concepts like as user input, conditional expressions, and switch-case logic.

**Understanding the Java Calculator Program**

The Java calculator software employs fundamental control flow techniques to accept user input and conduct arithmetic operations based on the user's selection.

**Java Program Implementation**

Below is the code for the simple calculator in Java:

java

Copy code

importjava.util.Scanner;

public class SimpleCalculator {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Read the first number from the user

System.out.print("Enter the first number: ");

double num1 = scanner.nextDouble();

// Read the second number from the user

System.out.print("Enter the second number: ");

double num2 = scanner.nextDouble();

// Read the operation from the user

System.out.print("Enter the operation (+, -, \*, /): ");

char operation = scanner.next().charAt(0);

// Perform the calculation based on the operation

double result;

switch (operation) {

case '+':

result = num1 + num2;

System.out.println("Result: " + num1 + " + " + num2 + " = " + result);

break;

case '-':

result = num1 - num2;

System.out.println("Result: " + num1 + " - " + num2 + " = " + result);

break;

case '\*':

result = num1 \* num2;

System.out.println("Result: " + num1 + " \* " + num2 + " = " + result);

break;

case '/':

// Check for division by zero

if (num2 != 0) {

result = num1 / num2;

System.out.println("Result: " + num1 + " / " + num2 + " = " + result);

} else {

System.out.println("Error: Division by zero is not allowed.");

}

break;

default:

System.out.println("Error: Invalid operation. Please enter one of +, -, \*, /.");

break;

}

// Close the scanner

scanner.close();

}

}

**Explanation of the Code**

1. Reading User Input: :The application reads user input using a Scanner object. It asks the user to enter two integers and an arithmetic operation (addition, subtraction, multiplication, or division).
2. Operation Selection: The program utilizes a switch statement to select the appropriate operation based on the user's input (+, -, \*, or /). Depending on the operator selected, the relevant arithmetic operation is performed.
3. Handling Division by Zero: When the user picks division (/), the software determines if the second number is zero. If it is, the software returns an error since division by zero is undefined.
4. Displaying the outcome: The software clearly displays the outcome of the chosen operation, including the whole expression (e.g., "12.0 \* 56.0 = 672.0").
5. Error Handling for Invalid Operations: If a user enters an invalid operator (other than +, -, \*, or /), the software displays an error message and prompts them to provide a correct operation.

**Sample Output**

Here’s a sample run of the program:

Mathematical Copy code

Enter the first number: 12

Enter the second number: 56

Enter the operation (+, -, \*, /): \*

Result: 12.0 \* 56.0 = 672.0

This Java project is ideal for beginners as it teaches crucial topics like user input validation.   
• Use conditional statements and switch-case logic.   
• Perform arithmetic operations.   
• Perform simple error checks, such as division by zero.

**Possible Enhancements**

While this calculator is rudimentary, it may be enhanced with the following features:   
1. Enhance the software to support more operations such as modulus (%), exponentiation, and square root.   
2. Modify the software to handle continuous calculations rather than quitting after a single action.   
3. Consider adding a graphical user interface to the calculator, such as Java Swing or JavaFX, to improve usability.

**Conclusion**

This article showed how to create a small Java calculator that can do basic arithmetic calculations. The program is basic, yet it successfully covers some key programming ideas that all beginners should understand. As you progress in your programming career, you may expand on this foundation by adding more complex functions and making the calculator more powerful.

**References**

1. **W3Schools:** Java Switch Statements
2. **GeeksforGeeks:** Java User Input (Scanner Class)