CERTIFICATE

This is to certify that [Your Name], a student of Class XII-[Section] at [Your School Name], has successfully completed the project titled **“---------------------------”** under the supervision of [Project Supervisor Name] during the academic year [2025-26].

This project has been completed in fulfillment of the requirements for the Computer Science course and adheres to the guidelines issued by CBSE, New Delhi.

The project work meets the expected standards and has been evaluated by the undersigned.

Signature of Principal: __________________________

Signature of Supervisor: ________________________

Signature of External Examiner: ________________

Date: _____________

Place: _____________

Student Details:

Student Names:

Class : XII

Subject: Information Technology

School Name: Torch Bearers Convent School

Affiliation By.: CBSE Board

ABSTRACT

The Simple Calculator project is a software application developed to perform basic arithmetic operations, including addition, subtraction, multiplication, and division on two numbers. The project demonstrates the implementation of decision-making structures such as if-else and switch-case statements for selecting the desired operation based on user input. It also showcases the use of methods (functions) to modularize code for different arithmetic operations, enhancing code organization and reusability. This calculator provides a straightforward and interactive way to understand fundamental programming concepts like control flow and functional decomposition while supporting accurate and efficient computation of basic mathematical tasks.

This project was developed by Deepanshu, Devanshu, Prince, and Rahul (Jha) and serves as an educational tool for learning programming logic, method usage, and simple user input handling in Java.

Table of Contents

Introduction

The Simple Calculator project is a fundamental application designed to perform basic arithmetic operations such as addition, subtraction, multiplication, and division on two numbers. This project demonstrates crucial programming concepts like decision-making using if-else and switch-case statements, and modular programming through methods (functions). The calculator provides a user-friendly interface where users can input two numbers and select the desired operation, making it an effective learning tool for understanding control flow and functional decomposition in programming. By developing this project, team members gain practical experience in writing clean, organized code that can handle different operations with appropriate validation and error handling, such as managing division by zero. This project embodies essential programming skills useful for building more complex software applications.

Objectives

1. To develop a simple calculator application using Java that performs basic arithmetic operations such as addition, subtraction, multiplication, and division.
2. To implement decision-making in the program using if-else and switch-case statements for selecting operations.
3. To use methods (functions) for modular and reusable code to perform arithmetic calculations on two input numbers.
4. To create a user-friendly interface that allows users to input numbers and select operations easily.
5. To handle error cases like division by zero and invalid input gracefully through proper validation and decision structures.
6. To enhance understanding of fundamental programming concepts like conditional statements, methods, user input handling, and basic arithmetic processing.
7. To test and validate the calculator for accuracy and robustness in processing different arithmetic operations.

System Design and Classes

The Simple Calculator project is designed using a single class named SimpleCalculator which encapsulates all the core functionalities. This design promotes modularity and easy maintenance by using methods for each arithmetic operation.

Class: SimpleCalculator

Description:
This class contains methods to perform basic arithmetic operations such as addition, subtraction, multiplication, division, and also supports decision-making using if-else and switch-case constructs to select the correct operation based on user input.

Methods Overview

- public double add(double a, double b): Returns the sum of a and b.
- public double subtract(double a, double b): Returns the result of a minus b.
- public double multiply(double a, double b): Returns the product of a and b.
- public double divide(double a, double b): Performs division of a by b, includes check for division by zero to avoid errors.
- public void calculate(char operator, double num1, double num2): Uses a switch-case statement to decide which arithmetic method to call based on the operator provided (‘+’, ‘-’, ‘*’, ‘/’).

Design Flow

1. The user is prompted to input two numbers.
2. The user selects the operator for the operation.
3. The calculate method interprets the operator and calls the corresponding arithmetic method.
4. Result is computed and displayed.
5. Input validation and error handling are included, particularly checking for invalid operators and division by zero scenarios.

Simplified Class Diagram

- **SimpleCalculator**
  - Methods:
    - add(double, double): double
    - subtract(double, double): double
    - multiply(double, double): double
    - divide(double, double): double
    - calculate(char, double, double): void

Sample Pseudocode

text

```
class SimpleCalculator {
  method add(a, b) returns a + b
  method subtract(a, b) returns a - b
  method multiply(a, b) returns a * b
  method divide(a, b) {
    if b == 0
      display error "Division by zero not allowed"
    else
      returns a / b
  }

  method calculate(operator, num1, num2) {
    switch operator:
      case '+': call add(num1, num2)
      case '-': call subtract(num1, num2)
      case '*': call multiply(num1, num2)
      case '/': call divide(num1, num2)
      default: display "Invalid operator"
  }
}
```

This design allows clear separation of concerns, with each method dedicated to a single operation and the decision logic centralized in the calculate method. It demonstrates fundamental concepts of decision-making and method usage in Java programming effectively.

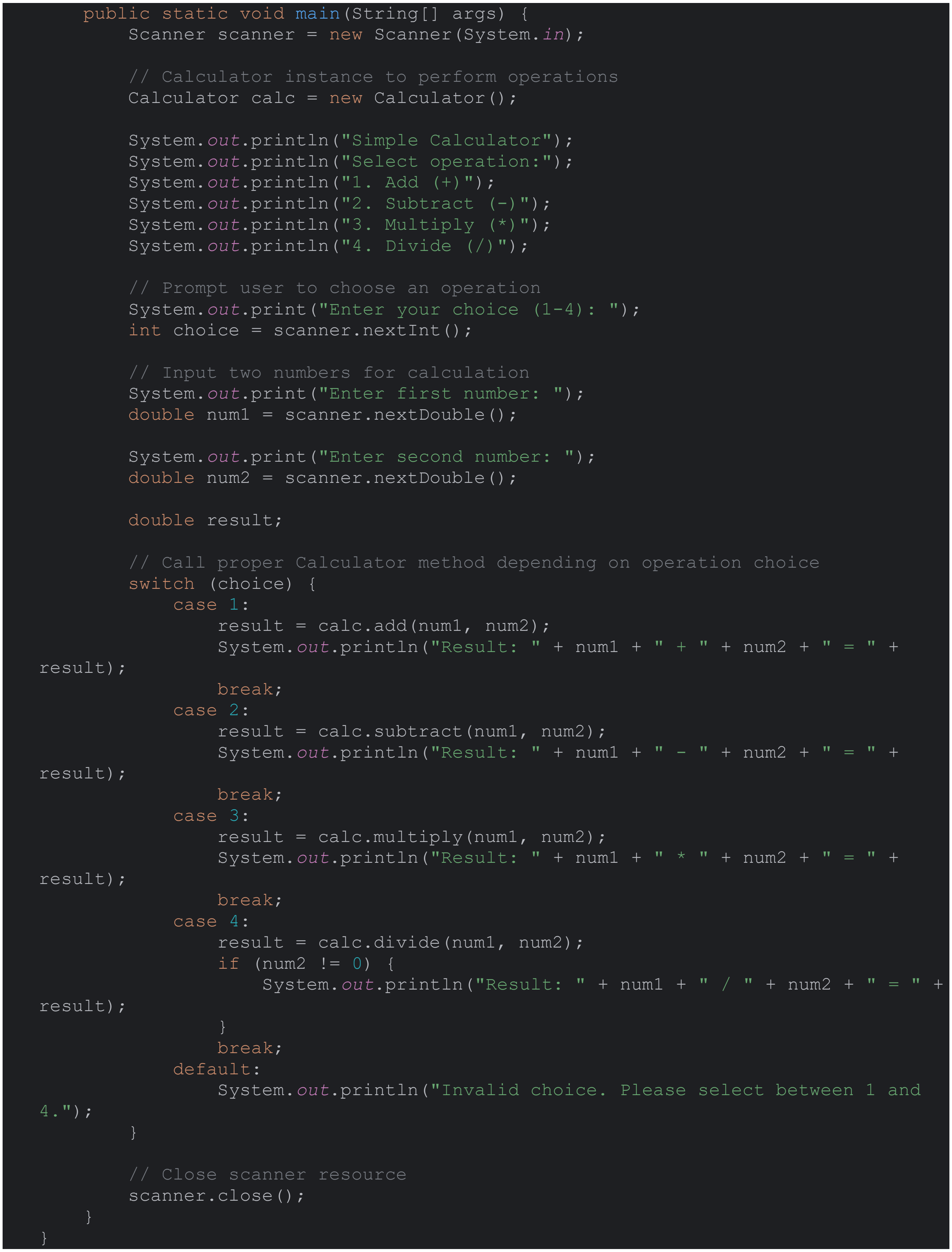
Core Components and Code Snippets

- Calculator.java

```java
/**
 * Calculator class provides basic arithmetic operations.
 * This keeps operation logic separate from UI/input handling.
 */
public class Calculator {

    /**
     * Adds two numbers.
     * @param a first number
     * @param b second number
     * @return sum of a and b
     */
    public double add(double a, double b) {
        return a + b;
    }

    /**
     * Subtracts second number from the first.
     * @param a first number
     * @param b second number
     * @return result of a minus b
     */
    public double subtract(double a, double b) {
        return a - b;
    }

    /**
     * Multiplies two numbers.
     * @param a first number
     * @param b second number
     * @return product of a and b
     */
    public double multiply(double a, double b) {
        return a * b;
    }

    /**
     * Divides first number by second, with zero check.
     * Returns 0 and prints error if dividing by zero.
     * @param a numerator
     * @param b denominator
     * @return quotient if valid, else 0
     */
    public double divide(double a, double b) {
        if (b == 0) {
            System.out.println("Error: Division by zero is not allowed.");
            return 0;
        }
        return a / b;
    }
}
```

- CalculatorApp.java

```java
import java.util.Scanner;

/**
 * Main class to run the calculator app.
 * Handles user input/output and calls Calculator methods.
 */
public class CalculatorApp {
```



```java
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Calculator instance to perform operations
        Calculator calc = new Calculator();

        System.out.println("Simple Calculator");
        System.out.println("Select operation:");
        System.out.println("1. Add (+)");
        System.out.println("2. Subtract (-)");
        System.out.println("3. Multiply (*)");
        System.out.println("4. Divide (/)");

        // Prompt user to choose an operation
        System.out.print("Enter your choice (1-4): ");
        int choice = scanner.nextInt();

        // Input two numbers for calculation
        System.out.print("Enter first number: ");
        double num1 = scanner.nextDouble();

        System.out.print("Enter second number: ");
        double num2 = scanner.nextDouble();

        double result;

        // Call proper Calculator method depending on operation choice
        switch (choice) {
            case 1:
                result = calc.add(num1, num2);
                System.out.println("Result: " + num1 + " + " + num2 + " = " +
result);
                break;
            case 2:
                result = calc.subtract(num1, num2);
                System.out.println("Result: " + num1 + " - " + num2 + " = " +
result);
                break;
            case 3:
                result = calc.multiply(num1, num2);
                System.out.println("Result: " + num1 + " * " + num2 + " = " +
result);
                break;
            case 4:
                result = calc.divide(num1, num2);
                if (num2 != 0) {
                    System.out.println("Result: " + num1 + " / " + num2 + " = " +
result);
                }
                break;
            default:
                System.out.println("Invalid choice. Please select between 1 and
4.");
        }

        // Close scanner resource
        scanner.close();
    }
}
```

Transaction Handling

In the Simple Calculator project, transaction handling refers to the process flow where the program takes two numerical inputs from the user and performs the selected arithmetic operation (addition, subtraction, multiplication, or division) using decision-making constructs like if-else or switch-case statements. The program uses methods (functions) to modularize each arithmetic operation for better readability and reusability.

Steps in Transaction Handling:

1. **User Input Selection:**
   - The program prompts the user to select an arithmetic operation by entering a choice, such as 1 for addition, 2 for subtraction, etc.
   - This selection is handled using either an if-else ladder or a switch-case construct, which guides the program to the corresponding operation.
2. **Input of Operands:**
   - After selecting the operation, the user is asked to input two numbers on which the arithmetic operation will be performed.
   - Input validation is done to ensure only valid numbers are entered, and the program can handle exceptions or invalid inputs gracefully.
3. **Performing the Operation:**
   - Based on the operation chosen, control transfers to the appropriate method:
     - add(num1, num2) returns the sum.
     - subtract(num1, num2) returns the difference.
     - multiply(num1, num2) returns the product.
     - divide(num1, num2) performs division with additional logic to check for division by zero to avoid errors.
   - Each method encapsulates the logic for the respective operation and returns the result.
4. **Displaying the Result:**
   - The result from the called method is displayed on the screen.
   - Proper formatting is applied for readability.
5. **Handling Errors:**
   - The program includes checks for invalid operator choices.
   - Division by zero is handled by displaying an error message instead of attempting the calculation.
   - Invalid inputs for numbers prompt the user to re-enter the values.
6. **Repeat Transaction:**
   - After completion of a calculation, the user is prompted whether to perform another calculation.
   - If the user chooses yes, the program loops back to step 1.
   - Otherwise, the program terminates.

Example of Transaction Handling using Switch-case in Java

java

```java
import java.util.Scanner;

public class SimpleCalculator {

    static double add(double a, double b) { return a + b; }
    static double subtract(double a, double b) { return a - b; }
    static double multiply(double a, double b) { return a * b; }
    static double divide(double a, double b) {
        if (b == 0) {
            System.out.println("Error: Division by zero is not allowed.");
            return Double.NaN;
        } else
            return a / b;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        char continueCalc;

        do {
            System.out.println("Choose operation: 1.Add 2.Subtract 3.Multiply 4.Divide");
            int choice = sc.nextInt();

            System.out.print("Enter first number: ");
            double num1 = sc.nextDouble();
            System.out.print("Enter second number: ");
            double num2 = sc.nextDouble();

            double result = 0;
            boolean valid = true;

            switch(choice) {
                case 1: result = add(num1, num2); break;
                case 2: result = subtract(num1, num2); break;
                case 3: result = multiply(num1, num2); break;
                case 4: result = divide(num1, num2); break;
                default:
                    System.out.println("Invalid operation choice.");
                    valid = false;
                    break;
            }

            if(valid && !Double.isNaN(result))
                System.out.println("Result: " + result);

            System.out.print("Perform another calculation? (y/n): ");
            continueCalc = sc.next().charAt(0);

        } while (continueCalc == 'y' || continueCalc == 'Y');

        sc.close();
    }
}
```

This program represents the transaction handling logic where the user continuously performs calculations until opting to exit. It demonstrates decision making, functions, error checks, and user interaction flow clearly.

Conclusion and Team Contribution

The Simple Calculator project successfully demonstrates the implementation of basic arithmetic operations—addition, subtraction, multiplication, and division—using decision-making constructs such as if-else and switch-case along with modular functions (methods). The application efficiently performs arithmetic operations on two input numbers, handling user input and invalid cases like division by zero gracefully. This project provided practical experience in applying core programming concepts such as control flow, conditional statements, and function-based decomposition, forming a foundation for more advanced programming tasks. Through this project, the team learned to design, develop, and test a functional program in a structured manner, emphasizing clarity and maintainability of code.

Team Contribution

- **Deepanshu:** Led the overall project design and wrote the main program logic including arithmetic methods.
- **Devanshu:** Developed the user input handling modules and implemented decision-making logic using if-else and switch-case conditions.
- **Prince:** Responsible for testing the program, handling error cases, and ensuring the calculator performs accurate calculations.
- **Rahul (Jha):** Documented the project report, created flowcharts and pseudocode, and contributed to final project presentation.

Each member played a key role in the successful completion of this project through collaborative coding, testing, and documentation efforts.