K.R. Mangalam University

School of Engineering & Technology

Fundamentals Of Java Programming

Assignment 2

Calculator Application Using Method Overloading

Submitted by:
Name: RAKESH G
Roll No: 2401201064
Course: BCA (AI & DS)

Submitted To:
Dr. Manish Kumar

CODE:

JAVA > JavaAssignment2.java > ...

```java
// Assignment 02: Calculator Application Using Method Overloading
// Submitted by: Rakesh G
// University: K.R. Mangalam University
// Course: BCA (AI & DS)

import java.util.Scanner;

class Calculator {

    // Method Overloading for Addition
    int add(int a, int b) {    add is never used
        return a + b;
    }

    double add(double a, double b) {
        return a + b;
    }

    int add(int a, int b, int c) {    add is never used
        return a + b + c;
    }

    // Subtraction
    int subtract(int a, int b) {
        return a - b;
    }

    // Multiplication
    double multiply(double a, double b) {
        return a * b;
    }

    // Division (with divide-by-zero handling)
    double divide(int a, int b) {
        try {
            if (b == 0) {
                throw new ArithmeticException("Division by zero is not allowed.");
            }
```

```java
class Calculator {
    double divide(int a, int b) {
                throw new ArithmeticException("Division by zero is not allowed.");
            }
            return (double) a / b;
        } catch (ArithmeticException e) {
            System.out.println("Error: " + e.getMessage());
            return 0;
        }
    }
}

public class JavaAssignment2 {

    Scanner sc = new Scanner(System.in);
    Calculator calc = new Calculator();


    void performAddition() {
        System.out.print("\nEnter first number: ");
        double a = sc.nextDouble();
        System.out.print("Enter second number: ");
        double b = sc.nextDouble();
        System.out.println("Result: " + calc.add(a, b));
    }

    void performSubtraction() {
        System.out.print("\nEnter first integer: ");
        int a = sc.nextInt();
        System.out.print("Enter second integer: ");
        int b = sc.nextInt();
        System.out.println("Result: " + calc.subtract(a, b));
    }


    void performMultiplication() {
        System.out.print("\nEnter first number: ");
        double a = sc.nextDouble();
```

```
47  public class JavaAssignment2 {
70      void performMultiplication() {
72          double a = sc.nextDouble();
73          System.out.print("Enter second number: ");
74          double b = sc.nextDouble();
75          System.out.println("Result: " + calc.multiply(a, b));
76      }
77
78
79      void performDivision() {
80          System.out.print("\nEnter dividend (int): ");
81          int a = sc.nextInt();
82          System.out.print("Enter divisor (int): ");
83          int b = sc.nextInt();
84          double result = calc.divide(a, b);
85          System.out.println("Result: " + result);
86      }
87
88
89      void mainMenu() {
90          int choice;
91          do {
92              System.out.println("\n=== Welcome to the Calculator Application ===");
93              System.out.println("1. Add Numbers");
94              System.out.println("2. Subtract Numbers");
95              System.out.println("3. Multiply Numbers");
96              System.out.println("4. Divide Numbers");
97              System.out.println("5. Exit");
98              System.out.print("Enter your choice: ");
99              choice = sc.nextInt();
100
101             switch (choice) {    Convert switch to rule switch
102                 case 1:
103                     performAddition();
104                     break;
105                 case 2:
106                     performSubtraction();
```

JAVA > JavaAssignment2.java > ...

```
47  public class JavaAssignment2 {
89      void mainMenu() {
95              System.out.println("3. Multiply Numbers");
96              System.out.println("4. Divide Numbers");
97              System.out.println("5. Exit");
98              System.out.print("Enter your choice: ");
99              choice = sc.nextInt();
100
101             switch (choice) {    Convert switch to rule switch
102                 case 1:
103                     performAddition();
104                     break;
105                 case 2:
106                     performSubtraction();
107                     break;
108                 case 3:
109                     performMultiplication();
110                     break;
111                 case 4:
112                     performDivision();
113                     break;
114                 case 5:
115                     System.out.println("Thank you for using the Calculator. Goodbye!");
116                     break;
117                 default:
118                     System.out.println("Invalid option. Please try again!");
119             }
120         } while (choice != 5);
121     }
122
        Run | Debug | Run main | Debug main
123     public static void main(String[] args) {
124         JavaAssignment2 ui = new JavaAssignment2();
125         ui.mainMenu();
126     }
127 }
128
```

OUTPUT:

```
=== Welcome to the Calculator Application ===
1. Add Numbers
2. Subtract Numbers
3. Multiply Numbers
4. Divide Numbers
5. Exit
Enter your choice: 1

Enter first number: 5
Enter second number: 5
Result: 10.0

=== Welcome to the Calculator Application ===
1. Add Numbers
2. Subtract Numbers
3. Multiply Numbers
4. Divide Numbers
5. Exit
Enter your choice: 2

Enter first integer: 3
Enter second integer: 2
Result: 1

=== Welcome to the Calculator Application ===
1. Add Numbers
2. Subtract Numbers
3. Multiply Numbers
4. Divide Numbers
5. Exit
Enter your choice: 3

Enter first number: 2
Enter second number: 3
```

```
=== Welcome to the Calculator Application ===
1. Add Numbers
2. Subtract Numbers
3. Multiply Numbers
4. Divide Numbers
5. Exit
Enter your choice: 3

Enter first number: 2
Enter second number: 3
Result: 6.0

=== Welcome to the Calculator Application ===
1. Add Numbers
2. Subtract Numbers
3. Multiply Numbers
4. Divide Numbers
5. Exit
Enter your choice: 4

Enter dividend (int): 4
Enter divisor (int): 0
Error: Division by zero is not allowed.
Result: 0.0

=== Welcome to the Calculator Application ===
1. Add Numbers
2. Subtract Numbers
3. Multiply Numbers
4. Divide Numbers
5. Exit
Enter your choice: 5
Thank you for using the Calculator. Goodbye!

d:\RAKESH\VSC\JAVA>
```

Explanation of the Code

The program demonstrates the concept of method overloading in Java by creating a Calculator Application that performs various arithmetic operations such as addition, subtraction, multiplication, and division. Method overloading allows multiple methods to have the same name but different parameter lists, enabling the same function name to work with different data types or numbers of arguments.

The program defines a class named Calculator which contains several add() methods, each overloaded to handle different types of data such as integers, doubles, and three-number additions. For example, one method adds two integers, another adds two double values, and another adds three numbers. This demonstrates how Java can automatically determine which method to call based on the arguments passed (a concept known as compile-time polymorphism).

Inside the main() method, the program first displays a menu and asks the user to choose the operation they want to perform. Using a Scanner object, the user's input is taken for different arithmetic operations. Based on the user's choice, the corresponding overloaded method is called to perform the selected operation.

The program also includes exception handling to manage invalid inputs gracefully. For example, if the user enters a non-numeric value or tries to divide by zero, the program catches the exception and displays an error message instead of crashing.

After performing the selected arithmetic operation, the result is displayed on the screen. The use of method overloading makes the program more flexible, readable, and organized.

Key Concepts Used

Concept	Description
Method Overloading	Multiple methods with the same name but different parameters.

Concept	Description
Compile-Time Polymorphism	The method to be executed is decided during compilation.
Scanner Class	Used to take user input for performing operations.
Exception Handling	Used to handle invalid inputs and division by zero safely.
Object-Oriented Design	The program is based on class and object concepts in Java.

Conclusion

This assignment demonstrates how method overloading can be used to create a flexible and efficient Calculator Application. It shows the practical use of object-oriented programming principles, compile-time polymorphism, and exception handling in Java to build real-world applications that are both robust and user-friendly.