# Java Calculator Program: Key Components and Their Importance

## Overview

This program demonstrates the use of encapsulation, method overloading, and user input in Java. It implements a simple calculator to perform addition operations on two or three numbers entered by the user.

## Key Components

### 1. Private Instance Variables

The private variables `num1`, `num2`, and `num3` ensure that the data members of the class are not directly accessible from outside the class. This encapsulation enhances data security and enforces controlled access through getter and setter methods.

### 2. Getter and Setter Methods

Getter and setter methods allow controlled access and modification of private fields. For example, the `setNum1(int num1)` method sets the value of `num1`, while `getNum1()` retrieves it. This approach provides flexibility to include validation logic in the future.

### 3. Method Overloading

The program demonstrates method overloading with two `add` methods. The first method takes two parameters (`int num1, int num2`) to calculate the sum of two numbers, while the second method takes three parameters (`int num1, int num2, int num3`) to calculate the sum of three numbers. Method overloading allows the same method name to be used with different parameter lists, enhancing code readability.

### 4. Scanner for User Input

The `Scanner` class is used to take input from the console. It allows the user to enter values for `num1`, `num2`, and `num3`, which are then used for the addition operations. The `scanner.close()` statement ensures that resources are freed once input is no longer needed.

### 5. Main Method

The `main` method serves as the entry point for the program. It creates an instance of the `Calculator` class, collects input from the user using the `Scanner` class, and demonstrates the use of getter, setter, and overloaded methods to perform and display the addition operations.

## Program Output

1. The program prompts the user to enter two numbers, calculates their sum using the two-parameter `add` method, and displays the result.  
2. It then prompts the user to enter a third number, calculates the sum of three numbers using the three-parameter `add` method, and displays the result.  
3. The `scanner.close()` statement ensures proper resource management.

## Best Practices Followed

- Use of encapsulation to restrict direct access to class fields.  
- Controlled access through getter and setter methods.  
- Demonstration of method overloading to handle different parameter lists.  
- Proper resource management by closing the `Scanner` after use.