# Write A Java Program That Takes 10 Integers From The User, Calculate Their Sum And Average, And Print

### Write A Java Program That Takes 10 Integers From The User, Calculate Their Sum And Average, And Print

Creating a Java program that accepts multiple user inputs, performs calculations, and displays the results is a fundamental skill for any aspiring programmer. In this guide, we will walk through how to develop a Java application that prompts the user to input ten integers, computes their total sum and average, and then outputs these results clearly. Whether you're a beginner looking to understand basic input handling or an experienced developer refining your skills, this comprehensive tutorial will help you write an efficient, user-friendly program.

---

## Understanding the Program Requirements

Before diving into the coding process, it's essential to understand what the program needs to accomplish:

### Key Objectives

- Accept 10 integers from the user via the console.
- Calculate the sum of these integers.
- Compute the average of the entered numbers.
- Display the sum and average in a clear format.

## Why This Program Is Useful

- Practices user input handling in Java.
- Demonstrates basic arithmetic operations.
- Introduces concepts of loops, arrays, and input validation.
- Provides a foundation for more complex data processing programs.

---

## Prerequisites and Setup

Before coding, ensure you have the following:

## Development Environment

- Java Development Kit (JDK) installed on your machine. (Version 8 or higher recommended)
- An IDE such as Eclipse, IntelliJ IDEA, or NetBeans, or a simple text editor like VS Code or Notepad++.
- Basic understanding of Java syntax, variables, loops, and input/output operations.

## Setting Up Your Project

1. Create a new Java project or file named `SumAndAverage.java`.
2. Ensure your IDE or editor is configured to compile and run Java programs.
3. Save your file in a directory accessible for compilation and execution.

---

# Step-by-Step Implementation

In this section, we'll build the Java program step-by-step, explaining each part of the code.

## 1. Import Necessary Packages

Java's `Scanner` class from `java.util` package helps read user input.

```java
import java.util.Scanner;
```

## 2. Declare the Main Class and Method

Create a class named `SumAndAverage` with the primary `main` method.

```java
public class SumAndAverage {
public static void main(String[] args) {
// Program code will go here
}
}
```

## 3. Initialize Variables

Declare variables to store:

- The total sum (`int sum`)

- The number of inputs (`int count`)
- An array to store the 10 integers (`int[] numbers`)
- The average (`double average`)

```java
int[] numbers = new int[10];
int sum = 0;
double average;
```

## 4. Set Up the Scanner for User Input

Create a `Scanner` object to read from the console.

```java
Scanner scanner = new Scanner(System.in);
```

## 5. Collect User Inputs

Use a loop to prompt the user ten times for integers.

```java
for (int i = 0; i < 10; i++) {
System.out.print("Enter integer " + (i + 1) + ": ");
// Read integer input
while (true) {
if (scanner.hasNextInt()) {
numbers[i] = scanner.nextInt();
break;
} else {
System.out.print("Invalid input. Please enter an integer: ");
scanner.next(); // discard invalid input
}
}
}
```

## 6. Calculate Sum of Inputs

Iterate over the array to compute the total sum.

```java
for (int num : numbers) {
sum += num;
}
```

## 7. Calculate Average

Compute the average as a double for precision.

```java
average = (double) sum / 10;
```

## 8. Display Results

Print the total sum and average with appropriate formatting.

```java
System.out.println("\nResults:");
System.out.println("Sum of the entered integers: " + sum);
System.out.printf("Average of the entered integers: %.2f\n", average);
```

## 9. Close Scanner Resource

It's good practice to close the `Scanner` object when done.

```java
scanner.close();
```

---

# Complete Java Program Code

Putting all the parts together, here is the full implementation:

```java
import java.util.Scanner;

public class SumAndAverage {
public static void main(String[] args) {
int[] numbers = new int[10];
int sum = 0;
double average;

Scanner scanner = new Scanner(System.in);

// Collect 10 integers from the user
for (int i = 0; i < 10; i++) {
System.out.print("Enter integer " + (i + 1) + ": ");
while (true) {
if (scanner.hasNextInt()) {
numbers[i] = scanner.nextInt();
break;
} else {
System.out.print("Invalid input. Please enter an integer: ");
scanner.next(); // discard invalid input
}
}
}

// Calculate sum
for (int num : numbers) {
sum += num;
}

// Calculate average
average = (double) sum / 10;
```

```
// Output the results
System.out.println("\nResults:");
System.out.println("Sum of the entered integers: " + sum);
System.out.printf("Average of the entered integers: %.2f\n", average);

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

---

# Additional Features and Enhancements

While the above code fulfills the core requirements, you might consider adding extra functionalities to make the program more robust and user-friendly.

## Input Validation

- Prevent the program from crashing if the user enters non-integer inputs.
- Use `try-catch` blocks for exception handling instead of `hasNextInt()` checks.

```java
// Example of input validation with exception handling
try {
int input = scanner.nextInt();
numbers[i] = input;
} catch (InputMismatchException e) {
System.out.print("Invalid input. Please enter an integer: ");
scanner.next(); // discard invalid input
i--; // repeat the input prompt
}
```

## Handling Negative and Zero Values

- The current implementation accepts any integer, including negatives and zero.
- If you need only positive integers, you can add checks.

```java
if (input > 0) {
numbers[i] = input;
} else {
System.out.println("Please enter a positive integer.");
i--; // prompt again
}
```

## Extended Calculations

- Find the maximum and minimum values among the inputs.
- Count how many inputs are above or below certain thresholds.
- Display sorted list of entered numbers.

## User Interface Enhancements

- Clear prompts and instructions.
- Re-prompt user if invalid input is detected.
- Use formatted output for better readability.

---

# Testing Your Java Program

Testing is a crucial step to ensure your program functions correctly. Here are some tips:

1. Run the program and input a variety of integers, including positive, negative, and zero.
2. Verify that the sum and average calculations are correct.
3. Test with invalid inputs like strings or special characters to check validation handling.
4. Ensure the output formatting is clear and readable.

Sample input sequence:

```
Enter integer 1: 10
Enter integer 2: 20
Enter integer 3: -5
Enter integer 4: 0
Enter integer 5: 15
Enter integer 6: 25
Enter integer 7: 5
Enter integer 8: -10
Enter integer 9: 30
Enter integer 10: 40
```

Expected output:

```
Results:
Sum of the entered integers: 125
Average of the entered integers: 12.50
```

---

# Conclusion

By following this guide, you learned how to write a simple yet effective Java program that takes ten integers from the user, calculates their sum and average, and displays the results. This foundational skill opens the door to more complex data processing tasks, such as handling larger datasets, performing statistical analyses, or developing interactive applications. Remember to always validate user input and consider edge cases to make your programs robust and user-friendly.

Happy coding!

# Frequently Asked Questions

## How can I read 10 integers from the user in Java?

You can use the Scanner class in Java. Instantiate it with Scanner scanner = new Scanner(System.in); then use a loop to call scanner.nextInt() ten times to read the integers.

## What is the best way to calculate the sum and average of 10 integers in Java?

Initialize a sum variable to 0, add each user-inputted integer to the sum inside a loop, then divide the sum by 10 to get the average as a double for better precision.

## How do I handle invalid inputs when the user enters non-integer values?

Use a try-catch block around scanner.nextInt() to catch InputMismatchException. Prompt the user to enter a valid integer again if an exception occurs.

## Can I write a Java program to process more than 10 integers easily?

Yes, by using a loop with a variable for the number of inputs, you can easily adapt the program to handle any number of integers, not just 10.

## What is a complete example of a Java program that reads 10 integers, calculates their sum and average, and prints the results?

Here's a simple example:

```java
import java.util.Scanner;

public class SumAndAverage {
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
int sum = 0;
int count = 10;
for (int i = 1; i <= count; i++) {
System.out.print("Enter integer " + i +": ");
while (!scanner.hasNextInt()) {
System.out.println("Invalid input. Please enter an integer.");
scanner.next();
}
int num = scanner.nextInt();
sum += num;
}
double average = (double) sum / count;
System.out.println("Sum: " + sum);
System.out.println("Average: " + average);
scanner.close();
}
}
```

```

## Additional Resources

Java Program to Take 10 Integers From the User, Calculate Their Sum and Average, and Print

Creating a Java program that interacts with users to input data, process it, and then display meaningful results is a fundamental exercise for novice and experienced programmers alike. This particular task—reading ten integers, computing their sum and average, and printing the outcomes—serves as a practical introduction to user input handling, data processing, and output formatting in Java.

In this comprehensive review, we will explore every facet of writing such a program, from understanding the core requirements and setting up the development environment to implementing the code, handling edge cases, and best practices. Our goal is to provide you with a thorough understanding so that you can confidently write, modify, and extend similar programs in your coding journey.

---

## Understanding the Core Requirements

Before diving into coding, it's crucial to comprehend what the program aims to accomplish. The key tasks are:

- Input: Read ten integers from the user.
- Processing:
- Calculate the sum of these integers.
- Calculate the average of these integers.
- Output: Display the sum and average in a clear, formatted manner.

This straightforward task encompasses core programming concepts such as user input, loops, data aggregation, and output formatting.

---

## Setting Up the Development Environment

To write and run the Java program, you should have:

- Java Development Kit (JDK): Installed on your system. The latest Long-Term Support (LTS) version is recommended.
- IDE or Text Editor: Such as IntelliJ IDEA, Eclipse, NetBeans, or simple editors like VSCode or Notepad++.
- Command Line Interface (CLI): For compiling and running Java programs if not using an IDE.

Once set up, you can create a new Java file, for example, `SumAndAverage.java`, where you'll write your code.

---

## Designing the Program Structure

Effective program design involves planning the sequence of operations and how data flows through the program.
```

Key Components:

1. Input Handling: Using `Scanner` class to read user input.
2. Loop for Data Collection: A `for` loop to iterate ten times for input collection.
3. Data Storage: An array or list to store the integers.
4. Calculations:
- Summing all integers.
- Computing the average.
5. Output Formatting: Presenting results with appropriate labels and decimal points.

Sample Program Skeleton:

```java
import java.util.Scanner;

public class SumAndAverage {
public static void main(String[] args) {
// Initialize Scanner for input
Scanner scanner = new Scanner(System.in);

// Declare variables
int[] numbers = new int[10];
int sum = 0;
double average;

// Collect user input
for (int i = 0; i < 10; i++) {
System.out.print("Enter integer " + (i + 1) + ": ");
// Read input and validate
int num = scanner.nextInt();
numbers[i] = num;
sum += num;
}

// Calculate average
average = (double) sum / 10;

// Display results
System.out.println("Sum of the entered integers: " + sum);
System.out.printf("Average of the entered integers: %.2f\n", average);

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

---

# Deep Dive into Each Aspect of Implementation

## User Input Handling with Scanner

Java's `Scanner` class resides in the `java.util` package and provides methods to parse primitive types and strings from various input sources, including standard input (keyboard).

Key Points:

- To read integers, use `nextInt()`.
- Always check for input validity to avoid exceptions.
- Close the `Scanner` object after use to free resources.

Sample Input Handling with Validation:

```java
for (int i = 0; i < 10; i++) {
while (true) {
System.out.print("Enter integer " + (i + 1) + ": ");
if (scanner.hasNextInt()) {
int num = scanner.nextInt();
numbers[i] = num;
sum += num;
break;
} else {
System.out.println("Invalid input. Please enter a valid integer.");
scanner.next(); // Consume the invalid input
}
}
}
```

This approach ensures the program is robust against invalid inputs, prompting the user again if a non-integer is entered.

---

## Looping for Data Collection

Using a `for` loop simplifies iterating exactly ten times. During each iteration:

- Prompt the user.
- Validate and read input.
- Store the number.
- Update the sum.

This structure makes the code concise and reduces the chance of errors.

## Storing Data: Arrays

Arrays are ideal for fixed-size collections:

```java
int[] numbers = new int[10];
```

- They allow storing all inputs if future processing (like sorting or further analysis) is needed.
- For this task, storing is optional—summing on the fly during input collection is sufficient, but storing gives flexibility.

## Calculations: Sum and Average

- Sum: Accumulated during input collection to optimize performance.
- Average: Computed after data collection as:

```java
average = (double) sum / 10;
```

Casting to `double` ensures floating-point division for precise average calculation.

## Formatting Output

Using `System.out.println()` for basic output, and `System.out.printf()` for formatted output, especially for floating-point numbers.

Formatting Example:

```java
System.out.printf("Average: %.2f\n", average);
```

This displays the average rounded to two decimal places, which is standard for numerical data presentation.

---

## Handling Edge Cases and Validation

While the primary logic is straightforward, consider the following:

- Invalid Inputs: Non-integer inputs should be handled gracefully.
- Empty Inputs: Not applicable here since user must provide 10 integers.
- Large Numbers: Java's `int` can handle values up to approximately 2 billion. For larger numbers, consider `long`.
- Division by Zero: Not a concern here as the divisor is fixed at 10.

Sample Validation Snippet:

```java
while (true) {
System.out.print("Enter integer " + (i + 1) + ": ");
if (scanner.hasNextInt()) {
int num = scanner.nextInt();
numbers[i] = num;
sum += num;
break;
} else {
System.out.println("Invalid input. Please enter a valid integer.");
scanner.next(); // discard invalid input
}
}
```

---

## Enhancements and Best Practices

Once the basic program is working, consider the following enhancements:

- Allow Dynamic Input Size: Modify the program to accept `N` integers where `N` can be user-defined.
- Input Validation for Range: Restrict inputs within specific ranges if necessary.
- User-Friendly Output: Display the list of entered numbers alongside sum and average.
- Exception Handling: Use `try-catch` blocks to manage unexpected errors gracefully.
- Modular Code: Break the logic into separate methods for input collection, calculation, and output for better maintainability.

Example of Modular Approach:

```java
public static int[] collectInputs(int size, Scanner scanner) { ... }
public static int calculateSum(int[] numbers) { ... }
public static double calculateAverage(int sum, int count) { ... }
public static void displayResults(int sum, double average) { ... }
```

---

## Sample Complete Program with Validation and Formatting

```java
import java.util.Scanner;

public class SumAndAverage {
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
int[] numbers = new int[10];
int sum = 0;

// Input collection with validation
for (int i = 0; i < 10; i++) {
while (true) {
System.out.print("Enter integer " + (i + 1) + ": ");
if (scanner.hasNextInt()) {
int num = scanner.nextInt();
numbers[i] = num;
sum += num;
break;
} else {
System.out.println("Invalid input. Please enter a valid integer.");
scanner.next(); // discard invalid input
}
}
}

// Calculate average
double average = (double) sum / 10;

// Print results
System.out.println("\nResults:");
System.out.println("----------");
System.out.println("Entered numbers: ");
```

```
for (int num : numbers) {
System.out.print(num + " ");
}
System.out.println("\nSum of the entered integers: " + sum);
System.out.printf("Average of the entered integers: %.2f\n", average);

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

This program is user-friendly, robust, and adheres to good coding principles.

---

## Testing and Validation

Before deploying or submitting your code, thorough testing is essential:

- Test with typical inputs: Random integers within the expected

# Write A Java Program That Takes 10 Integers From The User Calculate Their Sum And Average And Print

Write A Java Program That Takes 10 Integers From The User Calculate Their Sum And Average And Print

## Related Articles


- What Do The Three Sisters Mean When They Say, "You Will Always Be Esperanza. You Will Always Be Mango
- You Are The HR Manager In A Company In London, England. You Have Just Appointed A New Project Manager
- Two Complementary Angles Are In The Ratio 3: 2. The Number Of Degrees In The Smaller Angle IsA. 18 B.

Back to Home