Finding Prime Numbers in a Given Interval Using Java

K. Naga Vihari

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

Abstract
Prime numbers, which are natural numbers larger than one that have no divisors other than themselves, have long piqued the interest of mathematicians and programmers alike. This article describes Java software that finds and displays prime integers inside a user-defined interval. The software achieves simplicity and efficiency by utilizing the modulus operator and streamlining the prime-checking method to investigate divisibility only up to a number's square root. This strategy considerably decreases temporal complexity, allowing the software to handle greater periods easily. The essay also discusses the practical uses of prime numbers in fields such as cryptography, hashing algorithms, and digital signatures, emphasizing the significance of knowing prime number identification in both theoretical and applied settings.

Keywords: Performance Optimization, Prime Numbers, Java Programming.

Introduction
Prime numbers have interested mathematicians and programmers for decades. A prime number is any natural number higher than one that has no positive divisors other than one and itself. Checking for prime integers inside a defined interval is a typical programming challenge. This tutorial will walk you through a Java application for identifying and displaying prime integers inside a user-defined interval.

Prime Numbers: A Quick Overview
A prime number is a number larger than one that cannot be divided equally by any number other than one or itself. Examples are two, three, five, seven, and eleven. Finding prime numbers requires rapidly determining if a number is divisible by any integer other than 1 and itself. In this post, we'll look at a basic yet efficient method for detecting prime integers in Java.

The Java Program: Prime Numbers in an Interval
Below is a Java program that reads an interval from the user and then prints out all the prime numbers within that interval.

Java Code Implementation
java
Copy code

```
importjava.util.Scanner;

public class PrimeNumbersInInterval {
public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        // Read the interval from the user
System.out.print("Enter the starting value of the interval: ");
int start = scanner.nextInt();

System.out.print("Enter the ending value of the interval: ");
int end = scanner.nextInt();

System.out.println("Prime numbers between " + start + " and " + end + ":");

        // Find and print prime numbers in the given interval
for (inti = start; i<= end; i++) {
if (isPrime(i)) {
System.out.print(i + " ");
            }
        }

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

    // Method to check if a number is prime
public static booleanisPrime(int number) {
        // Return false for numbers less than 2 (1 and below are not prime)
if (number <= 1) {
return false;
        }

        // Check for factors from 2 up to the square root of the number
for (inti = 2; i<= Math.sqrt(number); i++) {
if (number % i == 0) {
return false; // Found a factor, so it's not prime
            }
        }

        // If no factors were found, the number is prime
return true;
    }
}
```

Explanation of the Code

1. Reading User Input: The application prompts users to input the interval's start and end values. These values specify the range in which we shall look for prime numbers.
2. Prime Number Identification: The Prime(int number) function determines if an integer is prime. It applies the following logic: Numbers less than or equal to one are not prime. The software checks for

divisibility of values ranging from 2 to the square root. This is an optimization: if a number has a factor bigger than its square root, it must also have a factor less than the square root.

3. Displaying Prime Numbers: The program iterates across the provided interval, using the isPrime() function on each integer. If the number is prime, it is printed on the console.

Sample Output

Here is a sample run of the program:

Copy code

```
Enter the starting value of the interval: 4
Enter the ending value of the interval: 50
Prime numbers between 4 and 50:
5 7 11 13 17 19 23 29 31 37 41 43 47
```

The biggest advantage is from restricting the check for factors to the square root of the integer. This greatly minimizes the temporal complexity, especially for longer periods. This software checks numbers from 2 to √n, which is substantially quicker than checking all numbers from 2 to n-1.

Practical Applications of Prime Numbers

Prime numbers have several practical uses, most notably in cryptography, where they are utilized in encryption methods such as RSA. They are also utilized in hashing techniques, random number creation, and digital signatures.

Conclusion

This article shows how to use Java to discover prime integers inside a user-defined interval. The software is simple to comprehend, performance-optimized, and serves as a strong basis for anyone learning about prime number techniques. Whether you're new to programming or just brushing up on the fundamentals, this exercise will help you improve your coding abilities.

References

1. **Baeldung:** Prime Numbers in Java
2. **GeeksforGeeks:** Prime Number Program in Java
3. **JavaTPoint:** Java Program to Check Prime Numbers