**SKILL DEVELOPMENT**

**Project Report**

**DLithe Consultancy Services Pvt. Ltd.**



**Project Report Assessment**

**Student Name: Bhoomi M Kokatnur**

**USN: 2JR23CS026**

**Assignment: Java**

**Organization:** DLithe Consultancy Services Pvt. Ltd.

**Supervisor’s Name: Archana SM**

## **Submitted to**

Signature of Training Supervisor Signature of Students

Date: Date:

**Table of Contents**

[Introduction 5](#_gjdgxs)

[Background. 6](https://docs.google.com/document/d/1dlZ-Wp1bwbwgrIhnEjPS92jJrV075AvT/edit" \l "heading=h.1fob9te)

[Use Case 6](#_3znysh7)

[Training Experience 9](#_2et92p0)

[Key Learnings 10](#_tyjcwt)

[Challenges 11](#_3dy6vkm)

Applications 11

[Conclusion](#_1t3h5sf) 12

1. **Introduction**

The **Prime Number Checker Program** is designed to determine whether a given number is a **prime number** or not. Prime numbers are natural numbers greater than 1 that are divisible only by 1 and themselves. This program takes an input number from the user and uses a logical algorithm to check its primality. Prime number checking is a fundamental concept in mathematics and computer science, with applications in cryptography, data security, and optimization algorithms. The program provides a simple and efficient way to verify primality, making it an excellent example of applying conditional statements and loops in programming.

This program accepts an input from the user and then checks its divisibility by all numbers less than itself (usually up to the square root of the number, for efficiency). If the number is divisible by any other number apart from 1 and itself, it is classified as a **composite number**. Otherwise, it is a prime number.

**2. Background**

Prime numbers have fascinated mathematicians for centuries due to their unique properties and fundamental role in number theory. A **prime number** is defined as a natural number greater than 1 that is divisible only by 1 and itself. The sequence of prime numbers begins with 2, 3, 5, 7, 11, and continues infinitely. Their unpredictability and mathematical significance have made them essential in various fields, including **cryptography**, **computer science**, and **digital security**.

The concept of checking whether a number is prime or not is one of the earliest problems introduced in computational logic. This problem is simple to understand but can become computationally intensive when working with very large numbers, which is especially important in applications such as encryption algorithms and secure communication protocols (e.g., RSA).

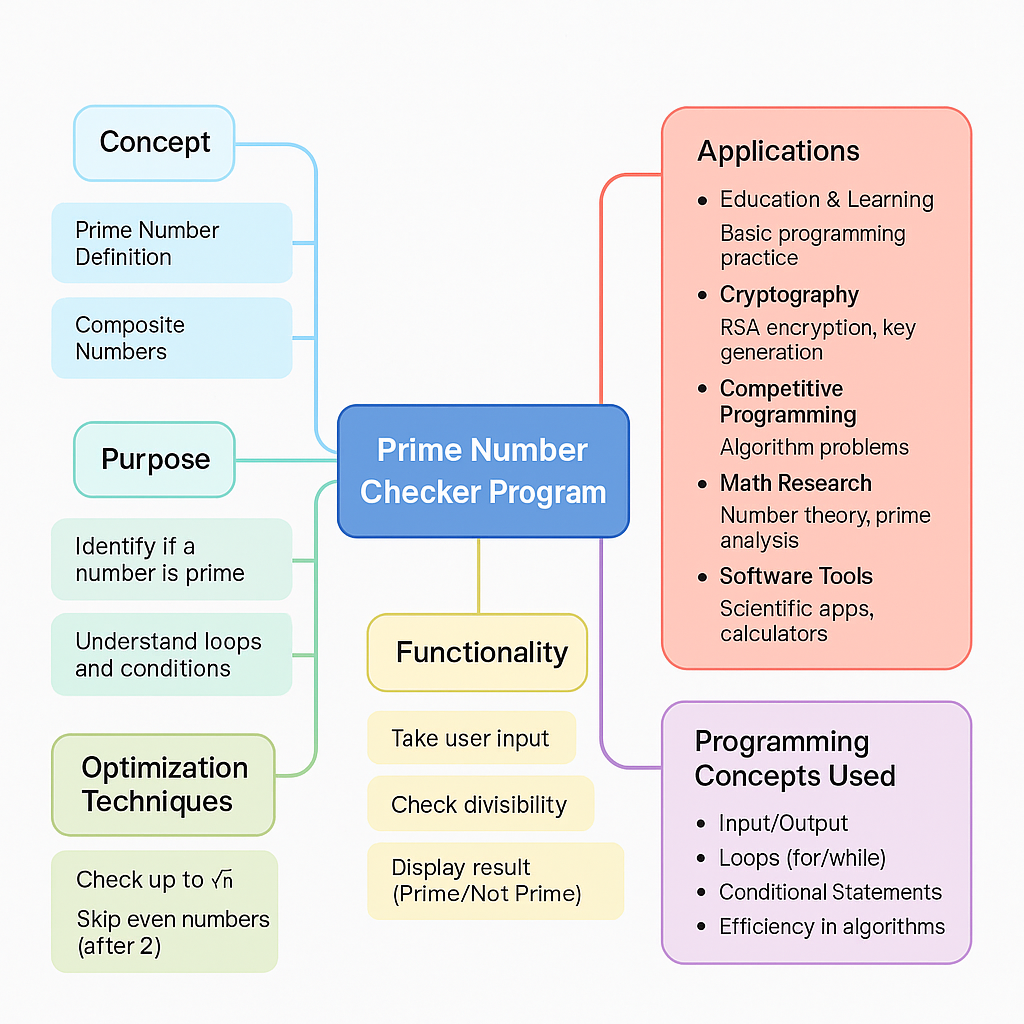
With the increasing use of digital systems, the need for fast and accurate primality testing has become more critical. The **Prime Number Checker Program** aims to provide a solution to this need by implementing an algorithm that efficiently checks whether a given number is prime. While basic methods involve checking divisibility up to the number itself, optimized approaches reduce this to checking up to the square root of the number, significantly improving performance.

**3. Use Case**

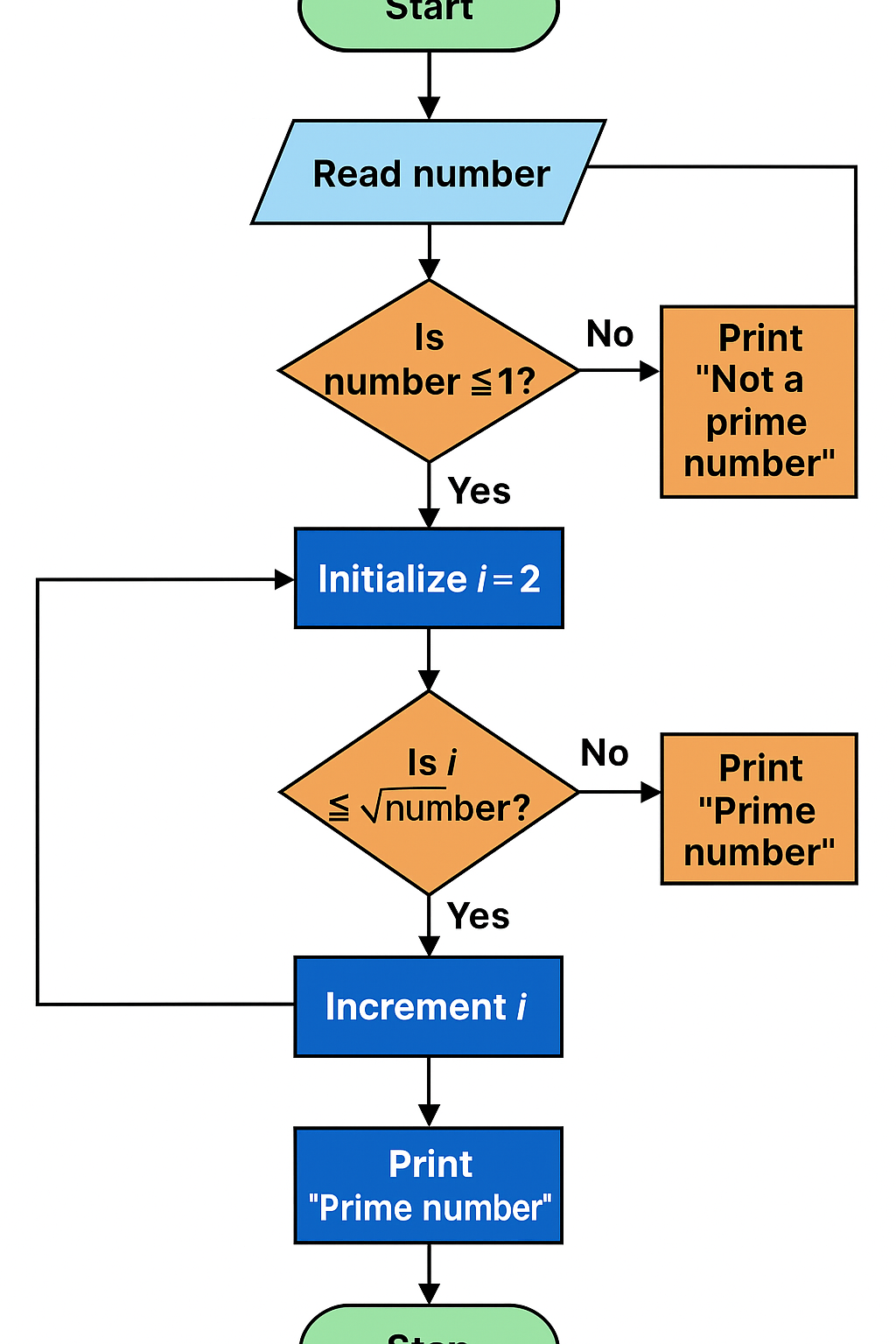
The **Prime Number Checker Program** serves a variety of practical and educational purposes. In the field of education, it is a fundamental tool for beginners to understand core programming concepts such as loops, conditionals, and logic building. It also helps students grasp basic mathematical ideas like divisibility and prime numbers. Beyond education, this program has significant applications in **cryptography** and **cybersecurity**, where large prime numbers are used in encryption algorithms like RSA to secure data and communications. The checker can be adapted to verify large numbers, making it useful in key generation and secure systems.

In mathematical research and number theory, this program is used to test and analyze prime-related properties and can support advanced algorithms. It is also widely used in **competitive programming** and **technical interviews**, where prime-checking problems often appear in various forms. Furthermore, it can be integrated into larger **software applications** such as scientific calculators, educational platforms, or simulation tools that require number classification. Lastly, in areas like **random number generation**, prime numbers help ensure better distribution and security, and this program plays a role in validating such numbers. Overall, the Prime Number Checker is a versatile and foundational tool with wide-ranging use cases.

**MIND MAP**

****

**FLOWCHART**

****

## **Project Development Images**



Technologies used:

**JAVA**

****

**4. Training Experience**

Learning to create a prime number checker in Java is a great exercise for understanding fundamental programming concepts. It begins with grasping the definition of a prime number—a number greater than 1 that has no divisors other than 1 and itself. The coding process involves taking user input using Java’s Scanner class, which helps practice input handling. Then, using conditional statements, the program handles special cases like numbers less than or equal to one, which are not prime. The core logic uses a loop to test whether the input number can be divided evenly by any number between 2 and the square root of the number—this optimization reduces unnecessary checks and improves efficiency. A boolean flag keeps track of whether the number remains prime throughout the checks. Finally, the program outputs the result, reinforcing the understanding of control flow and output formatting in Java. Through this exercise, learners not only practice loops, conditionals, and input/output operations but also develop problem-solving skills and gain insight into optimizing algorithms. It lays a solid foundation for more advanced programming challenges, such as checking primes in ranges or implementing more efficient algorithms like the Sieve of Eratosthenes.

**5. Key Learnings**

· **Understanding the concept of prime numbers:** Knowing what makes a number prime (only divisible by 1 and itself).

· · **Taking user input:** Using the Scanner class to read input from the console.

· · **Conditional statements:** Applying if-else to handle special cases (numbers ≤ 1 are not prime).

· · **Looping and iteration:** Using a for loop to check divisibility by numbers from 2 up to the square root of the input number.

· · **Optimization technique:** Checking divisors only up to √number to reduce unnecessary computations.

· · **Using boolean flags:** Tracking the primality status efficiently during the checks.

· · **Breaking loops early:** Stopping the loop when a divisor is found to improve performance.

· · **Output formatting:** Displaying clear and meaningful results to the user.

· · **Basic algorithm design:** Combining input, processing, and output steps logically.

**6. Challenges**

When learning to write a prime number checker in Java, beginners often face several challenges. Initially, understanding the mathematical concept of prime numbers—especially why numbers less than or equal to one are not considered prime—can be confusing. Another common difficulty is optimizing the code to check divisibility only up to the square root of the number, which requires both mathematical insight and programming skill. Handling special cases such as zero, one, and negative inputs correctly also poses a challenge, as does controlling loops efficiently by breaking early when a divisor is found. Additionally, beginners might struggle with validating user input to prevent errors or unexpected behavior. For very large numbers, the basic checking method may be too slow, highlighting the need to understand algorithmic efficiency.

**7. Applications**

The prime number checker program, while simple, has important applications in various fields of computer science and mathematics. It forms the foundation for understanding more complex algorithms used in cryptography, where prime numbers play a crucial role in securing digital communication through encryption techniques like RSA. Additionally, prime checking is essential in number theory research and is often used in generating prime numbers for hashing functions, random number generation, and error detection algorithms. Learning to implement a prime checker also helps build problem-solving skills and logical thinking, which are valuable in software development and algorithm design. Beyond academia, prime numbers and their identification are vital in blockchain technology, cybersecurity, and digital signatures, making this basic program a stepping stone to understanding these advanced applications.

1. **Conclusion**

In conclusion, writing a prime number checker in Java is a valuable learning exercise that helps develop fundamental programming skills such as input handling, loops, conditionals, and algorithm optimization. It also introduces important mathematical concepts and lays the groundwork for understanding more complex applications in cryptography and computer science. Mastering this simple program builds a strong foundation for tackling advanced coding challenges in the future.