

## **EXPERIMENT - 1**

### **AIM:**

To write a Java program that swaps the values of two numbers without using a third variable, using the Scanner class for input.

### **ALGORITHM:**

**Step 1:** Start the process.

**Step 2:** Open the Eclipse IDE.

**Step 3:** Declare two integer variables a and b and use the Scanner class to assign values to a and b from user input.

**Step 4:** Print the original values of a and b.

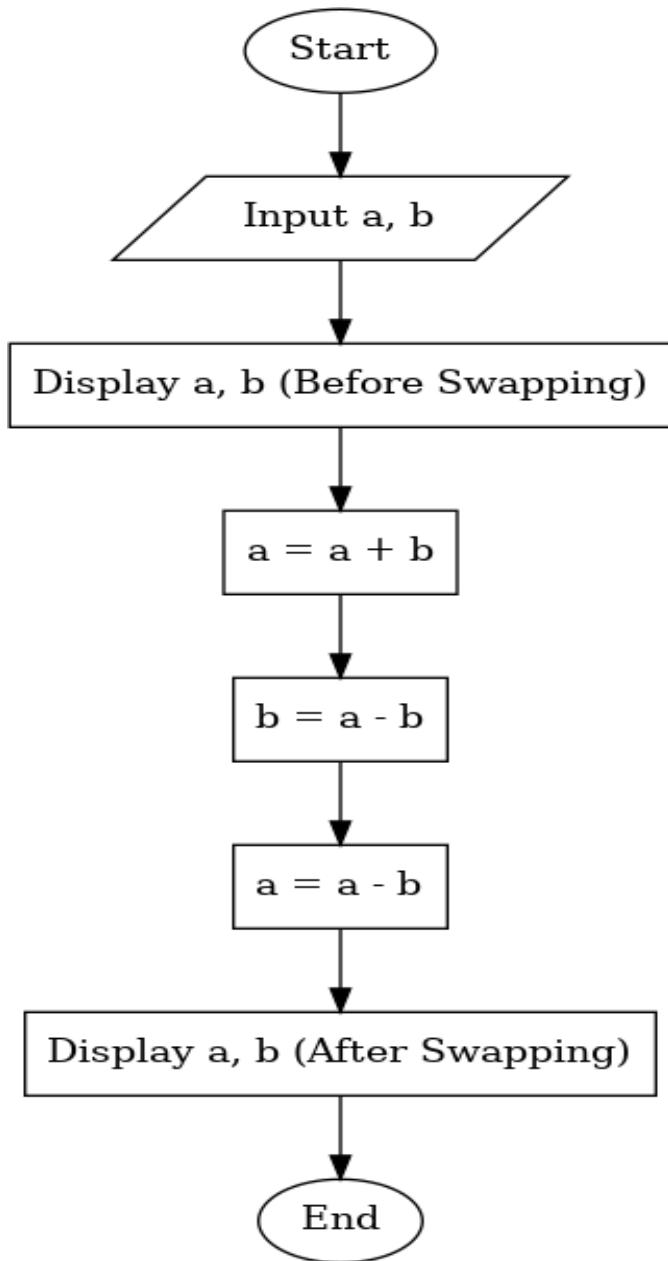
**Step 5:** Swap the values of a and b without using a third variable:

- Set  $a = a + b$ , then set  $b = a - b$ , and finally set  $a = a - b$ .

**Step 6:** Print the swapped values of a and b.

**Step 7:** End the process.

## **FLOW CHART:**



## **SOURCE CODE:**

```

import java.util.Scanner;
public class SwapNumbers {
    public static void main(String[] args) {
        int a, b;
  
```

```

Scanner scanner = new Scanner(System.in);
System.out.print("Enter first number (a): ");
a = scanner.nextInt();
System.out.print("Enter second number (b): ");
b = scanner.nextInt();
System.out.println("\nBefore Swapping:");
System.out.println("a = " + a);
System.out.println("b = " + b);
a = a + b;
b = a - b;
a = a - b;
System.out.println("\nAfter Swapping:");
System.out.println("a = " + a);
System.out.println("b = " + b);
scanner.close();
}
}

```

## **CODE EXPLANATION:**

1. **import java.util.Scanner;**
  - o Imports the **Scanner** class, which is used to read user input from the console.
2. **public class SwapNumbers {**
  - o Defines the class **SwapNumbers**, which contains the program logic.
3. **public static void main(String[] args) {**
  - o The entry point of the program. This method is executed when the program runs.
4. **int a, b;**
  - o Declares two integer variables **a** and **b** to store the numbers.
5. **Scanner scanner = new Scanner(System.in);**
  - o Creates a **Scanner** object to read input from the console.
6. **System.out.print("Enter first number (a): ");**
  - o Prompts the user to enter the first number.

7. **a = scanner.nextInt();**
  - o Reads an integer from the user and assigns it to the variable **a**.
8. **System.out.print("Enter second number (b): ");**
  - o Prompts the user to enter the second number.
9. **b = scanner.nextInt();**
  - o Reads another integer from the user and assigns it to the variable **b**.
10. **System.out.println("\nBefore Swapping:");**
  - o Prints a message indicating the start of the "before swapping" section.
11. **System.out.println("a = " + a);**
  - o Displays the value of **a** before swapping.
12. **System.out.println("b = " + b);**
  - o Displays the value of **b** before swapping.
13. **a = a + b;**
  - o Adds the values of **a** and **b** and stores the result in **a**. At this point, **a** holds the sum of the two numbers.
14. **b = a - b;**
  - o Subtracts **b** (the original value) from **a** (the sum of **a** and **b**). This operation effectively assigns the original value of **a** to **b**.
15. **a = a - b;**
  - o Subtracts the new **b** (which now holds the original value of **a**) from **a** (the sum of the original **a** and **b**). This operation assigns the original value of **b** to **a**.
16. **System.out.println("\nAfter Swapping:");**
  - o Prints a message indicating the start of the "after swapping" section.
17. **System.out.println("a = " + a);**
  - o Displays the value of **a** after swapping.
18. **System.out.println("b = " + b);**
  - o Displays the value of **b** after swapping.
19. **scanner.close();**
  - o Closes the **Scanner** object to release system resources. It is good practice to close the scanner after use.

**OUTPUT:**

```
Enter first number (a): 13
Enter second number (b): 15
```

```
Before Swapping:
```

```
a = 13
b = 15
```

```
After Swapping:
```

```
a = 15
b = 13
```

**RESULT:**

Thus, the program had been successfully executed.

## **EXPERIMENT - 2**

### **AIM:**

Write a Java program that prompts the user for an integer and displays all prime numbers up to that integer.

### **ALGORITHM:**

**Step 1:** Start the process.

**Step 2:** Open the Eclipse IDE.

**Step 3:** Create a Scanner object to read input from the user.

**Step 4:** Prompt the user to enter an integer.

**Step 5:** Store the input integer in a variable n.

**Step 6:** Print a message indicating the prime numbers up to n.

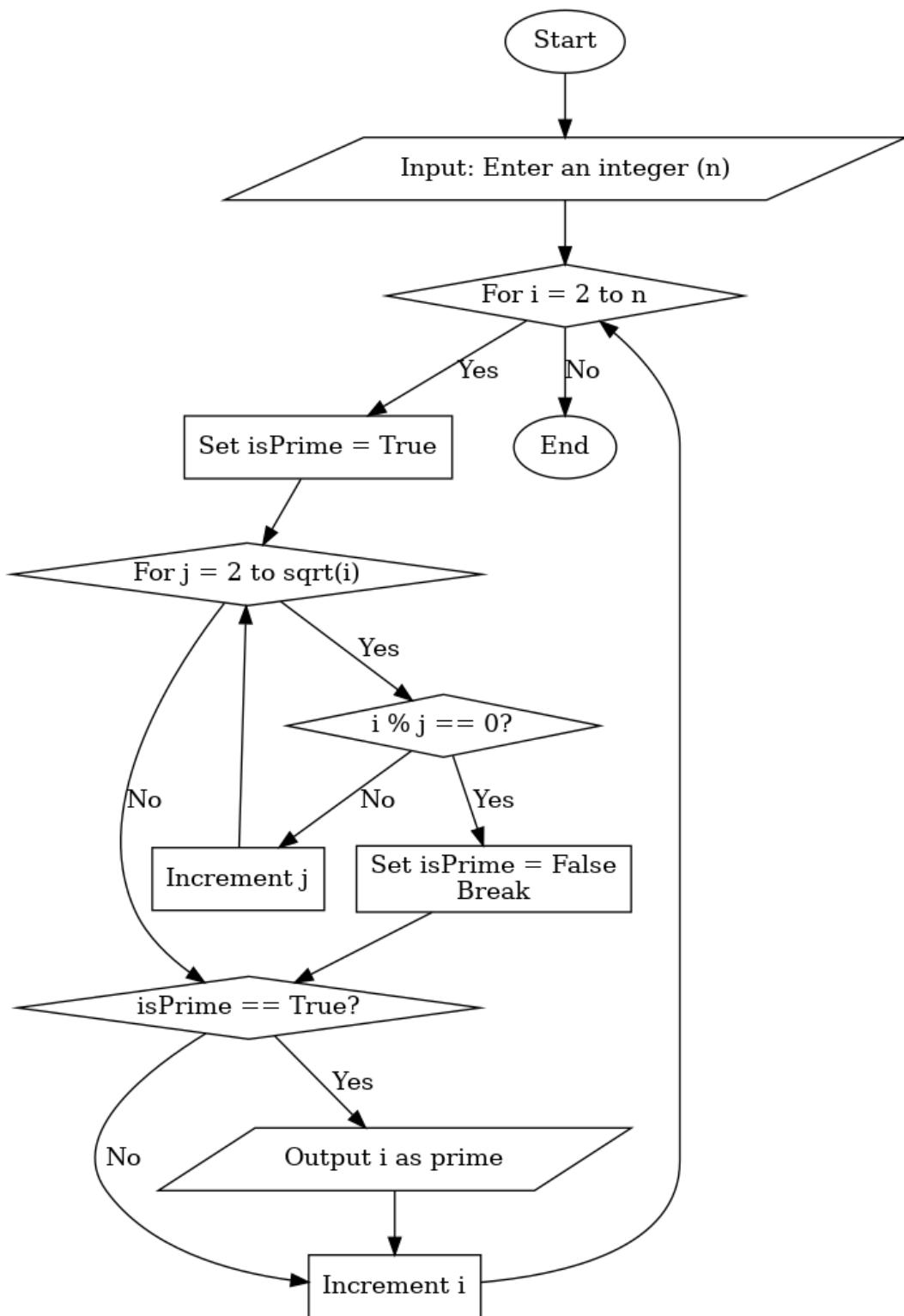
**Step 7:** For each number i from 2 to n:

- Now Initialize a flag variable isPrime as true. For each number j from 2 to the square root of I, If i is divisible by j, set isPrime to false and break the loop.

**Step 8:** If isPrime is still true, print i as a prime number.

**Step 9:** End the process.

## **FLOW CHART:**



## **SOURCE CODE:**

```
public class PrimeNumbers {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter an integer: ");
        int n = scanner.nextInt();
        System.out.println("Prime numbers up to " + n + " are:");
        for (int i = 2; i <= n; i++) {
            boolean isPrime = true;
            for (int j = 2; j * j <= i; j++) {
                if (i % j == 0) {
                    isPrime = false; // i is not prime
                    break; // exit the inner loop
                }
            }
            if (isPrime) {
                System.out.print(i + " ");
            }
        }
        scanner.close();
    }
}
```

## **CODE EXPLANATION:**

1. **public class PrimeNumbers {**
  - o Declares a class named **PrimeNumbers**, which contains the logic to find prime numbers.
2. **public static void main(String[] args) {**
  - o The entry point of the program. This method is executed when the program runs.
3. **Scanner scanner = new Scanner(System.in);**
  - o Creates a **Scanner** object to read input from the user.

4. **System.out.print("Enter an integer: ");**
  - Prompts the user to enter an integer value.
5. **int n = scanner.nextInt();**
  - Reads the integer entered by the user and stores it in the variable **n**.
6. **System.out.println("Prime numbers up to " + n + " are:");**
  - Prints a message indicating the program will display prime numbers up to **n**.
7. **for (int i = 2; i <= n; i++) {**
  - Starts a loop to check all numbers from **2** to **n**. The variable **i** represents the current number being checked.
8. **boolean isPrime = true;**
  - Assumes initially that the current number **i** is prime. This variable will be updated if the number is found to be non-prime.
9. **for (int j = 2; j \* j <= i; j++) {**
  - Starts a loop to check divisors for the current number **i**.
  - **j** starts at **2** and runs up to the square root of **i** (**j \* j <= i**) to optimize the check.
10. **if (i % j == 0) {**
  - Checks if **i** is divisible by **j**. If true, **i** is not a prime number.
11. **isPrime = false;**
  - Sets **isPrime** to **false**, indicating that **i** is not a prime number.
12. **break;**
  - Exits the inner loop early since **i** is confirmed not to be prime.
13. **}**
  - Ends the inner **for** loop.
14. **if (isPrime) {**
  - Checks if **isPrime** is still **true** after the inner loop. If so, **i** is a prime number.
15. **System.out.print(i + " ");**
  - Prints the current prime number **i**, followed by a space.
16. **}**
  - Ends the outer **for** loop.
17. **scanner.close();**
  - Closes the **Scanner** object to release system resources.

18. }

- Ends the `main` method.

19. }

- Ends the `PrimeNumbers` class.

### **OUTPUT:**

```
Enter an integer: 20
Prime numbers up to 20 are:
2 3 5 7 11 13 17 19
```

### **RESULT:**

Thus, the program had been successfully executed.

## **EXPERIMENT - 3**

### **AIM:**

Write a Java program that multiplies two matrices, taking input from the user, and then calculates and prints the product of the matrices.

### **ALGORITHM:**

**Step 1:** Start the process.

**Step 2:** Open the Eclipse IDE.

**Step 3:** Input the number of rows and columns for both matrices (matrix A and matrix B) from the user.

**Step 4:** Check if matrix multiplication is possible:

- Ensure that the number of columns of matrix A is equal to the number of rows of matrix B.
- If the condition is not satisfied, print an error message and terminate the process.

**Step 5:** Input the elements of matrix A from the user.

**Step 6:** Input the elements of matrix B from the user.

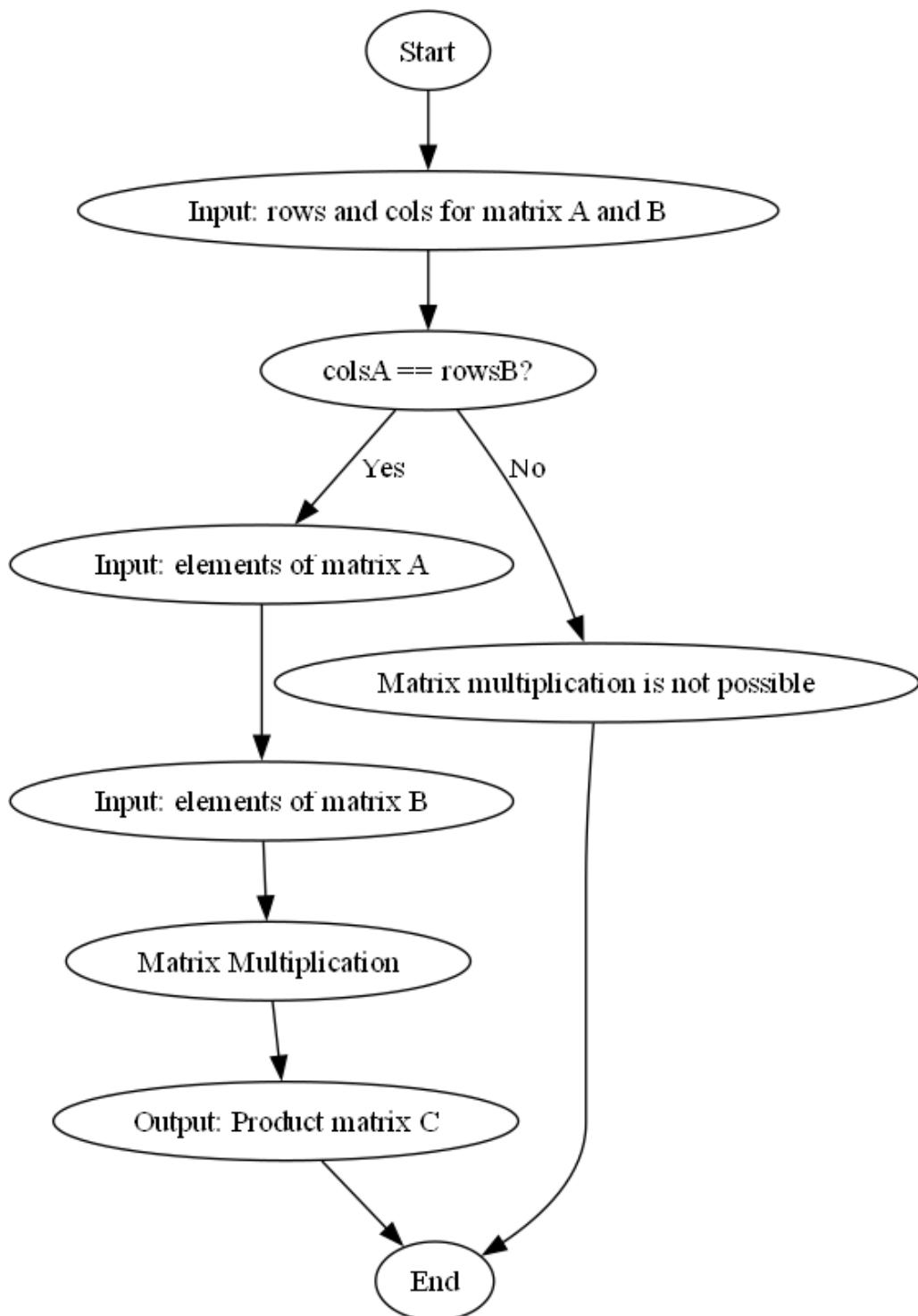
**Step 7:** Perform matrix multiplication:

- Multiply each row element of matrix A by each column element of matrix B.
- For each element of the result matrix, calculate the sum of the products.

**Step 8:** Store the result of the matrix multiplication in a new matrix, and print the resultant matrix.

**Step 9:** End the process.

## **FLOW CHART:**



## **SOURCE CODE:**

```

import java.util.Scanner;

public class MatrixMultiplication {

    public static void main(String[] args) {
        Scanner myobj = new Scanner(System.in);
        System.out.print("Enter the number of rows for matrix A: ");
        int rowsA = myobj.nextInt();
        System.out.print("Enter the number of columns for matrix A: ");
        int colsA = myobj.nextInt();
        System.out.print("Enter the number of rows for matrix B: ");
        int rowsB = myobj.nextInt();
        System.out.print("Enter the number of columns for matrix B: ");
        int colsB = myobj.nextInt();
        if (colsA != rowsB) {
            System.out.println("Matrix multiplication is not possible.");
            return;
        }
        int[][] a = new int[rowsA][colsA];
        int[][] b = new int[rowsB][colsB];
        int[][] c = new int[rowsA][colsB];
        System.out.println("Enter elements of matrix A:");
        for (int i = 0; i < rowsA; i++) {
            for (int j = 0; j < colsA; j++) {
                a[i][j] = myobj.nextInt();
            }
        }
        System.out.println("Enter elements of matrix B:");
        for (int i = 0; i < rowsB; i++) {
    
```

```

for (int j = 0; j < colsB; j++)
{
    b[i][j] = myobj.nextInt();
}
}

for (int i = 0; i < rowsA; i++)
{
    for (int j = 0; j < colsB; j++)
    {
        for (int k = 0; k < colsA; k++)
        {
            c[i][j] += a[i][k] * b[k][j];
        }
    }
}

System.out.println("The product of the matrices is:");
for (int i = 0; i < rowsA; i++)
{
    for (int j = 0; j < colsB; j++)
    {
        System.out.print(c[i][j] + " ");
    }
    System.out.println();
}
myobj.close();
}
}

```

## **CODE EXPLANATION:**

1. **import java.util.Scanner;**
  - o Imports the `Scanner` class for user input.
2. **public class MatrixMultiplication {**
  - o Declares a class named `MatrixMultiplication`.
3. **public static void main(String[] args) {**
  - o The main method where the program execution begins.
4. **Scanner myobj = new Scanner(System.in);**
  - o Creates a `Scanner` object to read input from the user.
5. **System.out.print("Enter the number of rows for matrix A: ");**
  - o Prompts the user to enter the number of rows for matrix A.
6. **int rowsA = myobj.nextInt();**
  - o Reads the number of rows for matrix A and stores it in `rowsA`.
7. **System.out.print("Enter the number of columns for matrix A: ");**
  - o Prompts the user to enter the number of columns for matrix A.
8. **int colsA = myobj.nextInt();**
  - o Reads the number of columns for matrix A and stores it in `colsA`.
9. **System.out.print("Enter the number of rows for matrix B: ");**
  - o Prompts the user to enter the number of rows for matrix B.
10. **int rowsB = myobj.nextInt();**
  - o Reads the number of rows for matrix B and stores it in `rowsB`.
11. **System.out.print("Enter the number of columns for matrix B: ");**
  - o Prompts the user to enter the number of columns for matrix B.
12. **int colsB = myobj.nextInt();**
  - o Reads the number of columns for matrix B and stores it in `colsB`.
13. **if (colsA != rowsB) {**
  - o Checks if matrix multiplication is possible. The number of columns in matrix A must equal the number of rows in matrix B.
14. **System.out.println("Matrix multiplication is not possible.");**
  - o Prints a message if matrix multiplication is not possible.
15. **return;**

- Exits the program if multiplication cannot be performed.

16. **int[][] a = new int[rowsA][colsA];**

- Declares a 2D array to store matrix A.

17. **int[][] b = new int[rowsB][colsB];**

- Declares a 2D array to store matrix B.

18. **int[][] c = new int[rowsA][colsB];**

- Declares a 2D array to store the result of matrix multiplication.

19. **System.out.println("Enter elements of matrix A:");**

- Prompts the user to enter the elements of matrix A.

20. **for (int i = 0; i < rowsA; i++) {**

- Loops through each row of matrix A.

21. **for (int j = 0; j < colsA; j++) {**

- Loops through each column of matrix A.

22. **a[i][j] = myobj.nextInt();**

- Reads the value for matrix A at position [i][j].

23. **System.out.println("Enter elements of matrix B:");**

- Prompts the user to enter the elements of matrix B.

24. **for (int i = 0; i < rowsB; i++) {**

- Loops through each row of matrix B.

25. **for (int j = 0; j < colsB; j++) {**

- Loops through each column of matrix B.

26. **b[i][j] = myobj.nextInt();**

- Reads the value for matrix B at position [i][j].

27. **for (int i = 0; i < rowsA; i++) {**

- Loops through each row of the result matrix.

28. **for (int j = 0; j < colsB; j++) {**

- Loops through each column of the result matrix.

29. **for (int k = 0; k < colsA; k++) {**

- Loops through the elements of the row of matrix A and column of matrix B to calculate the product.

30. **c[i][j] += a[i][k] \* b[k][j];**

- Multiplies the corresponding elements and adds to the result matrix at [i][j].

31. **System.out.println("The product of the matrices is:");**

- Prints the result matrix.

32. **for (int i = 0; i < rowsA; i++) {**

- Loops through each row of the result matrix.

33. **for (int j = 0; j < colsB; j++) {**

- Loops through each column of the result matrix.

34. **System.out.print(c[i][j] + " ");**

- Prints the value of the result matrix at position [i][j].

35. **System.out.println();**

- Moves to the next line after printing one row of the result matrix.

36. **myobj.close();**

- Closes the **Scanner** to release resources.

37. **}**

- Closes the **main** method.

38. **}**

- Closes the **MatrixMultiplication** class.

## **OUTPUT:**

```
Enter the number of rows for matrix A: 3
Enter the number of columns for matrix A: 3
Enter the number of rows for matrix B: 3
Enter the number of columns for matrix B: 3
Enter elements of matrix A:
9 8 7
6 5 4
3 2 1
Enter elements of matrix B:
1 2 3
4 5 6
7 8 9
The product of the matrices is:
90 114 138
54 69 84
18 24 30
```

## **RESULT:**

Thus, the program had been successfully executed.

## **EXPERIMENT - 4**

### **AIM:**

Write a Java program that reads a text file and displays the number of characters, lines, and words in the file.

### **ALGORITHM:**

**Step 1:** Start the process.

**Step 2:** Create a Java class and import necessary classes (BufferedReader, FileReader, and IOException).

**Step 3:** Define the file path of the text file to be read.

**Step 4:** Use a BufferedReader to read the file line by line.

**Step 5:** Initialize counters for characters, lines, and words.

**Step 6:** Read each line from the file.

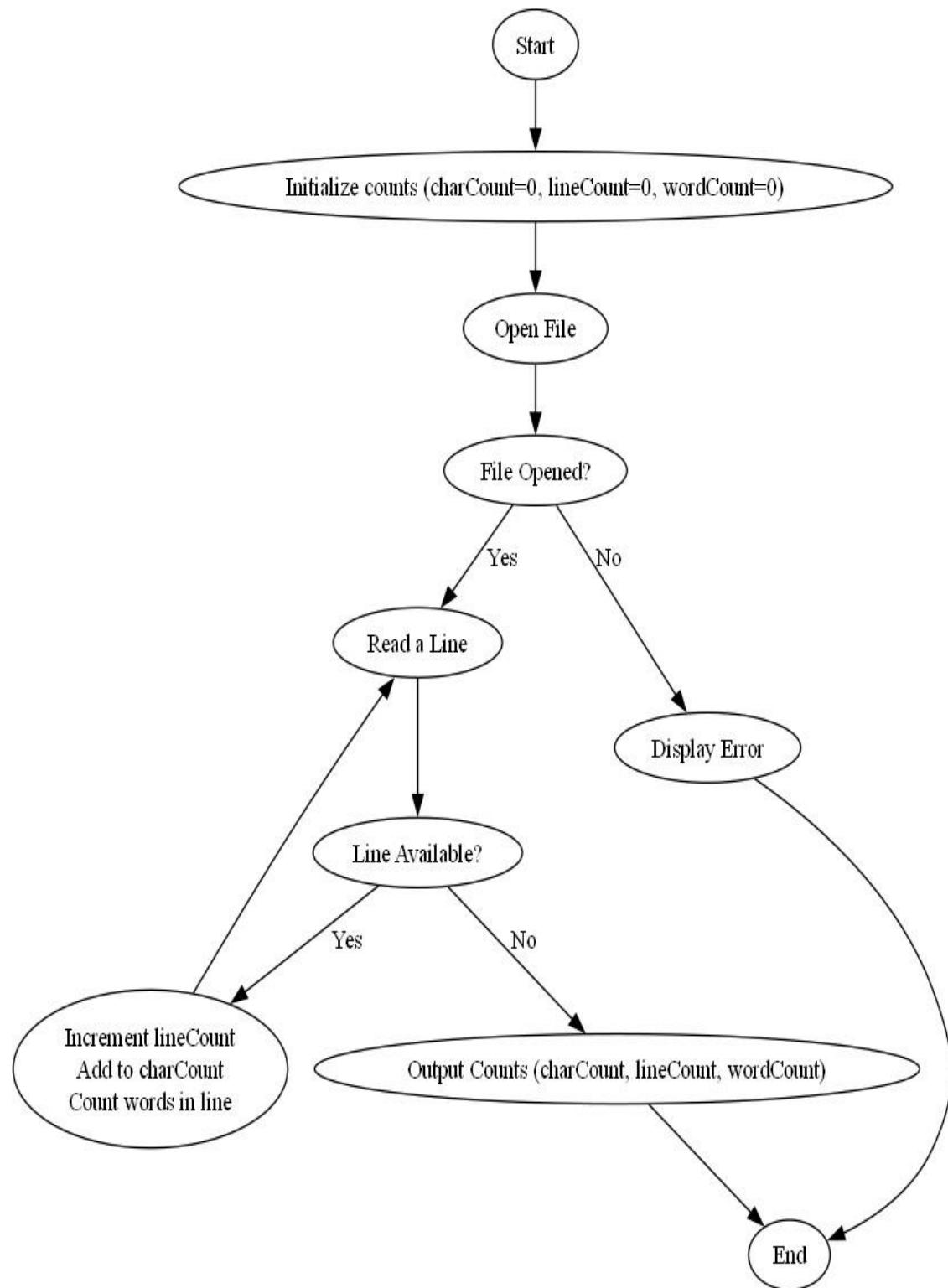
- Increment the line count for each line read.
- Increment the character count by the length of the line.
- Split the line into words and increment the word count based on the number of words.

**Step 7:** After reading the entire file, print the total number of characters, lines, and words.

**Step 8:** Handle any exceptions using a try-catch block for IOException.

**Step 9:** End the process.

## **FLOW CHART:**



## **SOURCE CODE:**

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
public class TextStatistics {
    public static void main(String[] args) {
        String filePath = "path/to/your/textfile.txt"; // Change to your file path
        try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
            String line;
            int characterCount = 0;
            int lineCount = 0;
            int wordCount = 0;
            while ((line = reader.readLine()) != null) {
                lineCount++;
                characterCount += line.length();
                wordCount += line.split("\\s+").length; // Split by whitespace
            }
            System.out.println("Characters: " + characterCount);
            System.out.println("Lines: " + lineCount);
            System.out.println("Words: " + wordCount);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

## **CODE EXPLANATION:**

```
import java.io.BufferedReader;
```

- **Purpose:** Imports the `BufferedReader` class to read text from a file efficiently, line by line.

```
import java.io.FileReader;
```

- **Purpose:** Imports the `FileReader` class, which is used to read the contents of a file.

```
import java.io.IOException;
```

- **Purpose:** Imports the `IOException` class, which is needed to handle input-output-related exceptions (e.g., file not found or read errors).

```
public class TextStatistics {
```

- **Purpose:** Declares a class named `TextStatistics`. This is the program's main class.

```
public static void main(String[] args) {
```

- **Purpose:** The `main` method is the entry point of the program where execution begins.

```
String filePath = "path/to/your/textfile.txt";
```

- **Purpose:** Defines a string variable `filePath` that stores the path to the text file to be analyzed. You need to replace "path/to/your/textfile.txt" with the actual file path.

```
try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
```

- **Purpose:**

1. Creates a `BufferedReader` object `reader` to read the file.
2. The `new FileReader(filePath)` reads the file at the specified path.
3. The `try-with-resources` ensures the `BufferedReader` is automatically closed after the operations complete, preventing resource leaks.

```
String line;
```

- **Purpose:** Declares a variable `line` to store each line of text read from the file.

```
int characterCount = 0;
```

- **Purpose:** Initializes a variable `characterCount` to 0. It will store the total number of characters in the file.

```
int lineCount = 0;
```

- **Purpose:** Initializes a variable `lineCount` to 0. It will store the total number of lines in the file.

```
int wordCount = 0;
```

- **Purpose:** Initializes a variable `wordCount` to 0. It will store the total number of words in the file.

```
while ((line = reader.readLine()) != null) {
```

- **Purpose:** Reads each line from the file until there are no more lines (`null` indicates the end of the file).
- **Explanation:**
  - `reader.readLine()` reads a single line of text from the file.
  - The `line` variable stores the current line.

```
lineCount++;
```

- **Purpose:** Increments the `lineCount` by 1 for each line read, keeping track of the total number of lines.

```
characterCount += line.length();
```

- **Purpose:** Adds the length of the current line to `characterCount`, counting all the characters (including spaces and punctuation) in the file.

```
wordCount += line.split("\\s+").length;
```

- **Purpose:** Counts the words in the current line and adds them to `wordCount`.
- **Explanation:**
  - `line.split("\\s+")` splits the line into an array of words, using one or more whitespace characters (`\s+`) as the delimiter.
  - `.length` gets the number of elements in the array, which represents the number of words.

```
}
```

- **Purpose:** Ends the `while` loop after processing all lines in the file.

```
System.out.println("Characters: " + characterCount);
```

- **Purpose:** Prints the total number of characters counted in the file.

```
System.out.println("Lines: " + lineCount);
```

- **Purpose:** Prints the total number of lines counted in the file.

```
System.out.println("Words: " + wordCount);
```

- **Purpose:** Prints the total number of words counted in the file.

```
} catch (IOException e) {
```

- **Purpose:** Catches any **IOException** that might occur during file reading (e.g., file not found or read errors).

```
e.printStackTrace();
```

- **Purpose:** Prints the stack trace of the exception to the console for debugging purposes.

```
}
```

- **Purpose:** Ends the **try-catch** block

```
}
```

- **Purpose:** Ends the **main** method and the program.

### **OUTPUT:**

```
Characters: 28
Lines: 3
Words: 6
```

### **RESULT:**

Thus, the program had been successfully executed.

## **EXPERIMENT - 5**

### **AIM:**

Generate random numbers between two given limits using Random class and print messages according to the value range generated.

### **ALGORITHM:**

**Step 1:** Start the Process.

**Step 2:** Import **Random** and **Scanner**.

**Step 3:** Create a **Scanner** and **Random** objects.

**Step 4:** Prompt for and read **lowerLimit** and **upperLimit**.

**Step 5:** Calculate **randomNumber** using

**random.nextInt(upperLimit - lowerLimit + 1) + lowerLimit.**

**Step 6:** Print the generated **randomNumber**.

**Step 7:** If **randomNumber < 0**, print "The number is negative.";

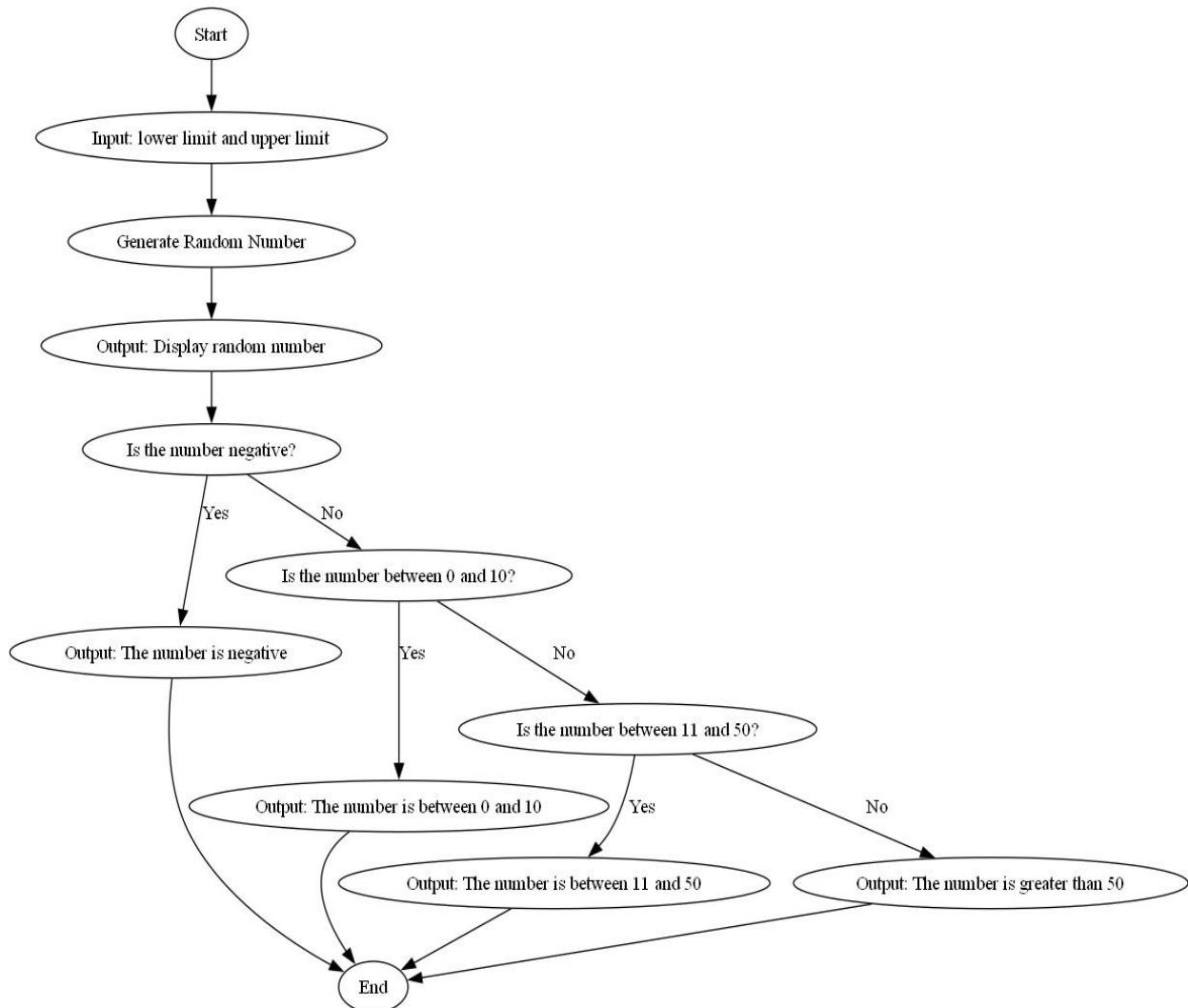
else if **0 ≤ randomNumber ≤ 10**, print "The number is between 0 and 10.";

else if **11 ≤ randomNumber ≤ 50**, print "The number is between 11 and 50.";

else print "The number is greater than 50."

**Step 8:** End the Process.

## **FLOW CHART:**



## **SOURCE CODE:**

```

import java.util.Random;
import java.util.Scanner;
public class RandomNumberGenerator {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Random random = new Random();
        // Input for range limits
        System.out.print("Enter the lower limit: ");
  
```

```

int lowerLimit = scanner.nextInt();
System.out.print("Enter the upper limit: ");
int upperLimit = scanner.nextInt();
// Generate a random number within the given limits
int randomNumber = random.nextInt(upperLimit - lowerLimit + 1) + lowerLimit;
// Print messages based on the random number generated
System.out.println("Generated Random Number: " + randomNumber);
if (randomNumber < 0) {
    System.out.println("The number is negative.");
} else if (randomNumber >= 0 && randomNumber <= 10) {
    System.out.println("The number is between 0 and 10.");
} else if (randomNumber > 10 && randomNumber <= 50) {
    System.out.println("The number is between 11 and 50.");
} else {
    System.out.println("The number is greater than 50.");
}
}
}
}

```

## **CODE EXPLANATION:**

1. **import java.util.Random;**

This imports the **Random** class, which is used for generating random numbers.

2. **import java.util.Scanner;**

This imports the **Scanner** class, which is used for reading user input from the console.

3. **public class RandomNumberGenerator {**

Declares the class named **RandomNumberGenerator**.

4. **public static void main(String[] args) {**

Defines the **main** method, which is the starting point of the program.

5. **Scanner scanner = new Scanner(System.in);**

Creates a **Scanner** object to take user input from the console.

6. **Random random = new Random();**  
Creates a **Random** object to generate random numbers.
7. **System.out.print("Enter the lower limit: ");**  
Prompts the user to enter the lower limit for the random number range.
8. **int lowerLimit = scanner.nextInt();**  
Reads an integer value from the user as the lower limit.
9. **System.out.print("Enter the upper limit: ");**  
Prompts the user to enter the upper limit for the random number range.
10. **int upperLimit = scanner.nextInt();**  
Reads an integer value from the user as the upper limit.
11. **int randomNumber = random.nextInt(upperLimit - lowerLimit + 1) + lowerLimit;**  
Generates a random number within the range **[lowerLimit, upperLimit]**. The formula ensures the number is inclusive of both limits.
12. **System.out.println("Generated Random Number: " + randomNumber);**  
Displays the generated random number to the user.
13. **if (randomNumber < 0) {**  
Checks if the generated random number is negative.
14. **System.out.println("The number is negative.");**  
Prints a message indicating the number is negative.
15. **} else if (randomNumber >= 0 && randomNumber <= 10) {**  
Checks if the generated number is between 0 and 10, inclusive.
16. **System.out.println("The number is between 0 and 10.");**  
Prints a message indicating the number falls in the range [0, 10].
17. **} else if (randomNumber > 10 && randomNumber <= 50) {**  
Checks if the generated number is between 11 and 50, inclusive.
18. **System.out.println("The number is between 11 and 50.");**  
Prints a message indicating the number falls in the range [11, 50].
19. **} else {**  
If none of the above conditions are met, this block executes.
20. **System.out.println("The number is greater than 50.");**  
Prints a message indicating the number is greater than 50.

21. }

Closes the **if-else** block.

22. }

Closes the **main** method.

23. }

Closes the class definition.

## **OUTPUT:**

```
Enter the lower limit: 2
Enter the upper limit: 40
Generated Random Number: 7
The number is between 0 and 10.
```

## **RESULT:**

Thus, the program had been successfully executed.