C-DAC Mumbai

Date 25/09/2024

Subject: Algorithm and Data Structure
Assignment 1

Solve the assignment with following thing to be added in each question.

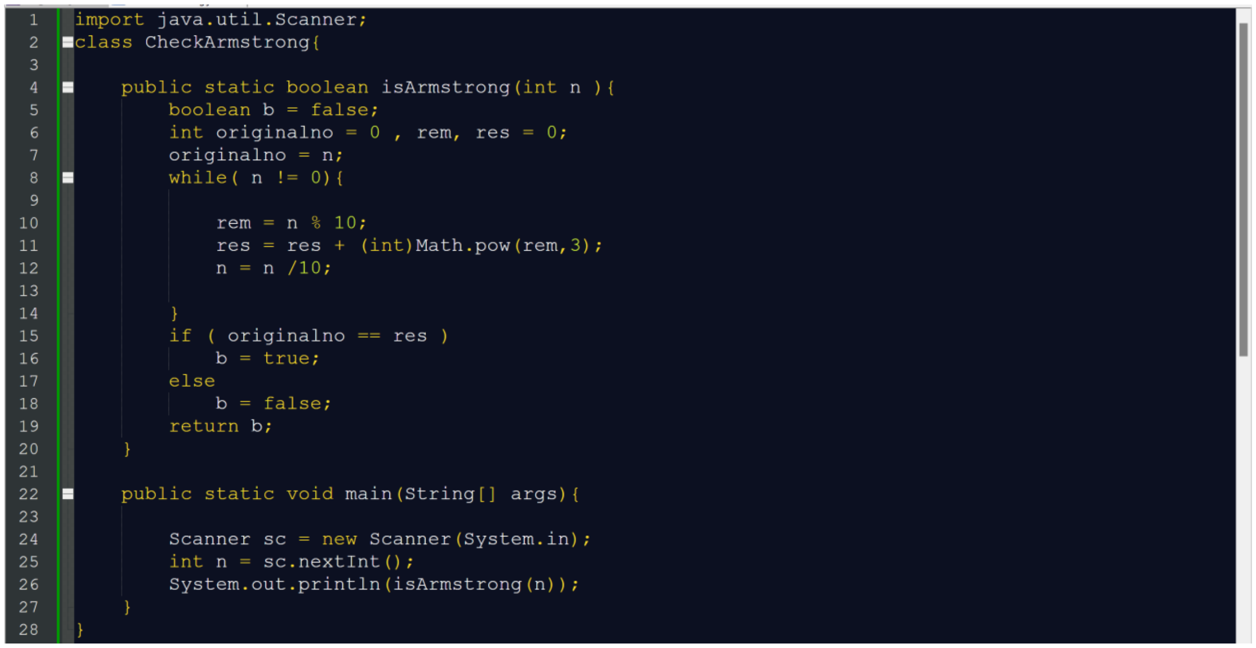
- -Program
- -Flow chart
- -Explanation
- -Output
- -Time and Space complexity

1. Armstrong Number
Problem: Write a Java program to check if a given number is an Armstrong number.
Test Cases:

Input: 153
Output: true
Input: 123
Output: false

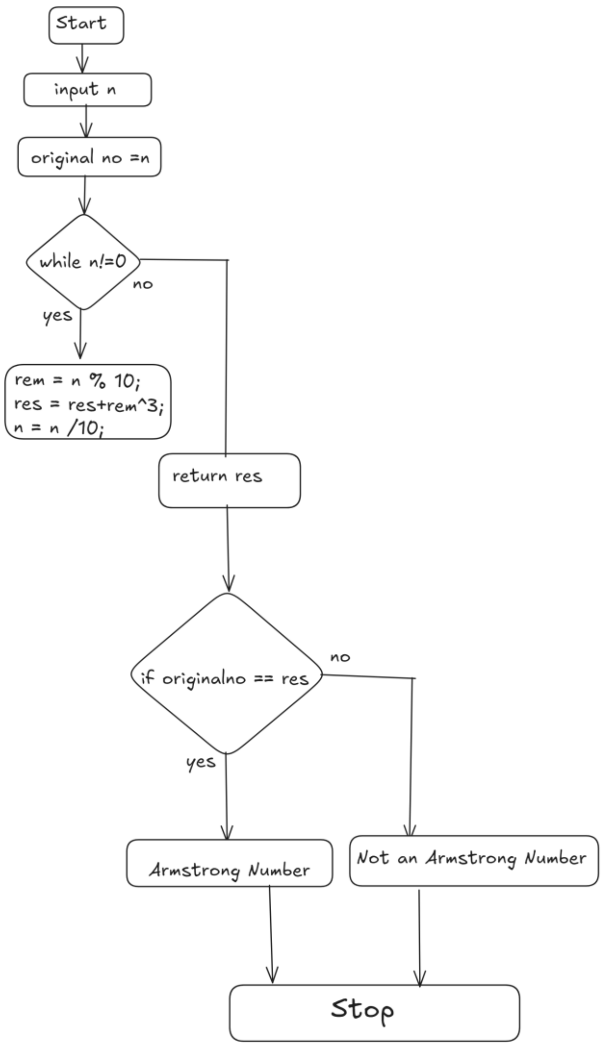
Program:

```
import java.util.Scanner;
class CheckArmstrong{
        public static boolean isArmstrong(int n ){
                boolean b = false;
                int originalno = 0 , rem, res = 0;
                originalno = n;
                while( n != 0){
                        rem = n % 10;
                        res = res + (int)Math.pow(rem,3);
                        n = n /10;
                }
                if ( originalno == res )
                        b = true;
                else
                        b = false;
                return b;
        }
        public static void main(String[] args){

                Scanner sc = new Scanner(System.in);
                int n = sc.nextInt();
                System.out.println(isArmstrong(n));
        }
}
```



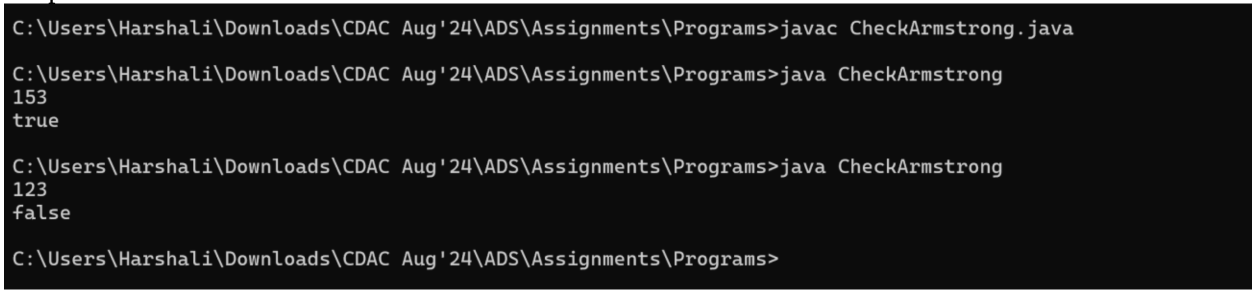
```java
import java.util.Scanner;
class CheckArmstrong{

    public static boolean isArmstrong(int n ){
        boolean b = false;
        int originalno = 0 , rem, res = 0;
        originalno = n;
        while( n != 0){

            rem = n % 10;
            res = res + (int)Math.pow(rem,3);
            n = n /10;

        }
        if ( originalno == res )
            b = true;
        else
            b = false;
        return b;
    }

    public static void main(String[] args){

        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        System.out.println(isArmstrong(n));
    }
}
```

Flowchart:



Explanation:
In first step we take a number as input and store it in one variable to save its value. After that we run a while loop until n!= 0, and in the loop we do the calculations
For example, take n as 153
In while loop,

Iteration	Remainder	Value of n	Result1(initially 0)
1st	153%10 = 3	153/10 = 15	0+3*3*3 = 27
2nd	15%10=5	15/10=1	27+5*5*5 = 152
3rd	1%10 = 1	1/10 = 0	152+1*1*1 = 153

Now as n is equal to 0 we come out of the loop. Next we check if original number is equal to result. If yes, then it is an Armstrong number, if not, then its not an Armstrong number.
Output:


```
C:\Users\Harshali\Downloads\CDAC Aug'24\ADS\Assignments\Programs>javac CheckArmstrong.java

C:\Users\Harshali\Downloads\CDAC Aug'24\ADS\Assignments\Programs>java CheckArmstrong
153
true

C:\Users\Harshali\Downloads\CDAC Aug'24\ADS\Assignments\Programs>java CheckArmstrong
123
false

C:\Users\Harshali\Downloads\CDAC Aug'24\ADS\Assignments\Programs>
```

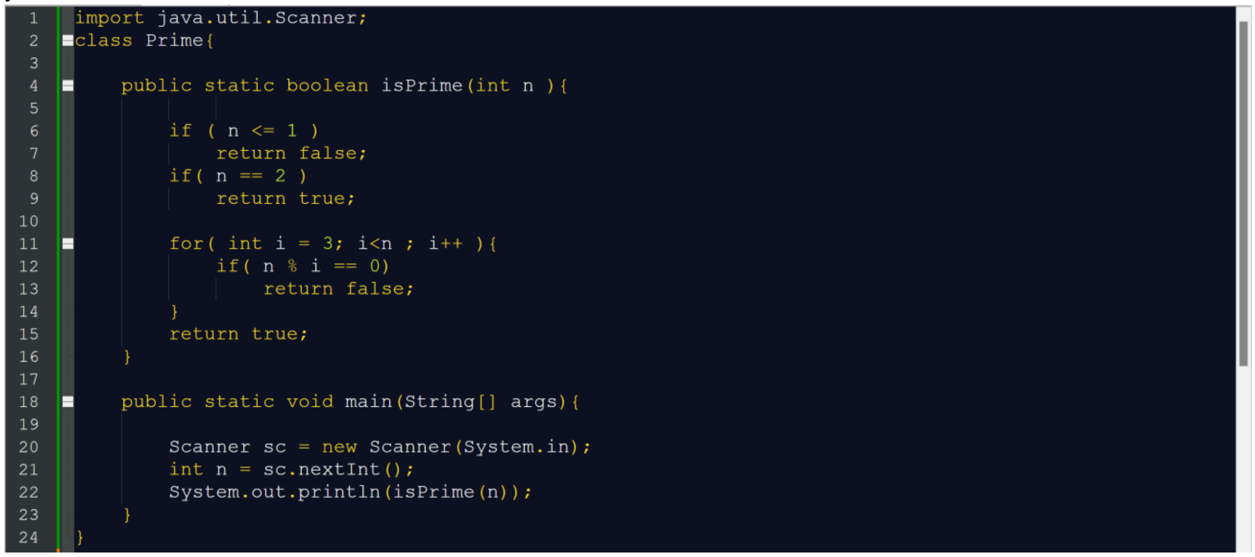
TC = O(n) SC = O(n)
This is because we need to iterate through each digit of the number to calculate the sum of the digits raised to the power of the number of digits.

2. Prime Number
Problem: Write a Java program to check if a given number is prime.

Test Cases:

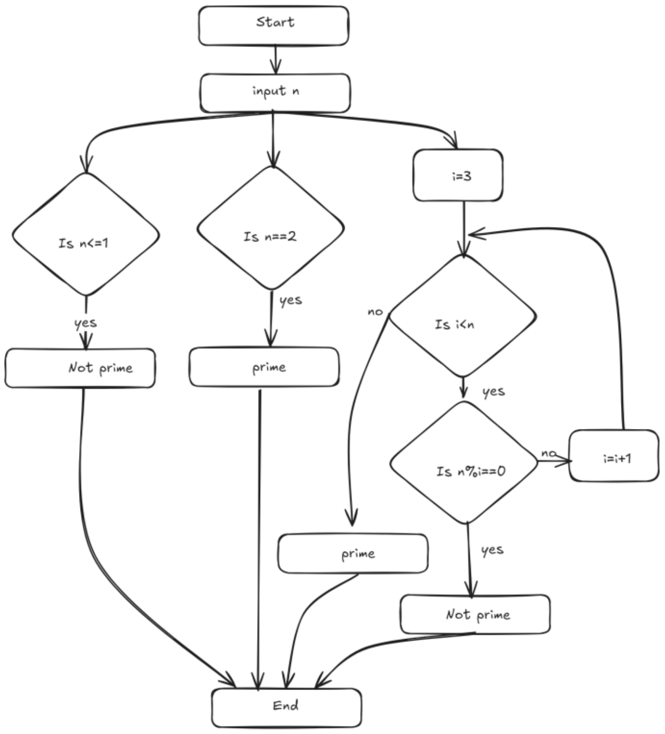
Input: 29
Output: true
Input: 15
Output: false

Program:
```
import java.util.Scanner;
class Prime{

        public static boolean isPrime(int n ){

                if ( n <= 1 )
                        return false;
                if( n == 2 )
                        return true;

                for( int i = 3; i<n ; i++ ){
                        if( n % i == 0)
                                return false;
                }
        return true;
        }

        public static void main(String[] args){

                Scanner sc = new Scanner(System.in);
                int n = sc.nextInt();
                System.out.println(isPrime(n));
        }
}
```

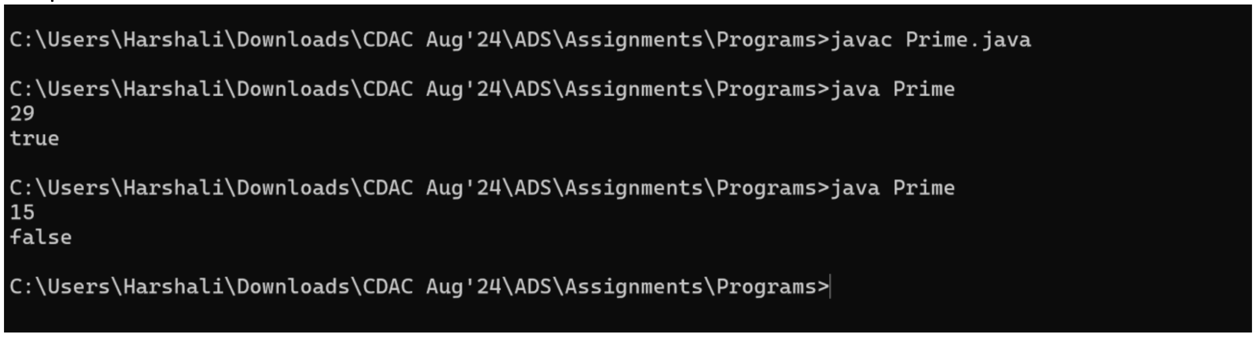


```
import java.util.Scanner;
class Prime{

    public static boolean isPrime(int n ){

        if ( n <= 1 )
            return false;
        if( n == 2 )
            return true;

        for( int i = 3; i<n ; i++ ){
            if( n % i == 0)
                return false;
        }
        return true;
    }

    public static void main(String[] args){

        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        System.out.println(isPrime(n));
    }
}
```

Flowchart:



Explanation:
The code checks if a given number is prime. The method first checks if n is less than or equal to 1 (non-prime numbers) or equal to 2 (prime), returning false or true accordingly.
For numbers greater than 2, it iterates through integers starting from 3 up to n-1.
If n is divisible by any of these numbers, it returns false (indicating the number is not prime).
Output:



```
C:\Users\Harshali\Downloads\CDAC Aug'24\ADS\Assignments\Programs>javac Prime.java

C:\Users\Harshali\Downloads\CDAC Aug'24\ADS\Assignments\Programs>java Prime
29
true

C:\Users\Harshali\Downloads\CDAC Aug'24\ADS\Assignments\Programs>java Prime
15
false

C:\Users\Harshali\Downloads\CDAC Aug'24\ADS\Assignments\Programs>
```

Time complexity: O(n)
Space complexity: O(1)

3. Factorial
Problem: Write a Java program to compute the factorial of a given number.

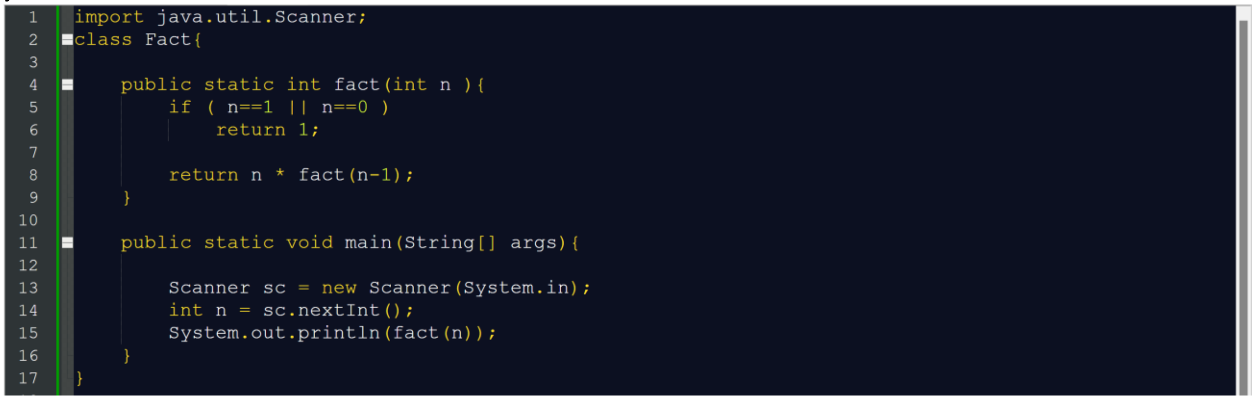
Test Cases:

Input: 5
Output: 120
Input: 0
Output: 1

Program:

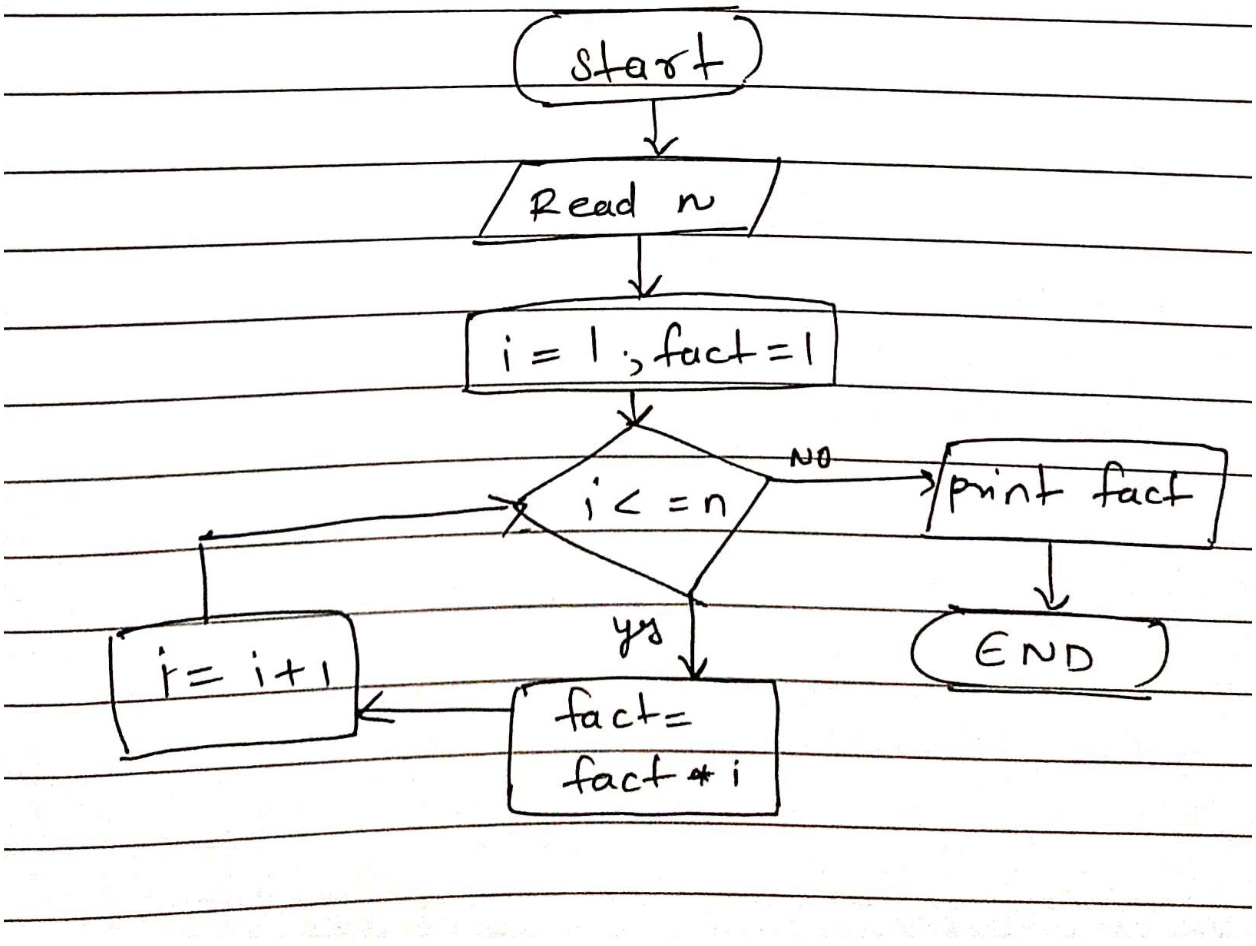
```
import java.util.Scanner;
class Fact{

        public static int fact(int n ){
                if ( n==1 || n==0 )
                        return 1;

                return n * fact(n-1);
        }

        public static void main(String[] args){

                Scanner sc = new Scanner(System.in);
                int n = sc.nextInt();
                System.out.println(fact(n));
        }
```

}



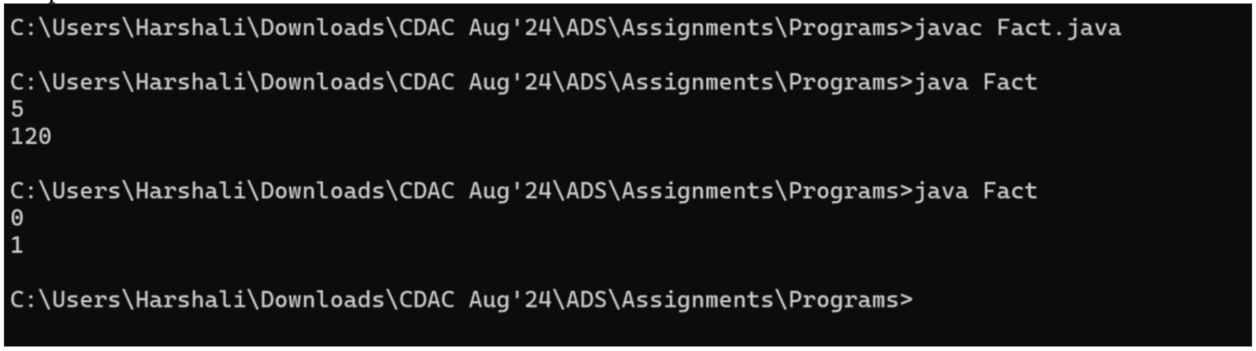
```
import java.util.Scanner;
class Fact{

    public static int fact(int n ){
        if ( n==1 || n==0 )
            return 1;

        return n * fact(n-1);
    }

    public static void main(String[] args){

        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        System.out.println(fact(n));
    }
}
```

Flowchart:



Explanation:
This program calculates the factorial of a given number using recursion.

1. The fact(int n) method is a recursive function that has a base case: if n is 0 or 1, it returns 1 (since 0! = 1! = 1).
2. For values of n greater than 1, the method calls itself with n-1 and multiplies the result by n, thus calculating n * (n-1) * ... * 1.

Output:

```
C:\Users\Harshali\Downloads\CDAC Aug'24\ADS\Assignments\Programs>javac Fact.java

C:\Users\Harshali\Downloads\CDAC Aug'24\ADS\Assignments\Programs>java Fact
5
120

C:\Users\Harshali\Downloads\CDAC Aug'24\ADS\Assignments\Programs>java Fact
0
1

C:\Users\Harshali\Downloads\CDAC Aug'24\ADS\Assignments\Programs>
```
Time complexity: O(n)

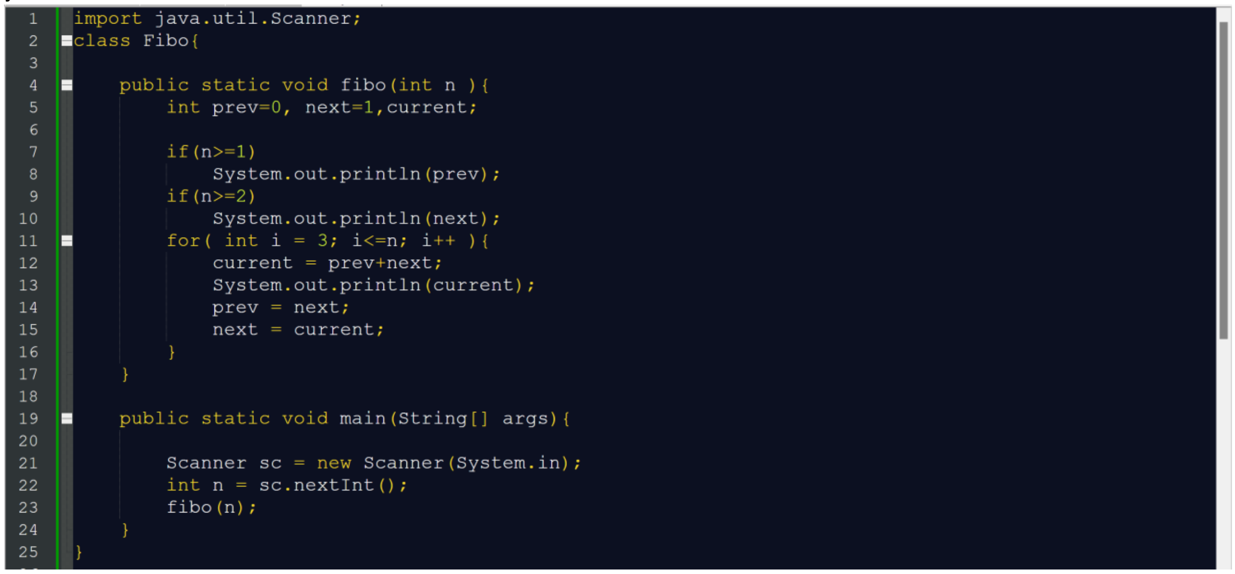
Space complexity: O(n) (due to call stack

4. Fibonacci Series
Problem: Write a Java program to print the first n numbers in the Fibonacci series.

Test Cases:

Input: n = 5
Output: [0, 1, 1, 2, 3]
Input: n = 8
Output: [0, 1, 1, 2, 3, 5, 8, 13]

Program:
```
import java.util.Scanner;
class Fibo{

	public static void fibo(int n ){
		int prev=0, next=1,current;

		if(n>=1)
			System.out.println(prev);
		if(n>=2)
			System.out.println(next);
		for( int i = 3; i<=n; i++ ){
			current = prev+next;
			System.out.println(current);
			prev = next;
			next = current;
		}
	}
	public static void main(String[] args){

		Scanner sc = new Scanner(System.in);
		int n = sc.nextInt();
		fibo(n);
	}
}
```

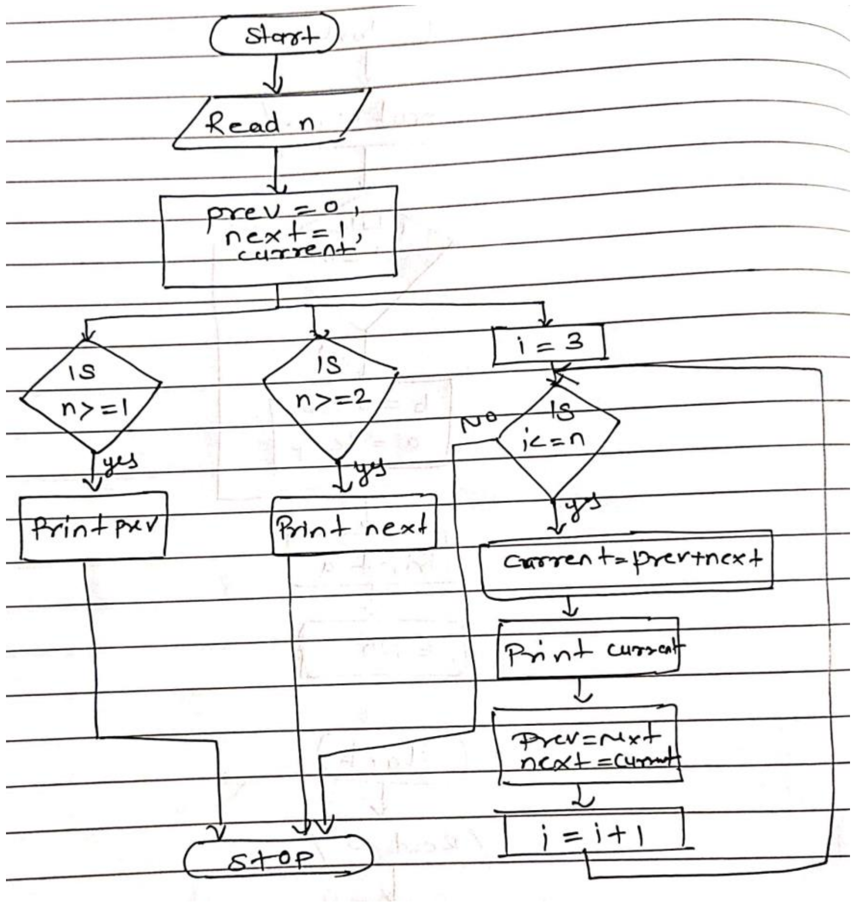


```
import java.util.Scanner;
class Fibo{

    public static void fibo(int n ){
        int prev=0, next=1,current;

        if(n>=1)
            System.out.println(prev);
        if(n>=2)
            System.out.println(next);
        for( int i = 3; i<=n; i++ ){
            current = prev+next;
            System.out.println(current);
            prev = next;
            next = current;
        }
    }

    public static void main(String[] args){

        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        fibo(n);
    }
}
```

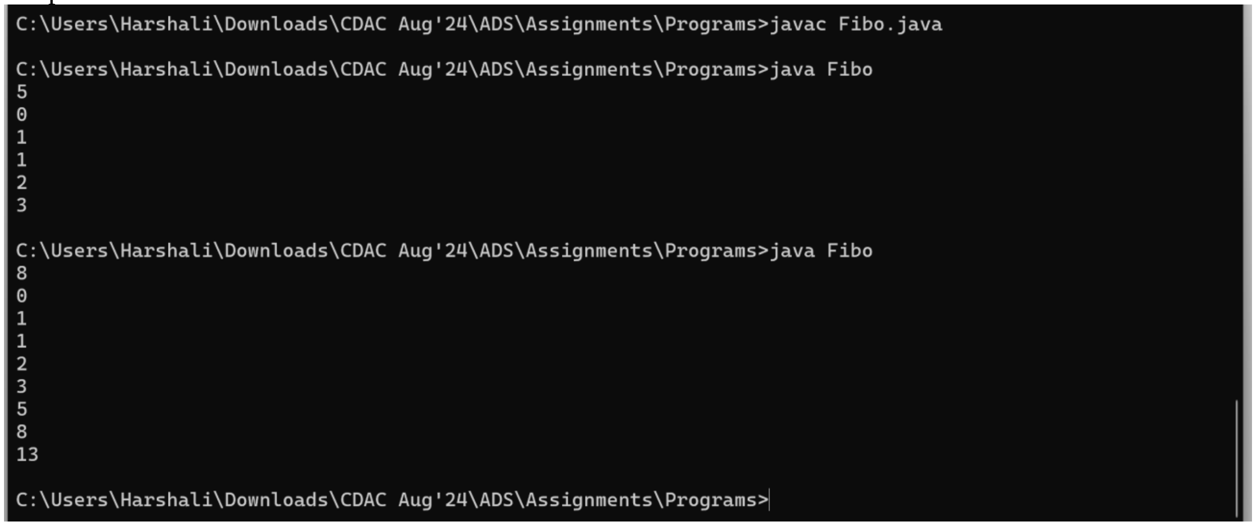
Explanation:
This program generates the first n numbers of the Fibonacci sequence.

1. The fibo(int n) method prints the Fibonacci sequence up to the nth term. It initializes two variables, prev (0) and next (1), which are the first two Fibonacci numbers.
2. If n is at least 1, it prints the first Fibonacci number (0). If n is at least 2, it prints the second Fibonacci number (1).
3. For values of n greater than 2, the program enters a loop starting from the 3rd Fibonacci number, calculating each subsequent number as the sum of the two preceding ones. It prints each value and updates the variables prev and next accordingly.

Flowchart:



Output:



```
C:\Users\Harshali\Downloads\CDAC Aug'24\ADS\Assignments\Programs>javac Fibo.java

C:\Users\Harshali\Downloads\CDAC Aug'24\ADS\Assignments\Programs>java Fibo
5
0
1
1
2
3

C:\Users\Harshali\Downloads\CDAC Aug'24\ADS\Assignments\Programs>java Fibo
8
0
1
1
2
3
5
8
13

C:\Users\Harshali\Downloads\CDAC Aug'24\ADS\Assignments\Programs>
```

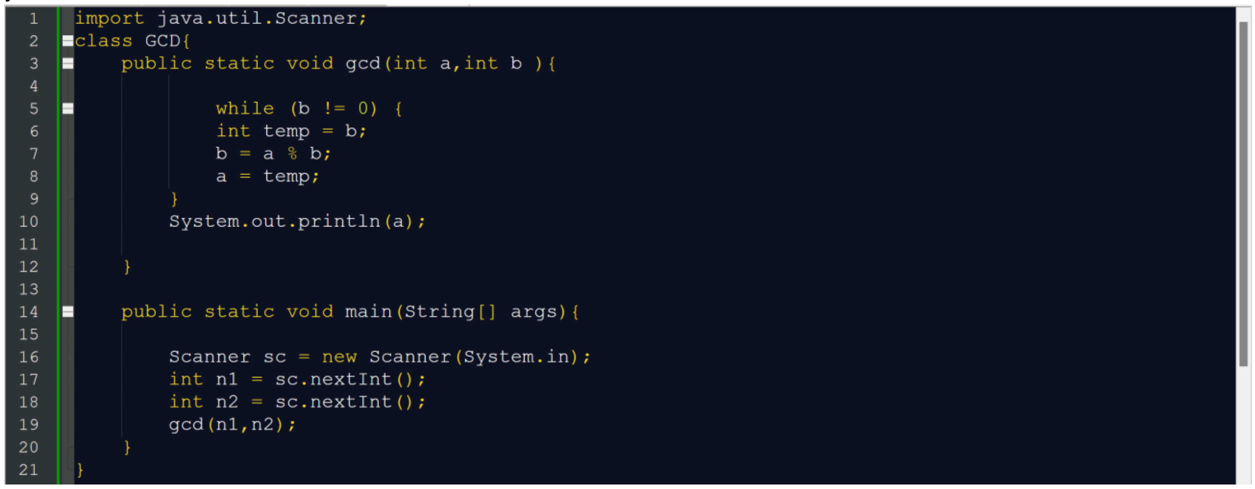
Time complexity: O(n)

Space complexity: O(1)

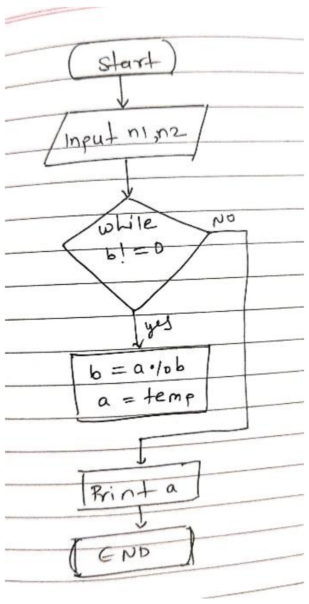
5. Find GCD
Problem: Write a Java program to find the Greatest Common Divisor (GCD) of two numbers.

Test Cases:

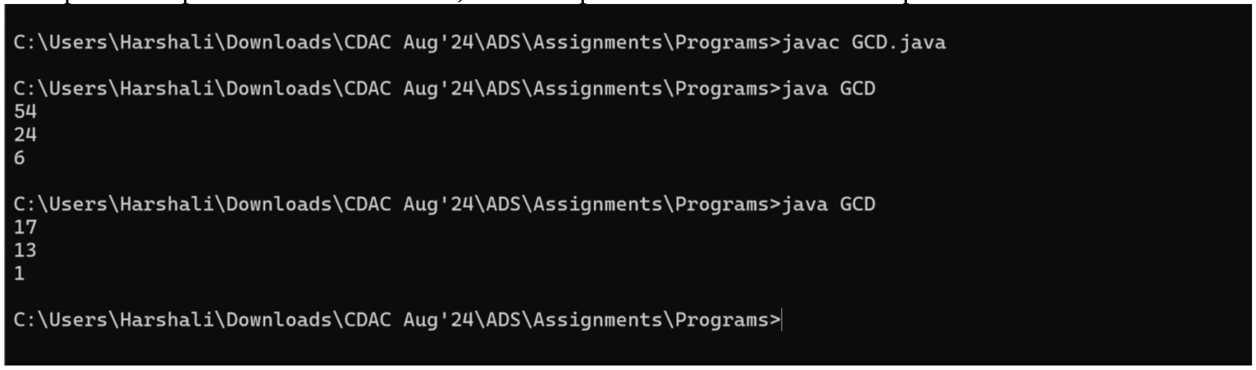
Input: a = 54, b = 24
Output: 6
Input: a = 17, b = 13
Output: 1

Program:
```
import java.util.Scanner;
class GCD{
        public static void gcd(int a,int b ){

                        while (b != 0) {
       int temp = b;
       b = a % b;
       a = temp;
    }
    System.out.println(a);
        }
        public static void main(String[] args){
                Scanner sc = new Scanner(System.in);
                int n1 = sc.nextInt();
                int n2 = sc.nextInt();
                gcd(n1,n2);
        }
}
```

```
import java.util.Scanner;
class GCD{
    public static void gcd(int a,int b ){

            while (b != 0) {
            int temp = b;
            b = a % b;
            a = temp;
        }
        System.out.println(a);

    }

    public static void main(String[] args){

        Scanner sc = new Scanner(System.in);
        int n1 = sc.nextInt();
        int n2 = sc.nextInt();
        gcd(n1,n2);
    }
}
```

Flowchart:



Explanation:
This program is to find GCD of two numbers using the Euclidean algorithm.
In the gcd(int a, int b) method, it keeps dividing the larger number by the smaller one until the remainder is zero.
The remainder is assigned to b, and a becomes the old value of b.
This process repeats until b becomes 0, at which point a holds the GCD. Output:



```
C:\Users\Harshali\Downloads\CDAC Aug'24\ADS\Assignments\Programs>javac GCD.java

C:\Users\Harshali\Downloads\CDAC Aug'24\ADS\Assignments\Programs>java GCD
54
24
6

C:\Users\Harshali\Downloads\CDAC Aug'24\ADS\Assignments\Programs>java GCD
17
13
1

C:\Users\Harshali\Downloads\CDAC Aug'24\ADS\Assignments\Programs>
```

Time complexity of the Euclidean algorithm is O(log(min(a,b))). This is because the size of the numbers involved is reduced exponentially with each iteration.
Space complexity: O(1)

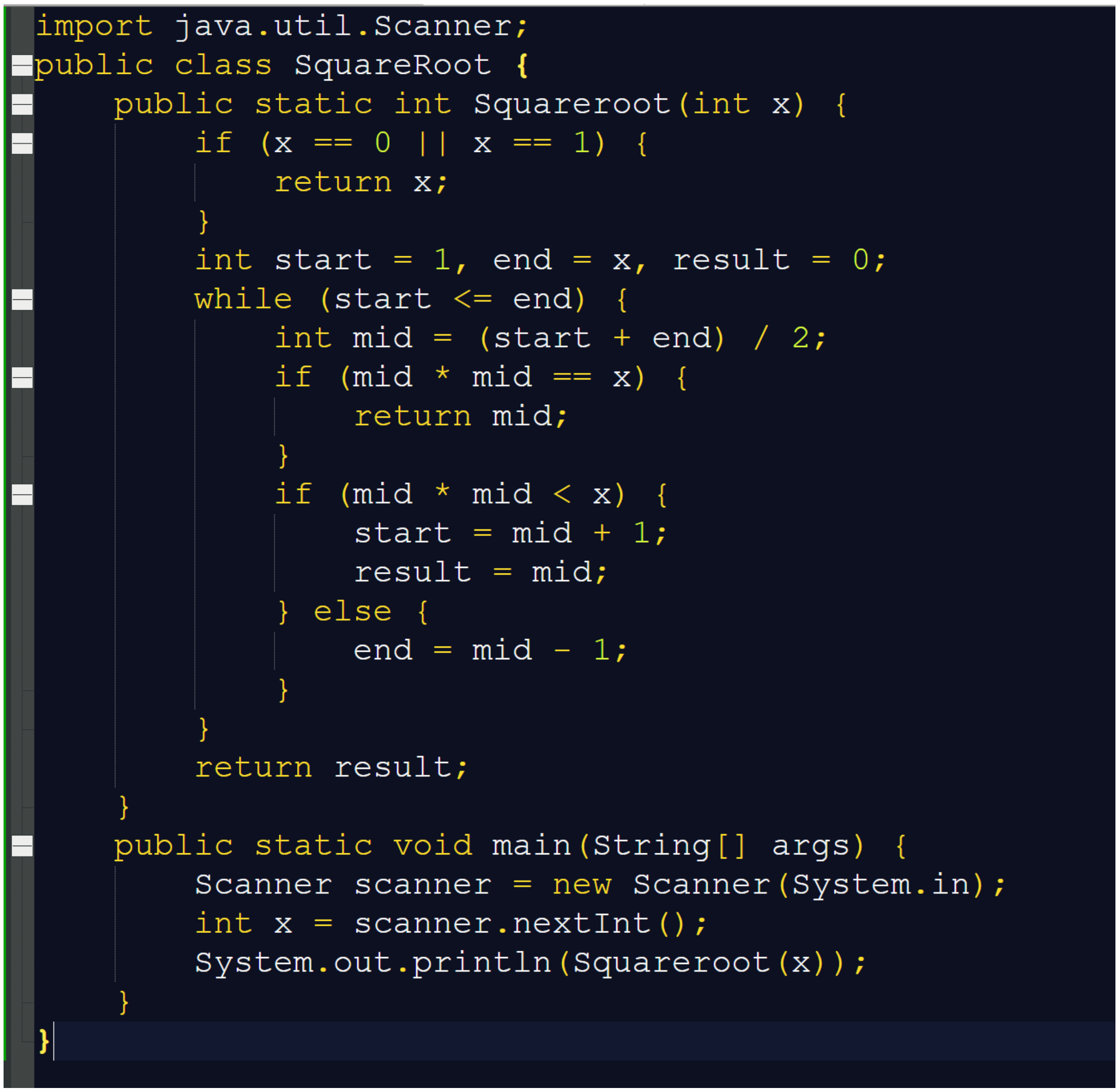
6. Find Square Root
Problem: Write a Java program to find the square root of a given number (using integer approximation).

Test Cases:

Input: x = 16
Output: 4
Input: x = 27
Output: 5

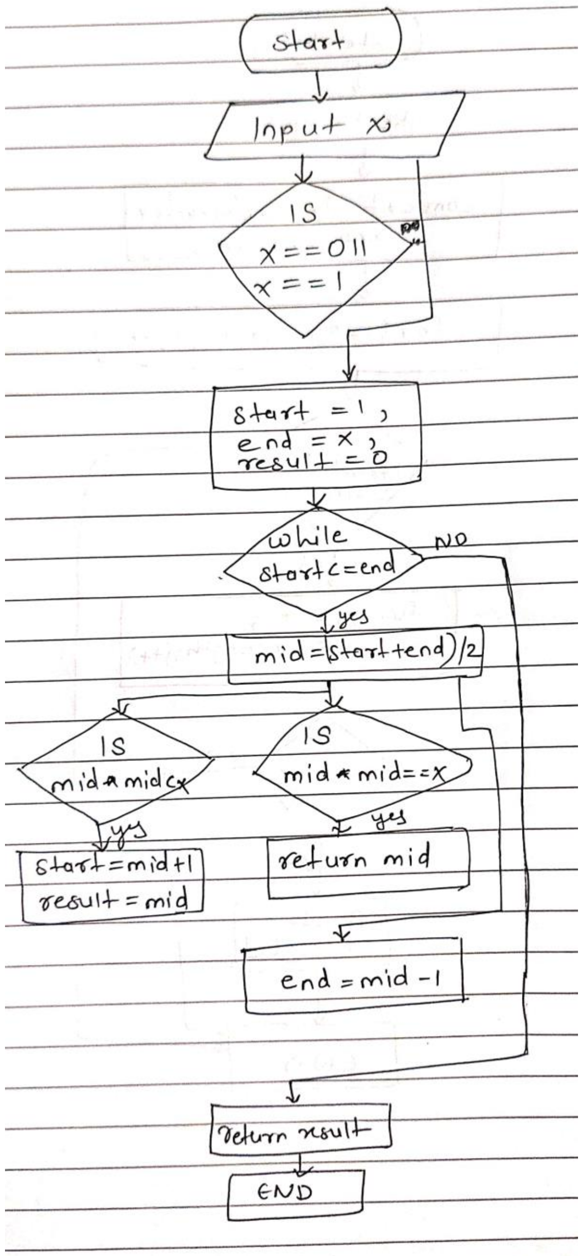
Program:

```
import java.util.Scanner;
public class SquareRoot {
             public static int Squareroot(int x) {
        if (x == 0 || x == 1) {
            return x;
        }
        int start = 1, end = x, result = 0;
        while (start <= end) {
            int mid = (start + end) / 2;
            if (mid * mid == x) {
                return mid;
            }
            if (mid * mid < x) {
                start = mid + 1;
                result = mid;
            } else {
                end = mid - 1;
            }
        }
        return result;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int x = scanner.nextInt();
        System.out.println(Squareroot(x));
    }
}
```

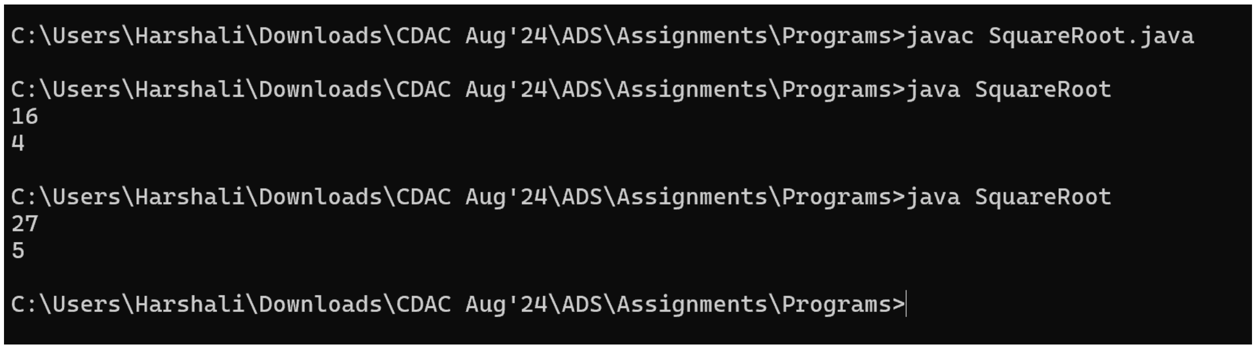


```java
import java.util.Scanner;
public class SquareRoot {
    public static int Squareroot(int x) {
        if (x == 0 || x == 1) {
            return x;
        }
        int start = 1, end = x, result = 0;
        while (start <= end) {
            int mid = (start + end) / 2;
            if (mid * mid == x) {
                return mid;
            }
            if (mid * mid < x) {
                start = mid + 1;
                result = mid;
            } else {
                end = mid - 1;
            }
        }
        return result;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int x = scanner.nextInt();
        System.out.println(Squareroot(x));
    }
}
```

Flowchart:



Explanation:
This program calculates the square root of a given number x using a binary search approach
1. The Squareroot(int x) method first handles two simple cases: if x is 0 or 1, it returns x directly.
2. For numbers greater than 1, the method uses binary search. It starts with start as 1 and end as x. The mid value is calculated, and its square is checked.
3. If mid * mid equals x, it returns mid (the exact square root). If mid * mid is less than x, it updates start and continues searching in the right half, keeping track of the closest result. Otherwise, it adjusts end to search the left half.
Output:

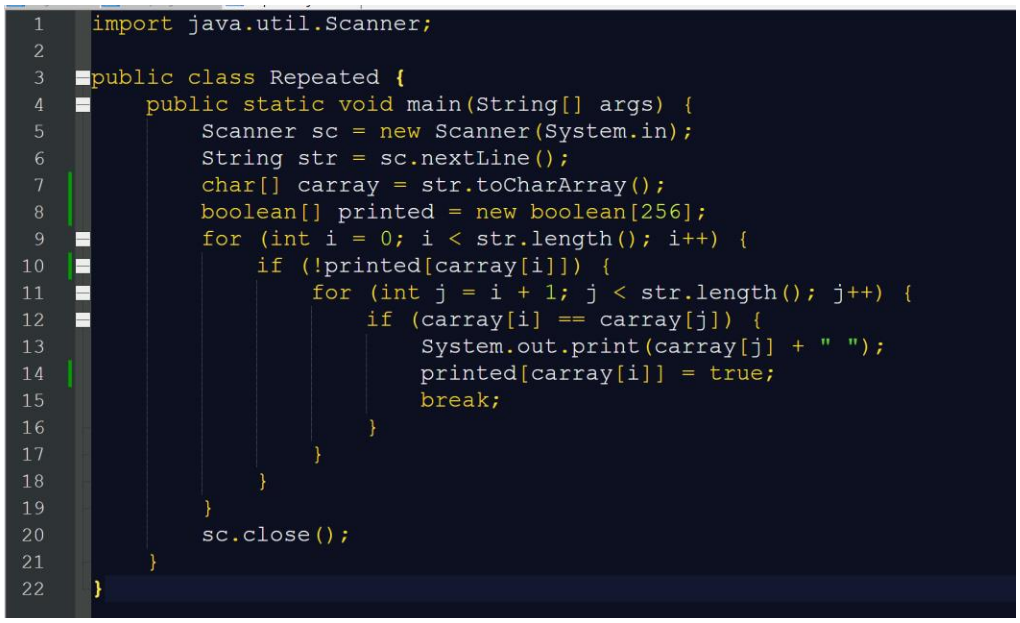

```
C:\Users\Harshali\Downloads\CDAC Aug'24\ADS\Assignments\Programs>javac SquareRoot.java

C:\Users\Harshali\Downloads\CDAC Aug'24\ADS\Assignments\Programs>java SquareRoot
16
4

C:\Users\Harshali\Downloads\CDAC Aug'24\ADS\Assignments\Programs>java SquareRoot
27
5

C:\Users\Harshali\Downloads\CDAC Aug'24\ADS\Assignments\Programs>
```

Time complexity: O(logx)
Space complexity: O(1)
7. Find Repeated Characters in a String
Problem: Write a Java program to find all repeated characters in a string.

Test Cases:

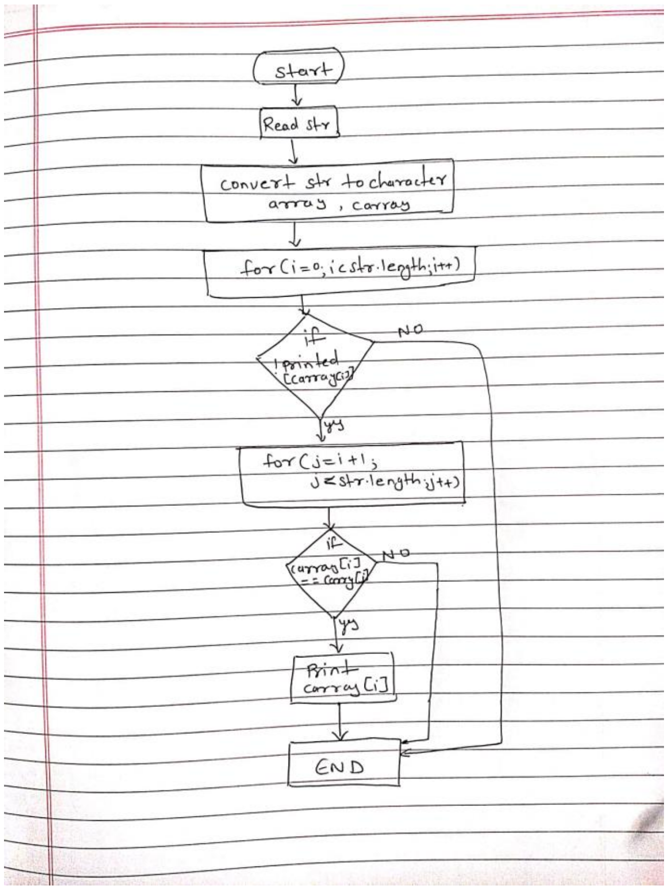
Input: "programming"
Output: ['r', 'g', 'm']
Input: "hello"
Output: ['l']
Program:

```java
import java.util.Scanner;

public class Repeated {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String str = sc.nextLine();
        char[] carray = str.toCharArray();
        boolean[] printed = new boolean[256];
        for (int i = 0; i < str.length(); i++) {
            if (!printed[carray[i]]) {
                for (int j = i + 1; j < str.length(); j++) {
                    if (carray[i] == carray[j]) {
                        System.out.print(carray[j] + " ");
                        printed[carray[i]] = true;
                        break;
                    }
                }
            }
        }
        sc.close();
    }
}
```



```java
import java.util.Scanner;

public class Repeated {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String str = sc.nextLine();
        char[] carray = str.toCharArray();
        boolean[] printed = new boolean[256];
        for (int i = 0; i < str.length(); i++) {
            if (!printed[carray[i]]) {
                for (int j = i + 1; j < str.length(); j++) {
                    if (carray[i] == carray[j]) {
                        System.out.print(carray[j] + " ");
                        printed[carray[i]] = true;
                        break;
                    }
                }
            }
        }
        sc.close();
    }
}
```

Flowchart:

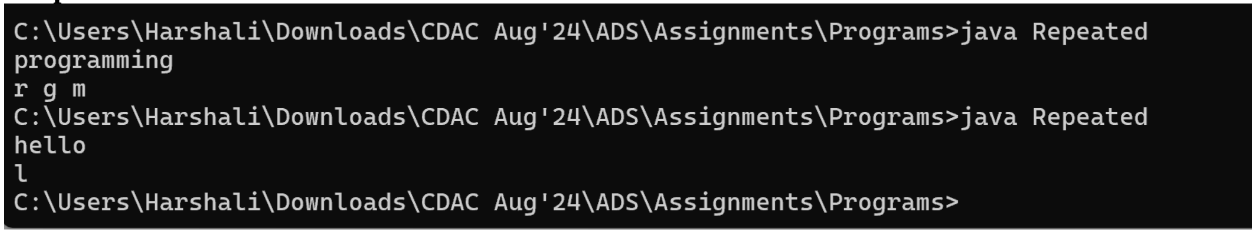

start
Read str
convert str to character array , carray
for (i=0; i<str.length; i++)
if !printed [carray[i]]
NO
yes
for (j=i+1; j<str.length; j++)
if carray[i] == carray[j]
NO
yes
Print carray[i]
END

Explanation:
This program finds and prints the first repeated characters in a given string. Here's how it works:

1. The user inputs a string, which is converted into a character array carray.

2. A boolean array printed of size 256 is used to keep track of whether a character has already been printed
3. The outer loop iterates through each character in the string. If the character has not been printed yet, the inner loop checks for any repeated occurrence of that character.
4. Once a repeated character is found, it's printed, and the corresponding printed entry is set to true to avoid printing the same character again.
5. The program continues until all repeated characters have been checked and printed.

Output:



```
C:\Users\Harshali\Downloads\CDAC Aug'24\ADS\Assignments\Programs>java Repeated
programming
r g m
C:\Users\Harshali\Downloads\CDAC Aug'24\ADS\Assignments\Programs>java Repeated
hello
l
C:\Users\Harshali\Downloads\CDAC Aug'24\ADS\Assignments\Programs>
```

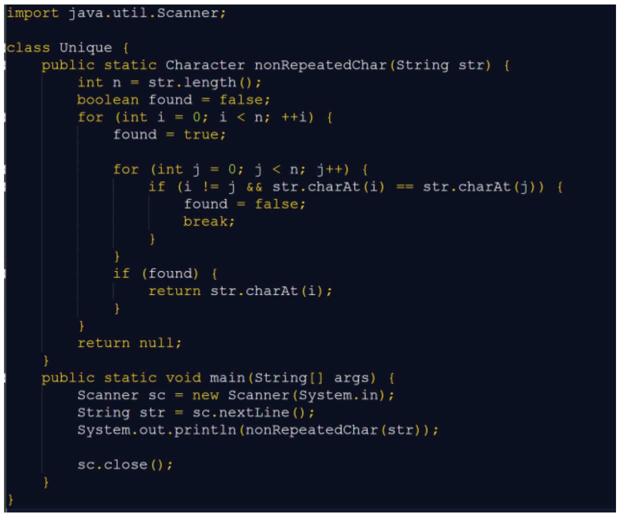
Time complexity: O(n^2)
Space complexity: O(n)

8. First Non-Repeated Character
Problem: Write a Java program to find the first non-repeated character in a string.

Test Cases:

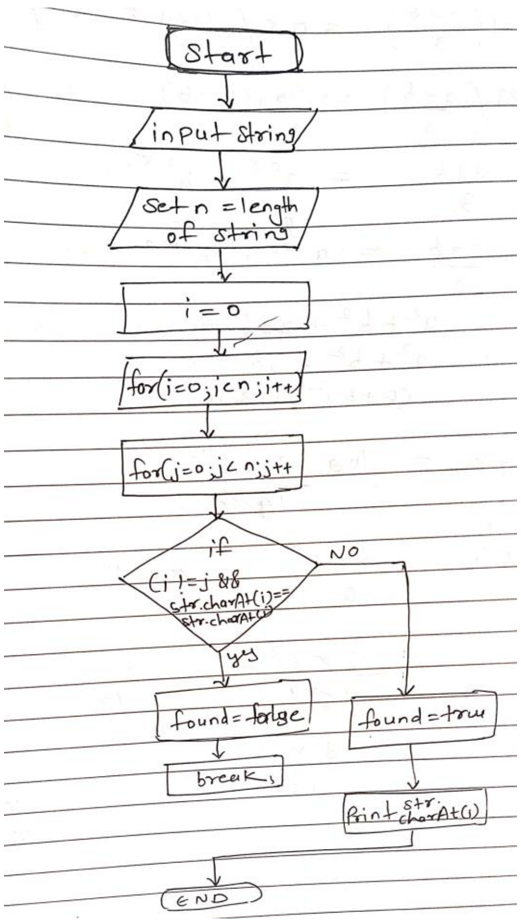
Input: "stress"
Output: 't'
Input: "aabbcc"
Output: null
Program:

```
import java.util.Scanner;

class Unique {
    public static Character nonRepeatedChar(String str) {
        int n = str.length();
        boolean found = false;
        for (int i = 0; i < n; ++i) {
            found = true;

            for (int j = 0; j < n; j++) {
                if (i != j && str.charAt(i) == str.charAt(j)) {
                    found = false;
                    break;
                }
            }
            if (found) {
                return str.charAt(i);
            }
        }
        return null;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String str = sc.nextLine();
        System.out.println(nonRepeatedChar(str));

        sc.close();
    }
}
```



```
import java.util.Scanner;

class Unique {
    public static Character nonRepeatedChar(String str) {
        int n = str.length();
        boolean found = false;
        for (int i = 0; i < n; ++i) {
            found = true;

            for (int j = 0; j < n; j++) {
                if (i != j && str.charAt(i) == str.charAt(j)) {
                    found = false;
                    break;
                }
            }
            if (found) {
                return str.charAt(i);
            }
        }
        return null;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String str = sc.nextLine();
        System.out.println(nonRepeatedChar(str));

        sc.close();
    }
}
```

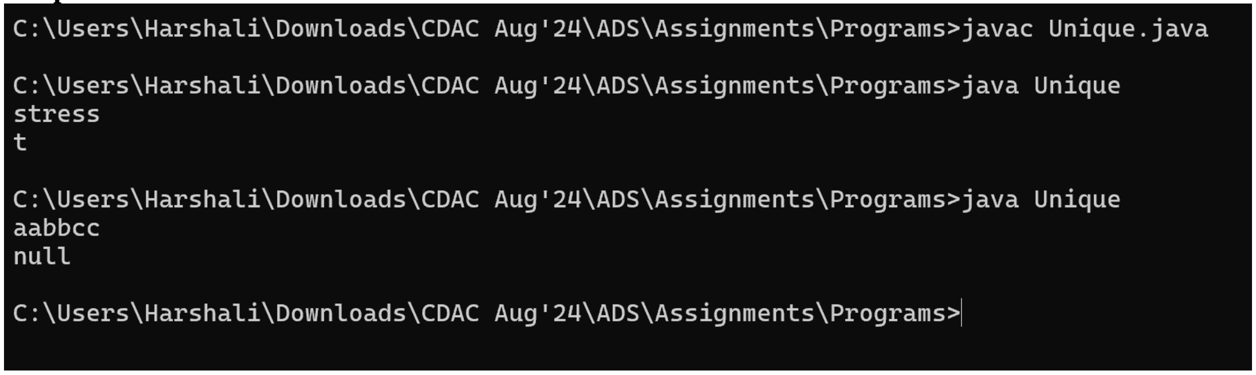
Flowchart:



Explanation:
This program finds the first non-repeated character in a given string

1. The nonRepeatedChar(String str) method iterates over each character in the string using two nested loops.
2. For each character at index i, it assumes the character is unique (found = true) and then checks it against every other character in the string using a second loop.
3. If the character at index i is found at any other position j (i.e., i != j and the characters match), found is set to false, and the inner loop breaks.
4. If found remains true after checking all other characters, the method returns that character as the first non-repeated one.
5. If no non-repeated character is found, the method returns null.
6. In the main method, the program takes user input and calls nonRepeatedChar, printing the result.

Output:



```
C:\Users\Harshali\Downloads\CDAC Aug'24\ADS\Assignments\Programs>javac Unique.java

C:\Users\Harshali\Downloads\CDAC Aug'24\ADS\Assignments\Programs>java Unique
stress
t

C:\Users\Harshali\Downloads\CDAC Aug'24\ADS\Assignments\Programs>java Unique
aabbcc
null

C:\Users\Harshali\Downloads\CDAC Aug'24\ADS\Assignments\Programs>
```

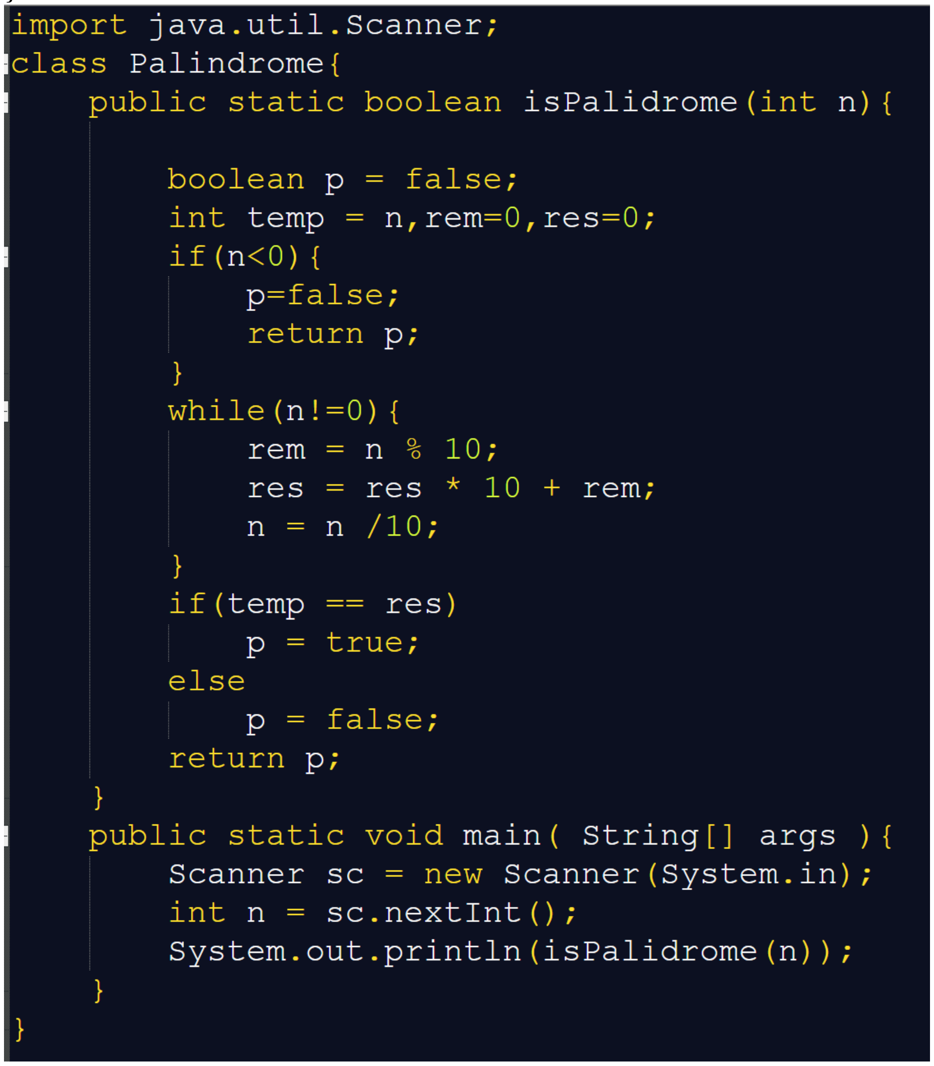
Time complexity: O(n^2)
Space complexity: O(n)

9. Integer Palindrome
Problem: Write a Java program to check if a given integer is a palindrome.

Test Cases:

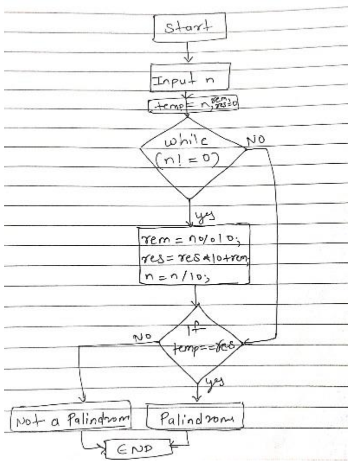
Input: 121
Output: true
Input: -121
Output: false
Program:

```
import java.util.Scanner;
class Palindrome{
        public static boolean isPalidrome(int n){

                boolean p = false;
                int temp = n,rem=0,res=0;
                if(n<0){
                        p=false;
                        return p;
                }
                while(n!=0){
                        rem = n % 10;
                        res = res * 10 + rem;
                        n = n /10;
                }
                if(temp == res)
                        p = true;
                else
                        p = false;
                return p;
        }

        public static void main( String[] args ){
                Scanner sc = new Scanner(System.in);
                int n = sc.nextInt();
                System.out.println(isPalidrome(n));
        }
}
```

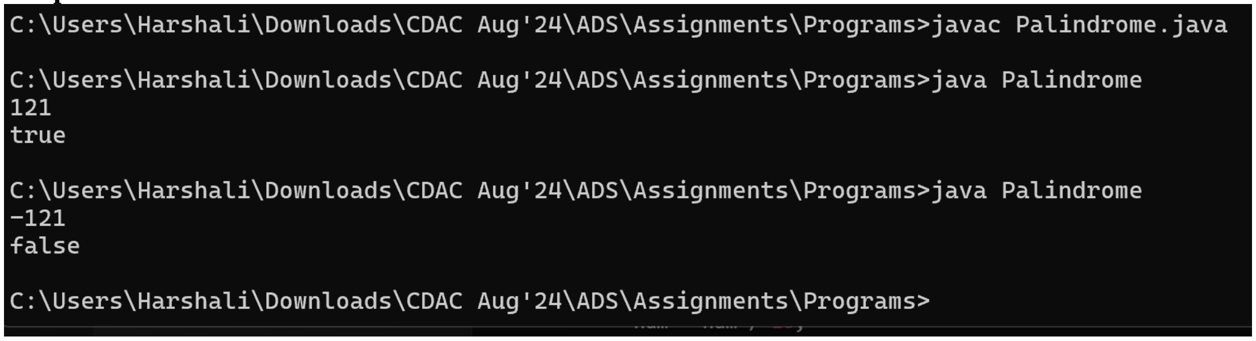


```
import java.util.Scanner;
class Palindrome{
    public static boolean isPalidrome(int n){

        boolean p = false;
        int temp = n,rem=0,res=0;
        if(n<0){
            p=false;
            return p;
        }
        while(n!=0){
            rem = n % 10;
            res = res * 10 + rem;
            n = n /10;
        }
        if(temp == res)
            p = true;
        else
            p = false;
        return p;
    }
    public static void main( String[] args ){
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        System.out.println(isPalidrome(n));
    }
}
```

Flowchart:



Explanation:
Program takes a number from user and stores it in temp.
If number is negative then its not a palindrome number.
Then we run a while loop until n not gets 0.In while loop, we calculate a reverse of a number as:
a)remainder by n%10
b)result by result*10+rem
c)n = n/10
At last, we check if temp is equal to result if yes then it’s a palindrome else its not a palindrome number.
Output:



```
C:\Users\Harshali\Downloads\CDAC Aug'24\ADS\Assignments\Programs>javac Palindrome.java

C:\Users\Harshali\Downloads\CDAC Aug'24\ADS\Assignments\Programs>java Palindrome
121
true

C:\Users\Harshali\Downloads\CDAC Aug'24\ADS\Assignments\Programs>java Palindrome
-121
false

C:\Users\Harshali\Downloads\CDAC Aug'24\ADS\Assignments\Programs>
```

Time complexity: O(log n)
Space complexity: O(1)

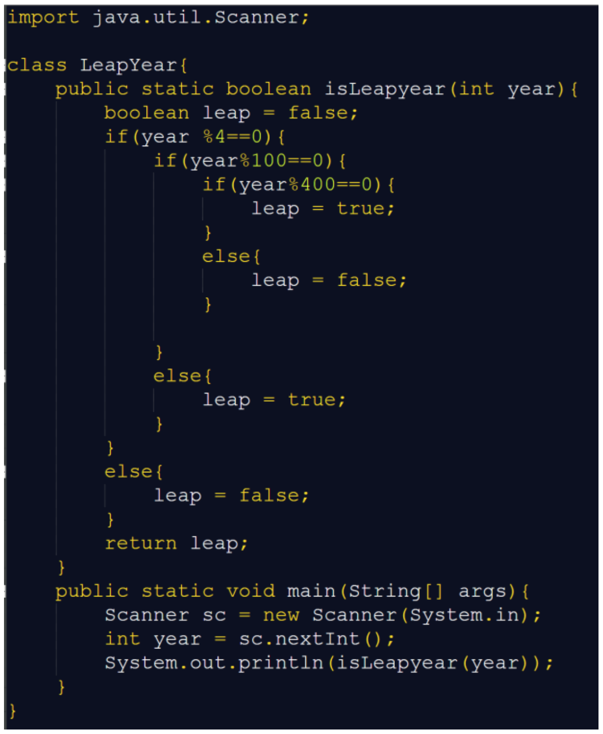
10. Leap Year
Problem: Write a Java program to check if a given year is a leap year.

Test Cases:

Input: 2020
Output: true
Input: 1900
Output: false
Program:

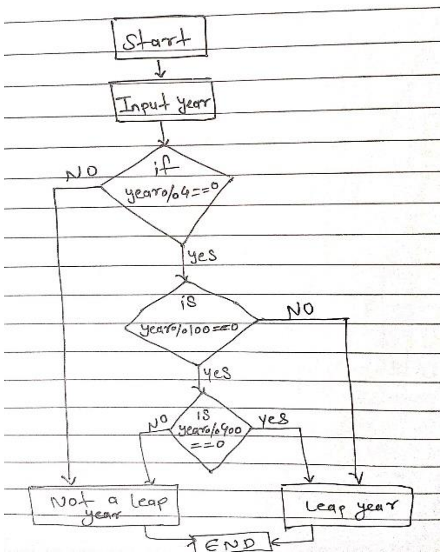
```
import java.util.Scanner;

class LeapYear{
	public static boolean isLeapyear(int year){
```

```
                boolean leap = false;
                if(year %4==0){
                        if(year%100==0){
                                if(year%400==0){
                                        leap = true;
                                }
                                else{
                                        leap = false;
                                }

                        }
                        else{
                                leap = true;
                        }
                }
                else{
                        leap = false;
                }
                return leap;
        }
        public static void main(String[] args){
                Scanner sc = new Scanner(System.in);
                int year = sc.nextInt();
                System.out.println(isLeapyear(year));
        }
}
```



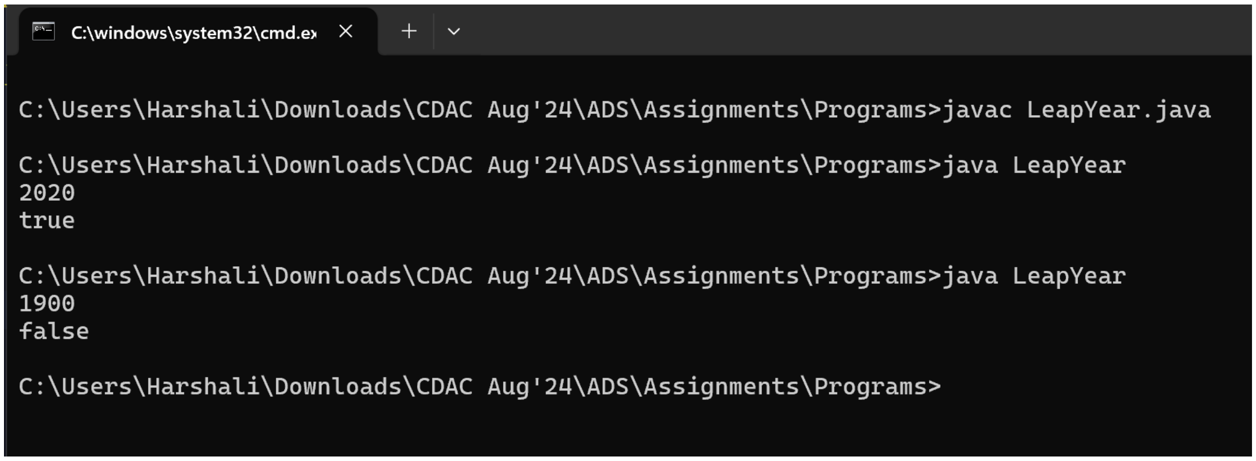
```
import java.util.Scanner;

class LeapYear{
    public static boolean isLeapyear(int year){
        boolean leap = false;
        if(year %4==0){
            if(year%100==0){
                if(year%400==0){
                    leap = true;
                }
                else{
                    leap = false;
                }

            }
            else{
                leap = true;
            }
        }
        else{
            leap = false;
        }
        return leap;
    }
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        int year = sc.nextInt();
        System.out.println(isLeapyear(year));
    }
}
```

Flowchart:



Explanation:
Program takes integer input for year and then checks for the below conditions and stores it into Boolean variable leap: A year is a leap year if it is completely divisible by 4 except 100.
If it is divisible by 100 and 400 also then it is a leap year.
Otherwise, it is not a leap year.

Output:



```
C:\Users\Harshali\Downloads\CDAC Aug'24\ADS\Assignments\Programs>javac LeapYear.java

C:\Users\Harshali\Downloads\CDAC Aug'24\ADS\Assignments\Programs>java LeapYear
2020
true

C:\Users\Harshali\Downloads\CDAC Aug'24\ADS\Assignments\Programs>java LeapYear
1900
false

C:\Users\Harshali\Downloads\CDAC Aug'24\ADS\Assignments\Programs>
```

Time complexity: O(1)

Space complexity: O(1)