

Assignment 1

1. Armstrong Number

Problem: Write a Java program to check if a given number is an Armstrong number.

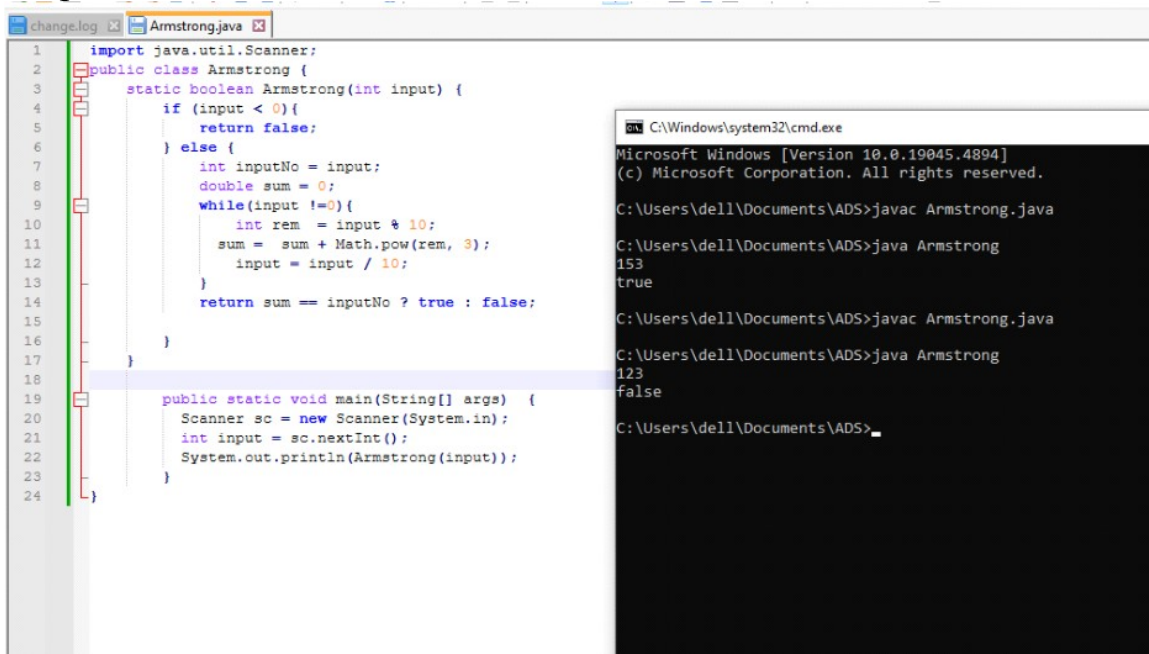
Test Cases:

Input: 153

Output: true

Input: 123

Output: false



```
1 import java.util.Scanner;
2 public class Armstrong {
3     static boolean Armstrong(int input) {
4         if (input < 0) {
5             return false;
6         } else {
7             int inputNo = input;
8             double sum = 0;
9             while(input != 0) {
10                 int rem = input % 10;
11                 sum = sum + Math.pow(rem, 3);
12                 input = input / 10;
13             }
14             return sum == inputNo ? true : false;
15         }
16     }
17 }
18
19 public static void main(String[] args) {
20     Scanner sc = new Scanner(System.in);
21     int input = sc.nextInt();
22     System.out.println(Armstrong(input));
23 }
24 }
```

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19045.4894]
(c) Microsoft Corporation. All rights reserved.

C:\Users\dell\Documents\ADS>javac Armstrong.java

C:\Users\dell\Documents\ADS>java Armstrong
153
true

C:\Users\dell\Documents\ADS>javac Armstrong.java

C:\Users\dell\Documents\ADS>java Armstrong
123
false

C:\Users\dell\Documents\ADS>
```

Explanation – I created a class Armstrong then in tht i took the base condition as input < 1 . If it is less than one i returned it false. In the elsecondition i kept while loop wih condition till the loop not reaches 0 and int he loop i kept calculating its seperate digit by using modulo 10 and then store its cube and sum. Kept claculating the digit by dividing it by 10. and check if this new sum is equal to the sum . If it is equal then true else return false.

Time Complexity – $O(\log n)$

Space Complexity- $O(1)$

2. Prime Number

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

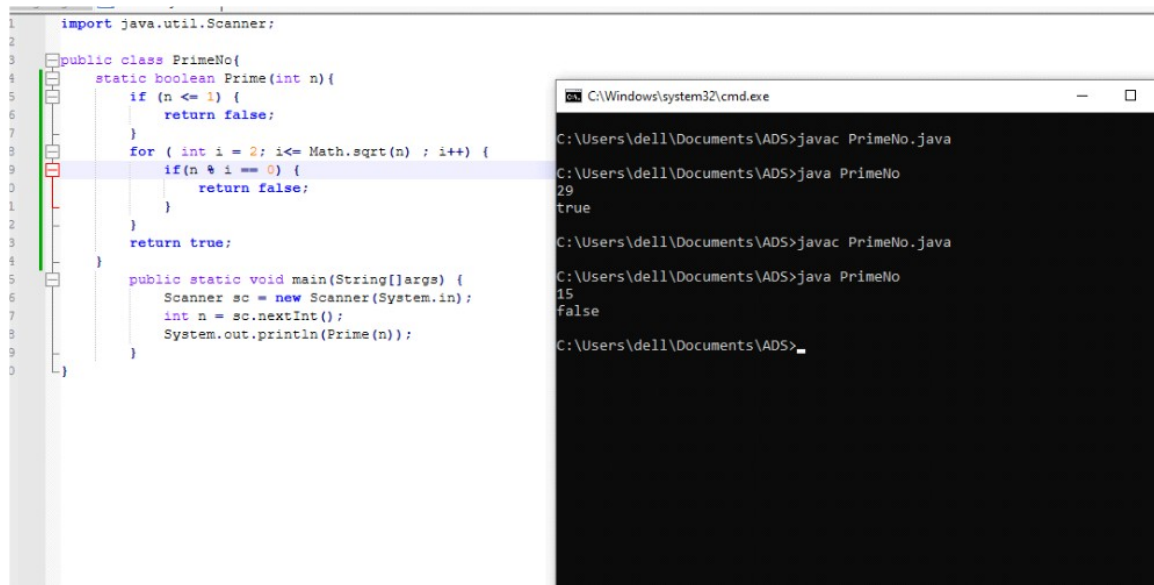
Test Cases:

Input: 29

Output: true

Input: 15

Output: false



```
1 import java.util.Scanner;
2
3 public class PrimeNo{
4     static boolean Prime(int n){
5         if (n <= 1) {
6             return false;
7         }
8         for ( int i = 2; i<= Math.sqrt(n) ; i++) {
9             if(n % i == 0) {
10                 return false;
11             }
12         }
13         return true;
14     }
15     public static void main(String[] args) {
16         Scanner sc = new Scanner(System.in);
17         int n = sc.nextInt();
18         System.out.println(Prime(n));
19     }
20 }
```

```
C:\Users\dell\Documents\ADS>javac PrimeNo.java
C:\Users\dell\Documents\ADS>java PrimeNo
29
true
C:\Users\dell\Documents\ADS>javac PrimeNo.java
C:\Users\dell\Documents\ADS>java PrimeNo
15
false
C:\Users\dell\Documents\ADS>
```

Explanation – At first I called the method prime. Then in it i kept the base condition as $n \leq 1$ if it is then return false. Then i took the for loop starting from 2 and till the square root of n to chieve it in less time complexity. we kept dividing the number with i . If its remainder is 0 then return false that it is a prime number else return true which means it is a pime number.

Time Complexity – $O(\sqrt{n})$

Space Complexity- $O(1)$

3. Factorial

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

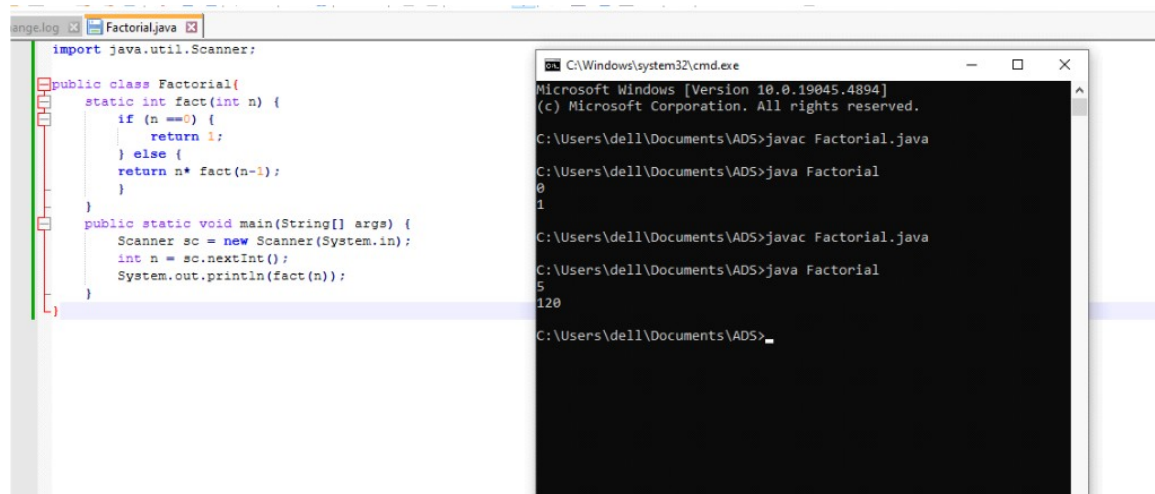
Test Cases:

Input: 5

Output: 120

Input: 0

Output: 1



The screenshot shows a Java IDE on the left with a file named 'Factorial.java'. The code is as follows:

```
import java.util.Scanner;

public class Factorial{
    static int fact(int n) {
        if (n == 0) {
            return 1;
        } else {
            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));
    }
}
```

On the right, a Windows command prompt window shows the execution of the program. It first compiles the file with 'javac Factorial.java' and then runs it three times with inputs 0, 5, and 120. The outputs are 1, 120, and 120 respectively.

Explanation - I have created a method called as fact of int return type. Then inside it I did check the condition that whether the input is equal to 0. If it is equal to 0 then it will return 1. If input is not 0, the method calculates $n * \text{fact}(n - 1)$. Which basically recursively calls until the $\text{fact}(n-1)$. This gives the factorial of a number

Time Complexity - $O(n)$

Space Complexity - $O(n)$

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]

```

import java.util.Scanner;

public class Fibonacci{
    static int fib(int n) {
        if (n <= 1) {
            return n;
        }
        return fib(n-1) + fib(n-2);
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        for(int i = 0; i < n; i++) {
            System.out.print(fib(i) + " ");
        }
    }
}

```

```

C:\Windows\system32\cmd.exe
C:\Users\dell\Documents\ADS>javac Fibonacci.java
C:\Users\dell\Documents\ADS>java Fibonacci
5
0 1 1 2 3
C:\Users\dell\Documents\ADS>java Fibonacci
8
0 1 1 2 3 5 8 13
C:\Users\dell\Documents\ADS>_

```

Explanation - I have created a method called as fib of int return type. Then inside it i checked the condition that whether the input is less than 1. If it is less than 1 then input is returned. If it is not less than 1 ,the method will calculate fib(n-1) + fib(n-2). There is a for loop which iterates from 0 to n. For each iteration, fib(i) is called to calculate the ith Fibonacci number.

Time Complexity – $O(n)$

Space Complexity- $O(n)$

5. Find GCD

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

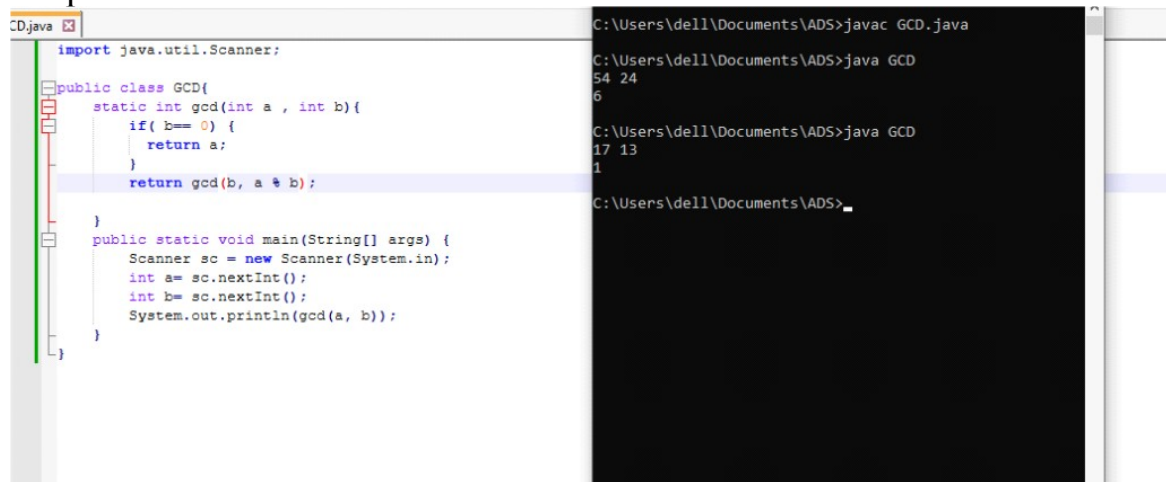
Test Cases:

Input: a = 54, b = 24

Output: 6

Input: a = 17, b = 13

Output: 1



The screenshot shows a Java IDE with two windows. The left window displays the source code for a GCD program. The right window shows the command prompt output for running the program.

```
import java.util.Scanner;

public class GCD{
    static int gcd(int a , int b){
        if( b== 0) {
            return a;
        }
        return gcd(b, a % b);
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int a= sc.nextInt();
        int b= sc.nextInt();
        System.out.println(gcd(a, b));
    }
}
```

Command Prompt Output:

```
C:\Users\dell\Documents\ADS>javac GCD.java
C:\Users\dell\Documents\ADS>java GCD
54 24
6
C:\Users\dell\Documents\ADS>java GCD
17 13
1
C:\Users\dell\Documents\ADS>
```

Explanation - I created a method called as gcd of int return type. Then inside it checked the condition that whether one of the input . b is equal to 0. If it is equal to 0 then only one input . a is returned. If it is not equal to 0 ,the method will calculate gcd(b, a%b).

Time Complexity – $O(\log \min(a, b))$

Space Complexity- $O(\log \min(a, b))$

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

Explanation - I created a method called as square of int return type. Inside this method I used `Math.sqrt(n)` which calculates the square root of `n` and returns it as a double. In the main method with the help of Scanner class taken the input from the user and invoked the square method.

Time Complexity – $O(1)$

Space Complexity- $O(1)$

7. Find Repeated Characters in a String

Problem: Write a Java program to find all repeated characters in a string.

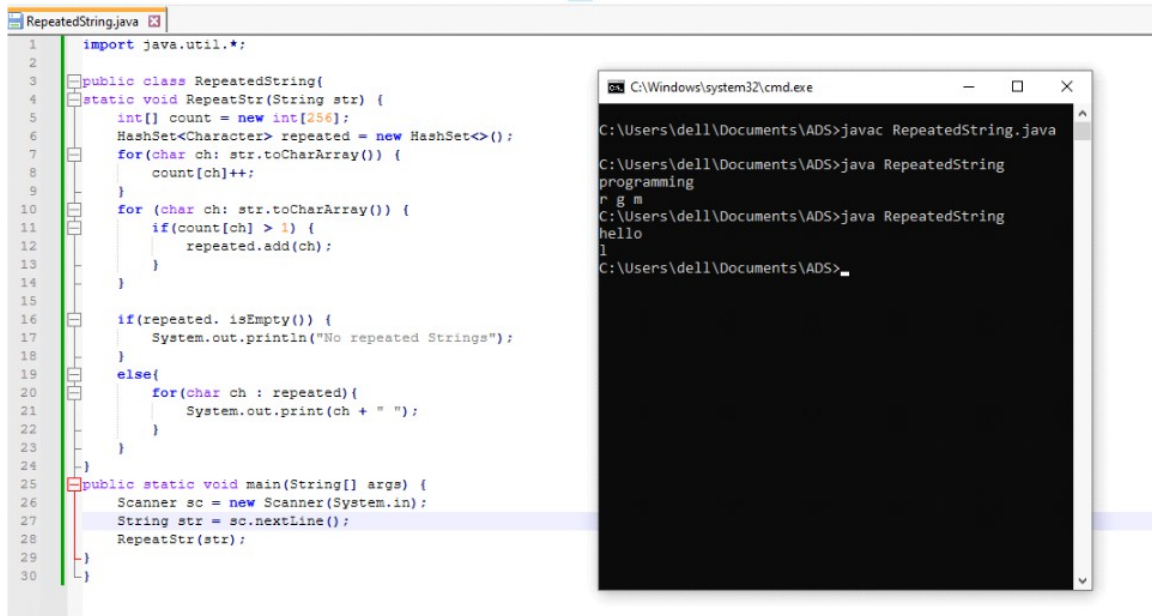
Test Cases:

Input: "programming"

Output: ['r', 'g', 'm']

Input: "hello"

Output: ['l']



```
1 import java.util.*;
2
3 public class RepeatedString{
4     static void RepeatStr(String str) {
5         int[] count = new int[256];
6         HashSet<Character> repeated = new HashSet<>();
7         for(char ch: str.toCharArray()) {
8             count[ch]++;
9         }
10        for (char ch: str.toCharArray()) {
11            if(count[ch] > 1) {
12                repeated.add(ch);
13            }
14        }
15
16        if(repeated.isEmpty()) {
17            System.out.println("No repeated Strings");
18        }
19        else{
20            for(char ch : repeated){
21                System.out.print(ch + " ");
22            }
23        }
24    }
25    public static void main(String[] args) {
26        Scanner sc = new Scanner(System.in);
27        String str = sc.nextLine();
28        RepeatStr(str);
29    }
30 }
```

```
C:\Windows\system32\cmd.exe
C:\Users\dell\Documents\ADS>javac RepeatedString.java
C:\Users\dell\Documents\ADS>java RepeatedString
programming
r g m
C:\Users\dell\Documents\ADS>java RepeatedString
hello
1
C:\Users\dell\Documents\ADS>_
```

Explanation - I have created a method called as RepeatStr of void return type. Then I initialized an array count of size 256 to store the frequency of each character in the string. . In the first for loop the string str is converted into a character array and iterates over each character ch. count[ch]++ increments the count for each character in the count array. In the second for loop it checks if the character appears more than once and adds the repeated character to the HashSet. Then if (repeated.isEmpty()) condition checks if there are no repeated characters. If true, it prints "No repeated Strings." And if repeated characters exist, the program iterates over the repeated HashSet and prints each repeated character.

Time Complexity – $O(n)$

Space Complexity- $O(n)$

8. First Non-Repeated Character

Problem: Write a Java program to find the first non-repeated character in a string.

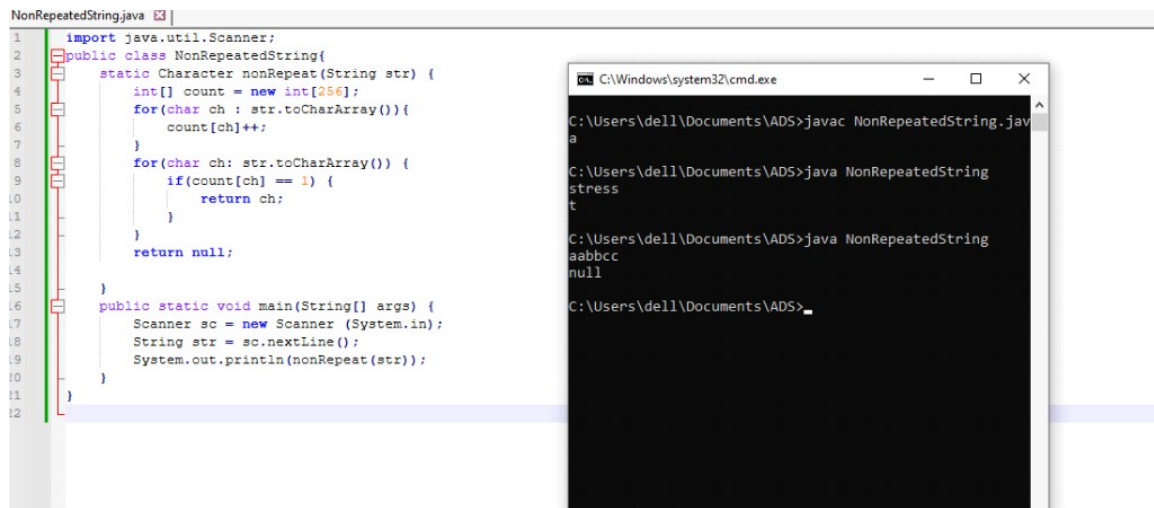
Test Cases:

Input: "stress"

Output: 't'

Input: "aabbcc"

Output: null



```
NonRepeatedString.java
1  import java.util.Scanner;
2  public class NonRepeatedString{
3      static Character nonRepeat(String str) {
4          int[] count = new int[256];
5          for(char ch : str.toCharArray()){
6              count[ch]++;
7          }
8          for(char ch: str.toCharArray()) {
9              if(count[ch] == 1) {
10                 return ch;
11             }
12         }
13         return null;
14     }
15
16     public static void main(String[] args) {
17         Scanner sc = new Scanner (System.in);
18         String str = sc.nextLine();
19         System.out.println(nonRepeat(str));
20     }
21 }
22 }
```

```
C:\Windows\system32\cmd.exe
C:\Users\dell\Documents\ADS>javac NonRepeatedString.java
C:\Users\dell\Documents\ADS>java NonRepeatedString
a
C:\Users\dell\Documents\ADS>java NonRepeatedString
stress
t
C:\Users\dell\Documents\ADS>java NonRepeatedString
aabbcc
null
C:\Users\dell\Documents\ADS>
```

Explanation- I have created a method called as nonRepeat of Character return type. Then I initialized an array count of size 256 to store the frequency of each character in the string. The size 256 is used because it covers all possible extended ASCII characters. In the first for loop the string str is converted into a character array and iterates over each character ch. count[ch]++ increments the count for each character in the count array. In the second for loop it checks if any character appeared only once and then that ch is returned. And if there are no characters that appeared only once, it will return null.

Time Complexity – $O(n)$

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



```
import java.util.Scanner;

public class Pallindrome {
    static boolean pallindrome(int n) {
        int num = n;
        int reversedNo = 0;
        while (n > 0) {
            int digit = n % 10;
            reversedNo = reversedNo * 10 + digit;
            n = n / 10;
        }
        return num == reversedNo ? true : false;
    }

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

```
C:\Windows\system32\cmd.exe
C:\Users\dell\Documents\ADS>javac Pallindrome.java
C:\Users\dell\Documents\ADS>java Pallindrome
121
true
C:\Users\dell\Documents\ADS>
```

Explanation - I have created a method pallindrome of boolean return type. Then I stored the original value of n in the variable num. Initialized reversedNo to store the reversed version of the original number. Then inside the while loop which continues as long as n is greater than 0 the last digit of n is extracted. The reversedNo is updated by shifting its digits left and adding the extracted digit. The last digit is removed from n by performing integer division by 10. Then the num is compared with reversedNo. If they are equal, true is returned; otherwise false is returned

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.

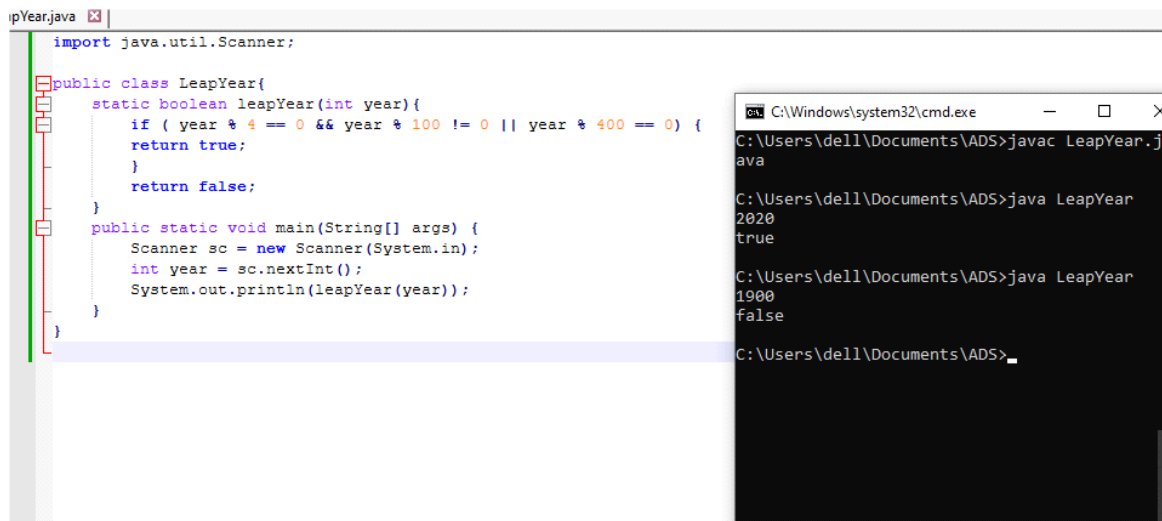
Test Cases:

Input: 2020

Output: true

Input: 1900

Output: false



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

public class LeapYear{
    static boolean leapYear(int year){
        if ( year % 4 == 0 && year % 100 != 0 || year % 400 == 0 ) {
            return true;
        }
        return false;
    }

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

C:\Windows\system32\cmd.exe
C:\Users\dell\Documents\ADS>javac LeapYear.java
C:\Users\dell\Documents\ADS>java LeapYear
2020
true
C:\Users\dell\Documents\ADS>java LeapYear
1900
false
C:\Users\dell\Documents\ADS>
```

Explanation - I created a method called as leapYear of int return type. The if condition checks if the year is divisible by 4 and also not divisible by 100 same time and at last checks if the year is divisible by 400. If any of these conditions are true, true is returned, indicating it is a leap year. Else it is non leap year.

Time Complexity – $O(1)$

Space Complexity- $O(1)$