Batch: A1 **Roll No.: 16010123012**

Experiment / assignment / tutorial No.1

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of the Staff In-charge with date

TITLE : Armstrong number

AIM:
Write a Java program to display armstrong numbers in the given range (Make use of a function).

Variations :

Implementation of Program with One class

Accessibility with static and non-static methods within class and outside class.

Expected OUTCOME of Experiment:
CO1: Apply the features of object oriented programming languages. (C++ and Java)
CO2: Explore arrays, vectors, classes and objects in C++ and Java

---

Books/ Journals/ Websites referred:

1. E. Balagurusamy, “Programming with Java”, McGraw-Hill.
2. E. Balagurusamy, “Object Oriented Programming with C++”, McGraw-Hill.

Pre Lab/ Prior Concepts:
The Scanner class is a class in java.util, which allows the user to read values of various types. There are far more methods in class Scanner than you will need in this course. We only cover a small useful subset, ones that allow us to read in numeric values from either the keyboard or file without having to convert them from strings and determine if there are more values to be read.

```
Scanner in = new Scanner(System.in);  // System.in is an InputStream
```

Numeric and String Methods

Method	Returns
int nextInt()	Returns the next token as an int. If the next token is not an integer,InputMismatchException is thrown.
long nextLong()	Returns the next token as a long. If the next token is not an integer,InputMismatchException is thrown.
float nextFloat()	Returns the next token as a float. If the next token is not a float or is out of range, InputMismatchException is thrown.
double nextDouble()	Returns the next token as a long. If the next token is not a float or is out of range, InputMismatchException is thrown.
String next()	Finds and returns the next complete token from this scanner and returns it as a string; a token is usually ended by whitespace such as a blank or line break. If not token exists,NoSuchElementException is thrown.
String nextLine()	Returns the rest of the current line, excluding any line separator at the end.
void close()	Closes the scanner.

The Scanner looks for tokens in the input. A token is a series of characters that ends with what Java calls whitespace. A whitespace character can be a blank, a tab character, a carriage return. Thus, if we read a line that has a series of numbers separated by blanks, the scanner will take each number as a separate token. .

The numeric values may all be on one line with blanks between each value or may be on separate lines. Whitespace characters (blanks or carriage returns) act as separators. The next method returns the next input value as a string, regardless of what is keyed. For example, given the following code segment and data

- int number = in.nextInt();
- float real = in.nextFloat();
- long number2 = in.nextLong();
- double real2 = in.nextDouble();
- String string = in.next();

Algorithm:

1. Input the Range, Read the lower bound and upper bound of the range.
2. Iterate through the Range
   Count the number of digits in n and store it in digits.
3. Calculate the Armstrong Sum
   While temp is not 0 -
   Extract the last digit of temp using digit = temp % 10.
   Raise the digit to the power of digits and add the result to sum.
   Remove the last digit from temp by performing integer division by 10
4. If sum equals n, then n is an Armstrong number
5. Repeat Step 3 until all numbers in the range are checked.

Implementation details:

Method 1 –

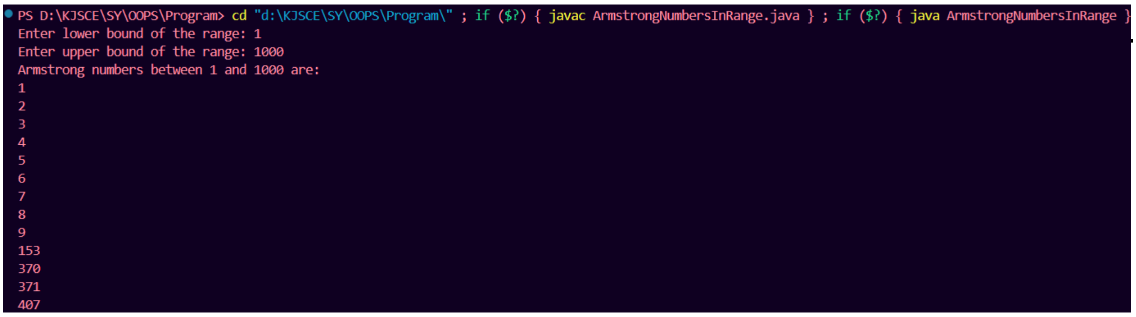
```
import java.util.Scanner;

public class ArmstrongNumbersInRange {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);



        System.out.print("Enter lower bound of the range: ");

        int lowerBound = scanner.nextInt();



        System.out.print("Enter upper bound of the range: ");

        int upperBound = scanner.nextInt();



        System.out.println("Armstrong numbers between " + lowerBound + " and " + upperBound + " are:");
        printArmstrongNumbersInRange(lowerBound, upperBound);
    }
    public static void printArmstrongNumbersInRange(int lowerBound, int upperBound) {
        for (int i = lowerBound; i <= upperBound; i++) {
            if (isArmstrong(i)) {
                System.out.println(i);
            }
        }
    }
    public static boolean isArmstrong(int number) {
        int originalNumber = number;
        int sum = 0;
        int digits = String.valueOf(number).length();
        while (number != 0) {
            int digit = number % 10;
            sum += Math.pow(digit, digits);
            number /= 10;
        }
        return sum == originalNumber;
    }
}
```

Method 2 -

```
import java.util.Scanner;

public class ArmstrongNumbers {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter lower bound of the range: ");

        int lowerBound = scanner.nextInt();

        System.out.print("Enter upper bound of the range: ");

        int upperBound = scanner.nextInt();

        System.out.println("Armstrong numbers between " + lowerBound + " and " + upperBound + " are:");

        for (int i = lowerBound; i <= upperBound; i++) {

            if (isArmstrong(i)) {

                System.out.println(i);

            }

        }

    }

    public static boolean isArmstrong(int number) {

        int originalNumber = number;

        int sum = 0;
        int digits = String.valueOf(number).length();

        while (number != 0) {

            int digit = number % 10;

            sum += Math.pow(digit, digits);

            number /= 10;

        }

        return sum == originalNumber;

    }

}
```

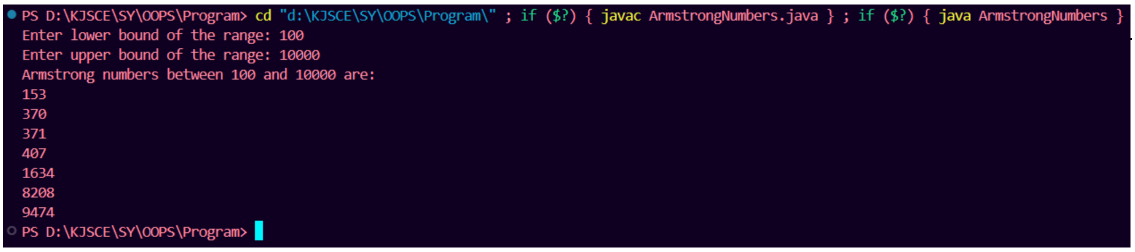
Output:

Method 1 –



```
PS D:\KJSCE\SY\OOPS\Program> cd "d:\KJSCE\SY\OOPS\Program\" ; if ($?) { javac ArmstrongNumbersInRange.java } ; if ($?) { java ArmstrongNumbersInRange }
Enter lower bound of the range: 1
Enter upper bound of the range: 1000
Armstrong numbers between 1 and 1000 are:
1
2
3
4
5
6
7
8
9
153
370
371
407
```

Method 2 –



```
PS D:\KJSCE\SY\OOPS\Program> cd "d:\KJSCE\SY\OOPS\Program\" ; if ($?) { javac ArmstrongNumbers.java } ; if ($?) { java ArmstrongNumbers }
Enter lower bound of the range: 100
Enter upper bound of the range: 10000
Armstrong numbers between 100 and 10000 are:
153
370
371
407
1634
8208
9474
PS D:\KJSCE\SY\OOPS\Program>
```

Conclusion:

Learnt how write a java program to display armstrong numbers in the given range by using different methods.

Date: 04/08/2024 **Signature of faculty in-charge**

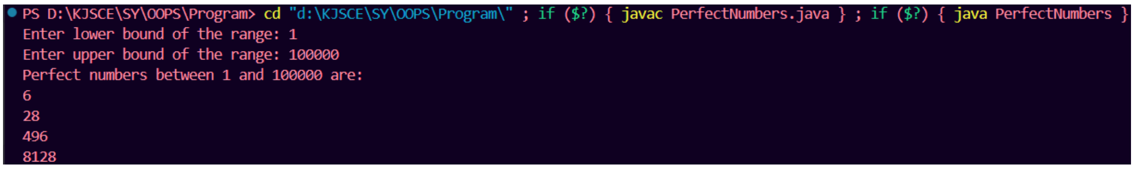
Post Lab Descriptive Questions:

Q.1 Write a program to find the perfect numbers between the range.
Q.2 Write a program to check whether the entered year is a leap year or not.
Q.3 Write a program to find gcd and lcm of two numbers (find gcd using recursive function).

Output:

Q1 -

```
import java.util.Scanner;

public class PerfectNumbers {

  public static void main(String[] args) {

    Scanner scanner = new Scanner(System.in);



    System.out.print("Enter lower bound of the range: ");

    int lowerBound = scanner.nextInt();



    System.out.print("Enter upper bound of the range: ");

    int upperBound = scanner.nextInt();



    System.out.println("Perfect numbers between " + lowerBound + " and " + upperBound + "
are:");

    PerfectNumbersInRange(lowerBound, upperBound);
  }
  public static void PerfectNumbersInRange(int lowerBound, int upperBound) {
    for(int number = lowerBound; number <= upperBound; number++) {
        if (isPerfectNumber(number)){
            System.out.println(number);
        }
    }
  }
  public static boolean isPerfectNumber(int number) {
    int sum = 0;
    for(int i = 1; i <= number / 2; i++) {
        if(number % i == 0) {
            sum += i;
        }
    }
    return sum == number;
  }
}
```

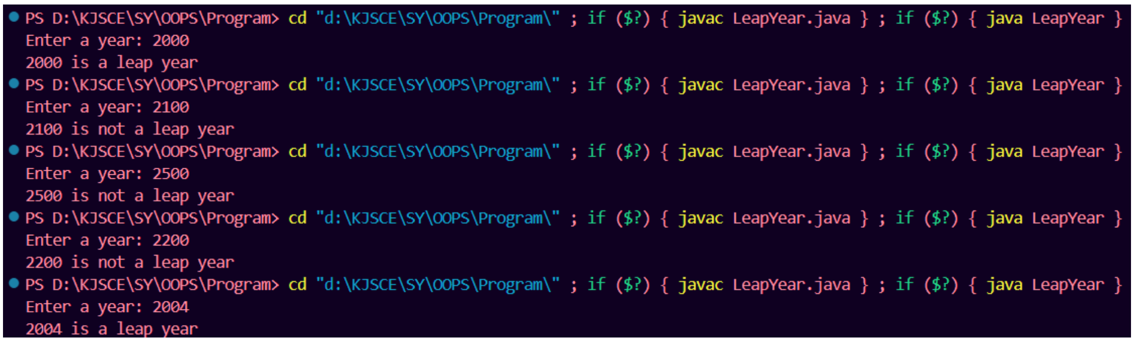


```
PS D:\KJSCE\SY\OOPS\Program> cd "d:\KJSCE\SY\OOPS\Program\" ; if ($?) { javac PerfectNumbers.java } ; if ($?) { java PerfectNumbers }
Enter lower bound of the range: 1
Enter upper bound of the range: 100000
Perfect numbers between 1 and 100000 are:
6
28
496
8128
```

Q2 –

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

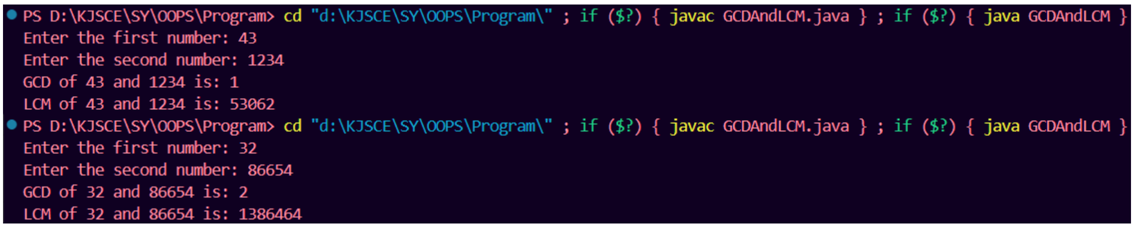
```
public class LeapYear {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter a year: ");
    int year = scanner.nextInt();
    if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)) {
      System.out.println(year + " is a leap year");
  } else {
      System.out.println(year + " is not a leap year");
  }
  scanner.close();
  }
}
```



```
PS D:\KJSCE\SY\OOPS\Program> cd "d:\KJSCE\SY\OOPS\Program\" ; if ($?) { javac LeapYear.java } ; if ($?) { java LeapYear }
Enter a year: 2000
2000 is a leap year
PS D:\KJSCE\SY\OOPS\Program> cd "d:\KJSCE\SY\OOPS\Program\" ; if ($?) { javac LeapYear.java } ; if ($?) { java LeapYear }
Enter a year: 2100
2100 is not a leap year
PS D:\KJSCE\SY\OOPS\Program> cd "d:\KJSCE\SY\OOPS\Program\" ; if ($?) { javac LeapYear.java } ; if ($?) { java LeapYear }
Enter a year: 2500
2500 is not a leap year
PS D:\KJSCE\SY\OOPS\Program> cd "d:\KJSCE\SY\OOPS\Program\" ; if ($?) { javac LeapYear.java } ; if ($?) { java LeapYear }
Enter a year: 2200
2200 is not a leap year
PS D:\KJSCE\SY\OOPS\Program> cd "d:\KJSCE\SY\OOPS\Program\" ; if ($?) { javac LeapYear.java } ; if ($?) { java LeapYear }
Enter a year: 2004
2004 is a leap year
```

Q3 –

```
import java.util.Scanner;

public class GCDAndLCM {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the first number: ");
        int num1 = scanner.nextInt();
        System.out.print("Enter the second number: ");
        int num2 = scanner.nextInt();

        int gcd = findGCD(num1, num2);
        System.out.println("GCD of " + num1 + " and " + num2 + " is: " + gcd);

        int lcm = (num1 * num2) / gcd;
        System.out.println("LCM of " + num1 + " and " + num2 + " is: " + lcm);
    }
    public static int findGCD(int a, int b) {
        if (b == 0) {
            return a;
        }
        return findGCD(b, a % b);
    }
}
```



```
PS D:\KJSCE\SY\OOPS\Program> cd "d:\KJSCE\SY\OOPS\Program\" ; if ($?) { javac GCDAndLCM.java } ; if ($?) { java GCDAndLCM }
Enter the first number: 43
Enter the second number: 1234
GCD of 43 and 1234 is: 1
LCM of 43 and 1234 is: 53062
PS D:\KJSCE\SY\OOPS\Program> cd "d:\KJSCE\SY\OOPS\Program\" ; if ($?) { javac GCDAndLCM.java } ; if ($?) { java GCDAndLCM }
Enter the first number: 32
Enter the second number: 86654
GCD of 32 and 86654 is: 2
LCM of 32 and 86654 is: 1386464
```