K. J. Somaiya College of Engineering, Mumbai-77
(A Constituent College of Somaiya Vidyavihar University)
Department of Computer Engineering

Name: Shreyans Tatiya **Batch: E2 Roll No.: 16010123325** **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:

Input:

- low and high values from the user to define the range of numbers to check.

isArmstrong Function:

- Initialize temp to num to preserve the original value.
- Initialize sum to 0 to store the sum of the digits raised to the power of the number of digits.
- Calculate the number of digits digits in num using String.valueOf(num).length().
- Loop until temp is 0:
- Calculate the last digit ld of temp using temp % 10.
- Add ld raised to the power of digits to sum using Math.pow(ld, digits).
- Remove the last digit from temp using temp /= 10.
- Return true if sum is equal to the original num, otherwise return false.

displayArmstrong Function:

- Loop from low to high (inclusive):
- Check if the current number i is an Armstrong number using isArmstrong(i).
- If it is, print i followed by a space.

Main Function:

- Create a Scanner object to read input from the user.
- Prompt the user to enter the start and end of the range.
- Read the input values and store them in low and high.
- Call displayArmstrong(low, high) to display the Armstrong numbers in the range.
- Close the Scanner object.

Implementation details:

```
// All Methods are static

import java.util.Scanner;

public class ArmstrongNumbers {
    public static boolean isArmstrong(int num) {
        int temp = num;
        int sum = 0;
        int digits = String.valueOf(num).length();
        while (temp > 0) {
            int ld = temp % 10;
            sum += Math.pow(ld, digits);
            temp /= 10;
        }
        return sum == num;
    }

    public static void displayArmstrong(int low, int high) {
        for (int i = low; i <= high; ++i) {
            if (isArmstrong(i)) {
                System.out.print(i + " ");
            }
        }
        System.out.println();
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the start of the range: ");
        int low = scanner.nextInt();
        System.out.print("Enter the end of the range: ");
        int high = scanner.nextInt();
```

```
        displayArmstrong(low, high);
        scanner.close();
    }
}



// Static and Non-Static methods within the same class

public class ArmstrongNumber {

    // Static Method to check if a number is Armstrong Number
    public static boolean isArmstrongNumber(int number) {
        int sum = 0;
        int temp = number;
        while (temp != 0) {
            int remainder = temp % 10;
            sum += Math.pow(remainder, 3);
            temp /= 10;
        }
        return sum == number;
    }

    // Non-Static method to display Armstrong Numbers in a range
    public void display(int start, int end) {
        for (int i = start; i <= end; i++) {
            if (isArmstrongNumber(i)) {
                System.out.println(i + " ");
            }
        }
        System.out.println();
    }

    public static void main(String[] args) {
        ArmstrongNumber armstrongNumber = new ArmstrongNumber();
        armstrongNumber.display(1, 10000);
    }
```

```java
}
```

ArmstrongNumber.java U ArmstrongNumberTest.java U

Programs > ArmstrongNumberTest.java > ArmstrongNumberTest

```java
// Static and Non-Static methods accessed from another clas
public class ArmstrongNumberTest {

    Run | Debug
    public static void main(String[] args) {
        int start = 1;
        int end = 1000;

        System.out.println(x:"listing static method from outside the class:");
        for (int i = start; i <= end; ++i) {
            if (ArmstrongNumber.isArmstrongNumber(i)) {
                System.out.print(i + " ");
            }
        }
        System.out.println();


        // Using non-static method from outside the class
        ArmstrongNumber armstrongNumber = new ArmstrongNumber();
        System.out.println(x:"Using non-static method from outside the class");
        armstrongNumber.display(start, end);
    }
}
```

Output:

```
java -cp /tmp/kwPqBiGKNy/ArmstrongNumbers
Enter the start of the range: 0
Enter the end of the range: 1000
0 1 2 3 4 5 6 7 8 9 153 370 371 407

=== Code Execution Successful ===
```

```
java -cp /tmp/qSeiJt4DnP/ArmstrongNumbers
Enter the start of the range: 0
Enter the end of the range: 10000
0 1 2 3 4 5 6 7 8 9 153 370 371 407 1634 8208 9474
```

Conclusion:

In conclusion, the Armstrong Numbers program successfully identifies and displays all Armstrong numbers within a user-defined range, providing an efficient and accurate solution for this mathematical problem.

Date: _____ **Signature of faculty in-charge**

Post Lab Descriptive Questions:

Q.1 Write a program to find the perfect numbers between the range.
Ans:

```
import java.util.Scanner;

public class PerfectNumbers {
    public static boolean isPerfect(int num) {
        int sum = 0;
        for (int i = 1; i < num; ++i) {
            if (num % i == 0) {
                sum += i;
            }
        }
        return sum == num;
    }

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

```
        System.out.println("Enter the lower limit of the range:");
        int lowerLimit = scanner.nextInt();
        System.out.println("Enter the upper limit of the range:");
        int upperLimit = scanner.nextInt();

        System.out.println("Perfect numbers between " + lowerLimit + "
and " + upperLimit + " are:");
        for (int i = lowerLimit; i <= upperLimit; i++) {
            if (isPerfect(i)) {
                System.out.println(i);
            }
        }
    }
}
```

Q.2 Write a program to check whether the entered year is a leap year or not.
Ans:

```
import java.util.Scanner;

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

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter a year:");
        int year = scanner.nextInt();
```

```
        if (isLeapYear(year)) {
            System.out.println(year + " is a leap year.");
        } else {
            System.out.println(year + " is not a leap year.");
        }
    }
}
```

Q.3 Write a program to find gcd and lcm of two numbers(find gcd using recursive function).

Ans:

```
import java.util.Scanner;

public class GCDLCM {
    public static int findGCD(int num1, int num2) {
        if (num2 == 0) {
            return num1;
        } else {
            return findGCD(num2, num1 % num2);
        }
    }

    public static int findLCM(int num1, int num2, int gcd) {
        return (num1 * num2) / gcd;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the first number:");
        int num1 = scanner.nextInt();
        System.out.println("Enter the second number:");
        int num2 = scanner.nextInt();

        int gcd = findGCD(num1, num2);
        int lcm = findLCM(num1, num2, gcd);

        System.out.println("GCD of " + num1 + " and " + num2 + " is " +
gcd);
```

```
        System.out.println("LCM of " + num1 + " and " + num2 + " is " +
lcm);
    }
}
```

Output:

Q1)

```
java -cp /tmp/9BD4rldCUg/PerfectNumbers
Enter the lower limit of the range:
0
Enter the upper limit of the range:
1000
Perfect numbers between 0 and 1000 are:
0
6
28
496
```

```
java -cp /tmp/81si96N1wI/PerfectNumbers
Enter the lower limit of the range:
0
Enter the upper limit of the range:
10000
Perfect numbers between 0 and 10000 are:
0
6
28
496
8128

=== Code Execution Successful ===
```

Q2)

```
java -cp /tmp/ozymCz23BI/LeapYear
Enter a year:
2016
2016 is a leap year.

=== Code Execution Successful ===
```

```
java -cp /tmp/NL7m3C1iDU/LeapYear
Enter a year:
1982
1982 is not a leap year.

=== Code Execution Successful ===
```

Q3)

```
java -cp /tmp/2QkvF45wsA/GCDLCM
Enter the first number:
10
Enter the second number:
34
GCD of 10 and 34 is 2
LCM of 10 and 34 is 170

=== Code Execution Successful ===
```

```
java -cp /tmp/QwH9x4BIfe/GCDLCM
Enter the first number:
1989
Enter the second number:
3891
GCD of 1989 and 3891 is 3
LCM of 1989 and 3891 is 2579733

=== Code Execution Successful ===
```