Batch: A1	**Roll No.: 16010123012**
Experiment / assignment / tutorial No. 2	
Grade: AA / AB / BB / BC / CC / CD /DD	
Signature of the Staff In-charge with date	

TITLE : Control Statements

AIM:

Write a Java program to generate and show all Kaprekar numbers less than 1000.
In number theory, a Kaprekar number for a given base is a non-negative integer, the representation of whose square in that base can be split into two parts that add up to the original number again. For instance, 45 is a Kaprekar number, because 45^2 = 2025 and 20 + 25 = 45.

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:

Java basic constructs (like if else statement, control structures, and data types
Programming languages provide various control structures that allow for more complicated execution paths.

A loop statement allows us to execute a statement or group of statements multiple times and following is the general form of a loop statement in most of the programming languages −

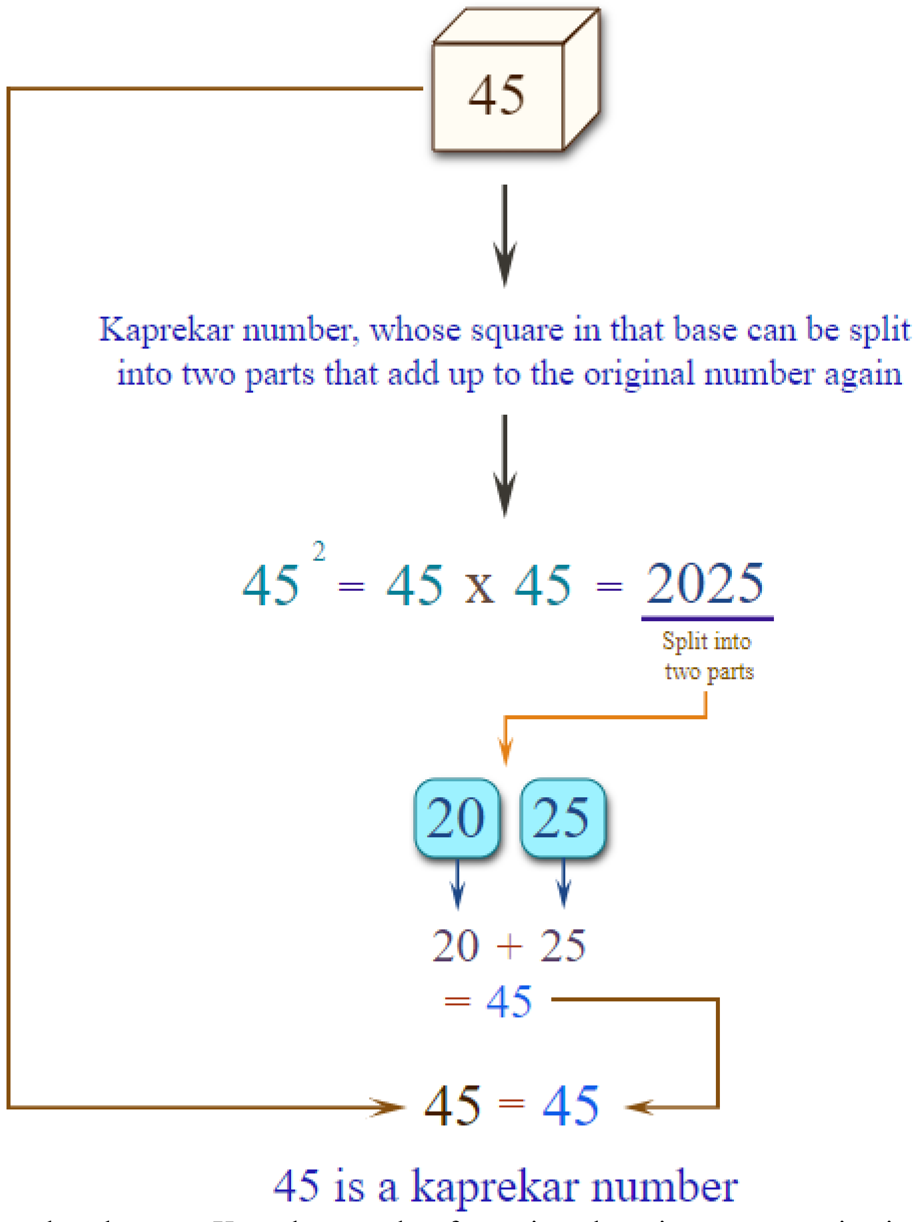
Sr.No.	Loop & Description
1	**while loop** Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.
2	**for loop** Execute a sequence of statements multiple times and abbreviates the code that manages the loop variable.
3	**do...while loop** Like a while statement, except that it tests the condition at the end of the loop body.

Loop Control Statements

Loop control statements change execution from its normal sequence. When execution leaves a scope, all automatic objects that were created in that scope are destroyed.

Java supports the following control statements. Click the following links to check their details.

Sr.No.	Control Statement & Description
1	**break statement** Terminates the loop or switch statement and transfers execution to the statement immediately following the loop or switch.
2	**continue statement** Causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.


45
Kaprekar number, whose square in that base can be split into two parts that add up to the original number again
45² = 45 x 45 = 2025
Split into two parts
20
25
20 + 25
= 45
45 = 45
45 is a kaprekar number

In number theory, a Kaprekar number for a given base is a non-negative integer, the representation of whose square in that base can be split into two parts that add up to the original number again. For instance, 45 is a Kaprekar number, because 452 = 2025 and 20 + 25 = 45.

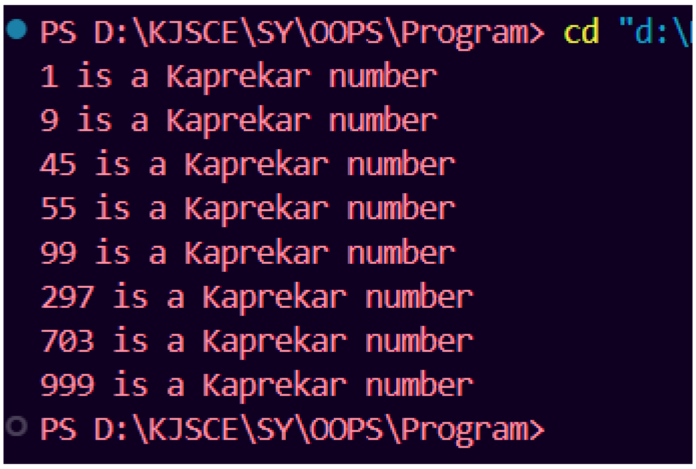
Algorithm:

1. Initialize Loop
2. Square the Number
3. Count Digits in Square
4. Reset Square
5. Determine Split Position
6. Calculate Divisor
7. Split the Square
8. Check Kaprekar Condition
9. Print if Kaprekar
10. Repeat for All Numbers

Implementation details:

```
public class KaprekarNumber {

    public static void main(String[] args) {

        for (int num = 1; num <= 1000; num++) {

            int len = 0;

            int temp = num * num;

            int lnum = 0, rnum = 0, d = 1;



            while (temp > 0) {

                temp = temp / 10;

                len++;

            }

            temp = num * num;

            int split;
            if (len % 2 == 0) {

                split = len / 2;

            } else {

                split = (len + 1) / 2;

            }

            while (split > 0) {

                d = d * 10;

                split--;

            }

            lnum = temp / d;

            rnum = temp % d;


            if ((lnum + rnum) == num) {

                System.out.println(num + " is a Kaprekar number");

            }

        }

    }

}
```

Output:



```
PS D:\KJSCE\SY\OOPS\Program> cd "d:\
1 is a Kaprekar number
9 is a Kaprekar number
45 is a Kaprekar number
55 is a Kaprekar number
99 is a Kaprekar number
297 is a Kaprekar number
703 is a Kaprekar number
999 is a Kaprekar number
PS D:\KJSCE\SY\OOPS\Program>
```

Conclusion:

We learned about Kaprekar numbers and wrote the code to find Kaprekar numbers in a given range. The above method efficiently finds and displays all Kaprekar numbers in the given range.

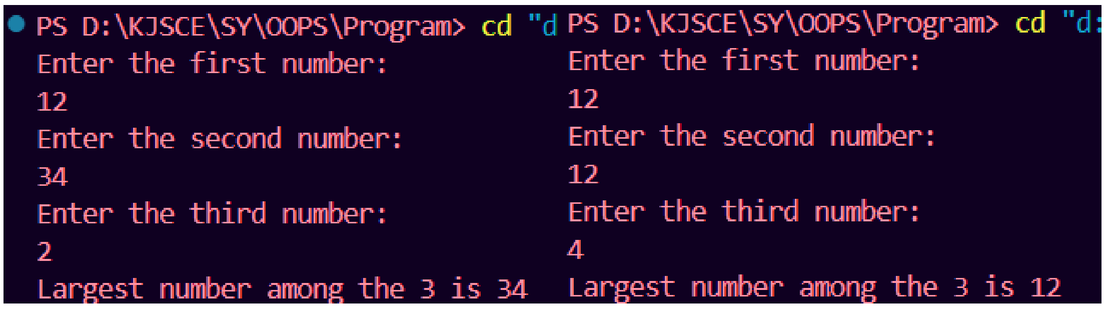
Date: 06/08/2024 **Signature of faculty in-charge**

<u>Post Lab Descriptive Questions:</u>

Q.1 Write a program to find the largest of three numbers using the if-else construct.
Q.2 Write a program to determine the sum of the following series for a given value of n:1+½+⅓+....+1/n

Output:
Q1

```
import java.util.Scanner;


public class Largest {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    int num1, num2, num3, largest;
    System.out.println("Enter the first number:");
    num1 = scanner.nextInt();
    System.out.println("Enter the second number:");
    num2 = scanner.nextInt();
    System.out.println("Enter the third number:");
    num3 = scanner.nextInt();

    if (num1 >= num2 && num1 >= num3) {
      largest = num1;
    } else if(num2 >= num1 && num2 >= num3) {
      largest = num2;
    } else {
      largest = num3;
    }
    System.out.println("Largest number among the 3 is " + largest);
    scanner.close();
  }
}
```



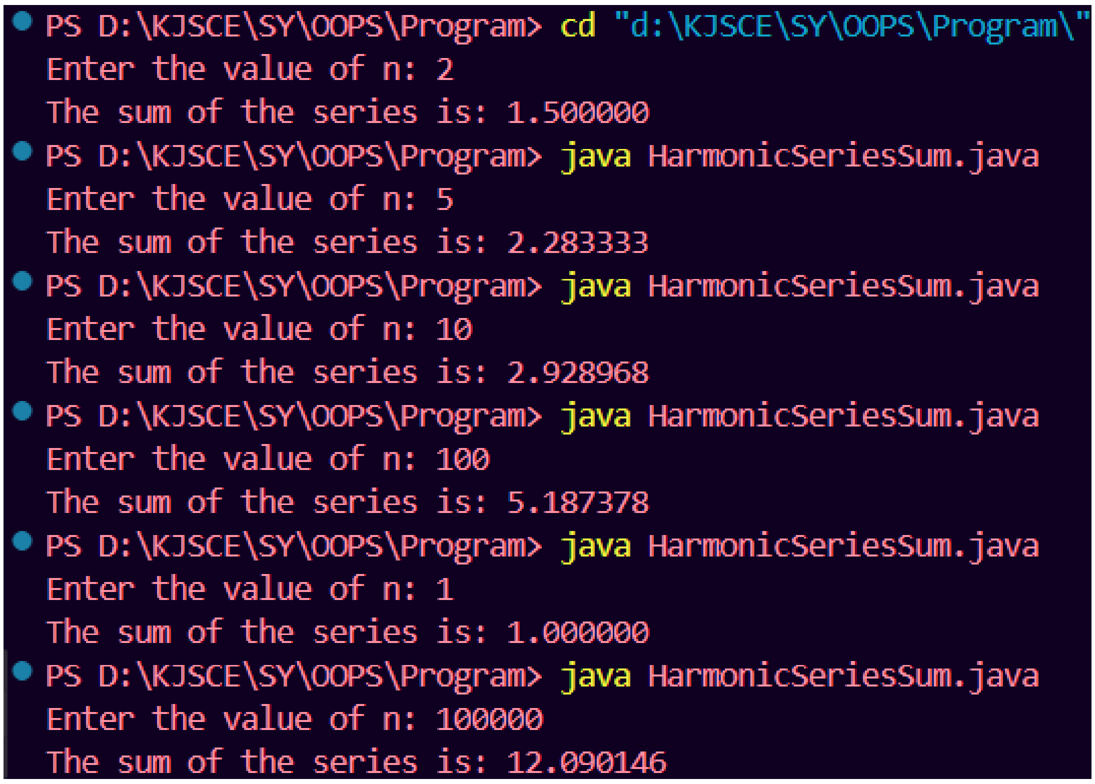
```
PS D:\KJSCE\SY\OOPS\Program> cd "d
Enter the first number:
12
Enter the second number:
34
Enter the third number:
2
Largest number among the 3 is 34
```

```
PS D:\KJSCE\SY\OOPS\Program> cd "d:
Enter the first number:
12
Enter the second number:
12
Enter the third number:
4
Largest number among the 3 is 12
```

Q2

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

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

        System.out.print("Enter the value of n: ");
        int n = scanner.nextInt();

        double sum = 0.0;
        for (int i = 1; i <= n; i++) {
            sum += 1.0 / i;
        }
        System.out.printf("The sum of the series is: %f", sum);
    }
}
```



```
PS D:\KJSCE\SY\OOPS\Program> cd "d:\KJSCE\SY\OOPS\Program\"
Enter the value of n: 2
The sum of the series is: 1.500000
PS D:\KJSCE\SY\OOPS\Program> java HarmonicSeriesSum.java
Enter the value of n: 5
The sum of the series is: 2.283333
PS D:\KJSCE\SY\OOPS\Program> java HarmonicSeriesSum.java
Enter the value of n: 10
The sum of the series is: 2.928968
PS D:\KJSCE\SY\OOPS\Program> java HarmonicSeriesSum.java
Enter the value of n: 100
The sum of the series is: 5.187378
PS D:\KJSCE\SY\OOPS\Program> java HarmonicSeriesSum.java
Enter the value of n: 1
The sum of the series is: 1.000000
PS D:\KJSCE\SY\OOPS\Program> java HarmonicSeriesSum.java
Enter the value of n: 100000
The sum of the series is: 12.090146
```