Batch: E2	**Roll No.: 16010123325**
Experiment / assignment / tutorial No.	
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

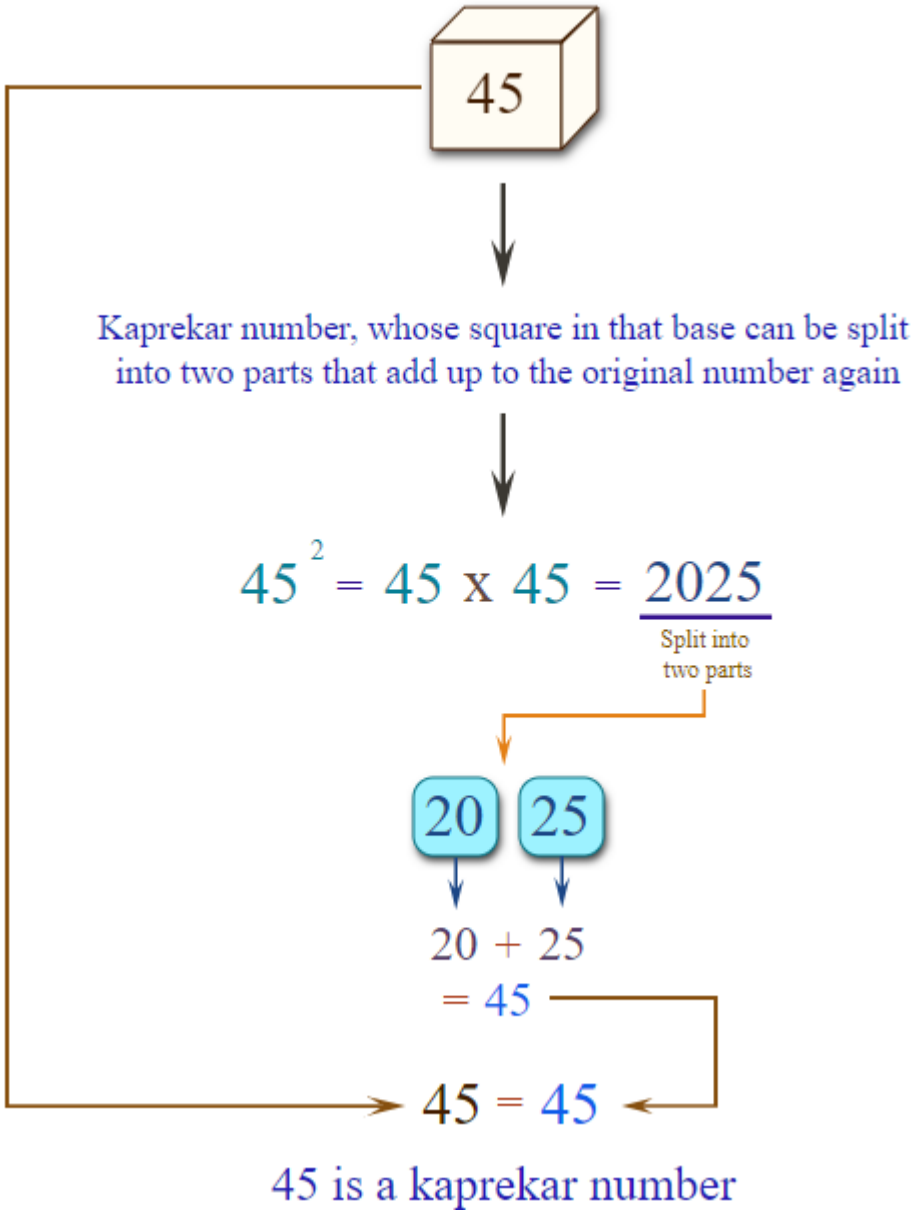
1	**while loop** Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.
2	**for loop** Execute a sequence of statements multiple times and abbreviates the code that manages the loop variable.
3	**do...while loop** Like a while 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.



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:

Kaprekar Number Algorithm:

1. Calculate the square of the input number n.
2. Count the number of digits in the square.

3. Iterate through all possible splits of the square into two parts, where the right part has i digits (from 1 to n-1).
4. For each split, calculate the left part by dividing the square by 10^i.
5. For each split, calculate the right part by taking the remainder of the square modulo 10^i.
6. For each split, calculate the sum of the left and right parts.
7. Check if the sum equals the original number n.
8. If the sum equals n, return true.
9. If no split results in a sum equal to n, return false.
10. Handle the base case where n is 1, in which case return true immediately.

Implementation details:

```
public class KaprekarNumber {
    static boolean kaprekar(int n) {
        if (n == 1) return true;
        int square = n * n;
        int cnt = 0;
        while (square != 0) {
            cnt++;
            square /= 10;
        }
        square = n * n;
        for (int i = 1; i < n; ++i) {
            int eq_part = (int)Math.pow(10, i);
            if (eq_part == n) continue;
            int sum = square/eq_part + square % eq_part;
            if (sum == n) return true;
        }
        return false;
    }

    public static void main(String[] args) {
        for (int i = 1; i < 1000; ++i) {
            if (kaprekar(i)) {
                System.out.print(i + " ");
            }
        }
    }
}
```

Output:

```
java -cp /tmp/Hz7zRPN3EP/KaprekarNumber
1
9
45
55
99
297
703
999

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

```
java -cp /tmp/lThoGB0vJh/KaprekarNumber
1 9 45 55 99 297 703 999
=== Code Execution Successful ===
```

Conclusion: The provided Java code is designed to find and print all Kaprekar numbers less than 1000. A Kaprekar number is a number whose square, when divided into two parts and added together, equals the original number

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

Post Lab Descriptive Questions:

Q.1 Write a program to find the largest of three numbers using the if-else construct.
Ans:

```java
import java.util.Scanner;

public class LargestNumber {
    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();
        System.out.print("Enter the third number: ");
        int num3 = scanner.nextInt();

        if (num1 >= num2 && num1 >= num3) {
            System.out.println("The largest number is " + num1);
        }
```

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

Q.2 Write a program to determine the sum of the following series for a given value of n:1+½+⅓+....+1/n

Ans:

```
import java.util.Scanner;

public class SeriesSum {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter value of n: ");
        int n = scanner.nextInt();
        scanner.close();

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

Output:

Q1)

```
java -cp /tmp/hNlIY3yjll/LargestNumber
Enter the first number: 10
Enter the second number: 3
Enter the third number: 1000
The largest number is 1000

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

```
java -cp /tmp/gEDpDBi7Vi/LargestNumber
Enter the first number: 90
Enter the second number: 83
Enter the third number: 78
The largest number is 90

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

Q2)

```
java -cp /tmp/iRD7LHMbrh/SeriesSum
Enter value of n: 10
The sum of the series is: 2.9289682539682538

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

```
java -cp /tmp/VsnSiUNKEh/SeriesSum
Enter value of n: 80
The sum of the series is: 4.965479278945517

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