

**Batch: D3                      Roll No.: 16010123294**

**Experiment / assignment / tutorial No. 02**

**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 –

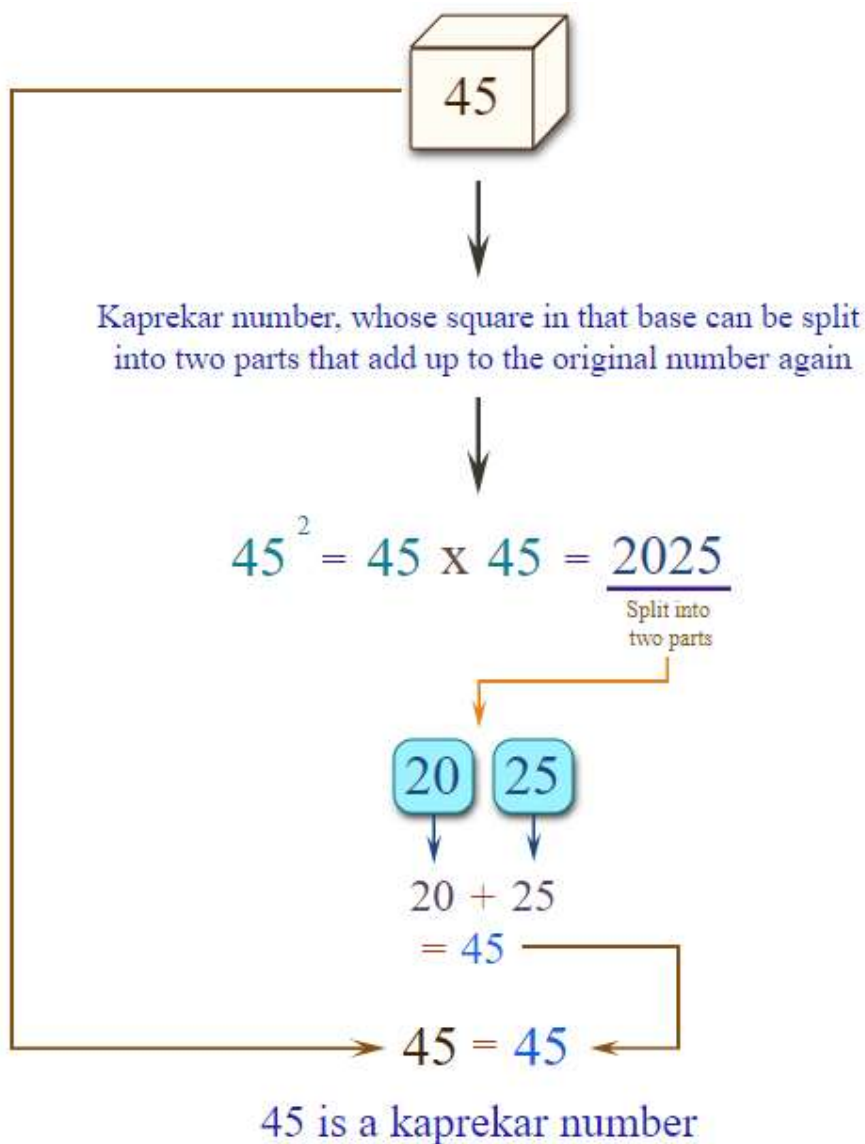
Sr.No.	Loop & Description
1	<a href="#"><u>while loop</u></a> Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.
2	<a href="#"><u>for loop</u></a> Execute a sequence of statements multiple times and abbreviates the code that manages the loop variable.
3	<a href="#"><u>do...while loop</u></a> 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	<a href="#"><u>break statement</u></a> Terminates the loop or switch statement and transfers execution to the statement immediately following the loop or switch.
2	<a href="#"><u>continue statement</u></a> 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  $45^2 = 2025$  and  $20 + 25 = 45$ .

### **Algorithm:**

To find all Kaprekar numbers less than 1000, start by iterating through each number from 1 to 999.

For each number, calculate its square and convert this square into a string.

Then, split this string into two parts at every possible position.

Convert the left and right parts back to integers and check if the sum of these two integers equals the original number, ensuring the right part is greater than zero.

If this condition is met, the number is a Kaprekar number. Additionally, account for the special case where the number is 1, which is inherently a Kaprekar number.

Print all numbers that satisfy these conditions.

### **Implementation details:**

```
public class KaprekarNumbers {  
  
    public static void main(String[] args) {  
  
        System.out.println("Kaprekar numbers less than 1000 are:");  
  
        for (int i = 1; i < 1000; i++) {  
  
            if (isKaprekar(i)) {  
  
                System.out.println(i);  
  
            }  
  
        }  
  
    }  
  
    public static boolean isKaprekar(int number) {  
  
        int square = number * number;
```

```
String squareStr = Integer.toString(square);

int len = squareStr.length();

for (int i = 1; i < len; i++) {

    int leftPart = Integer.parseInt(squareStr.substring(0, i));
    int rightPart = Integer.parseInt(squareStr.substring(i));

    if (rightPart > 0 && leftPart + rightPart == number) {
        return true;
    }
}

return number == 1; // Special case for 1, as 1^2=1 and needs to be included.
}
}
```

**Output:**

```
C:\Users\Saish\OneDrive\Desktop\C\java>java kaprekarNumbers.java
Kaprekar numbers less than 1000 are:
1
9
45
55
99
297
703
999
```

**Conclusion:**

**Learned to use methods to carry out tasks and applied the knowledge of the sum in the codes**

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

```
public class largest {  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
  
        // Prompt the user to enter three numbers  
        System.out.println("Enter the first number:");  
        int num1 = scanner.nextInt();  
  
        System.out.println("Enter the second number:");  
        int num2 = scanner.nextInt();  
  
        System.out.println("Enter the third number:");  
        int num3 = scanner.nextInt();  
  
        // Determine the largest number using if-else construct  
        int largest;  
  
        if (num1 >= num2 && num1 >= num3) {  
            largest = num1;  
        } else if (num2 >= num1 && num2 >= num3) {  
            largest = num2;  
        } else {  
            largest = num3;  
        }  
    }  
}
```

```
// Display the largest number
System.out.println("The largest number is: " + largest);

scanner.close();
}
}
```

Q.2 Write a program to determine the sum of the following series for a given value of n:  $1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}$

```
import java.util.Scanner;

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

        // Prompt the user to enter the value of n
        System.out.println("Enter the value of n:");
        int n = scanner.nextInt();

        // Initialize the sum to 0
        double sum = 0.0;

        // Compute the sum of the series
        for (int i = 1; i <= n; i++) {
            sum += 1.0 / i;
        }

        // Display the result
        System.out.println("The sum of the series is: " + sum);

        scanner.close();
    }
}
```

**Output:**

```
C:\Users\Saish\OneDrive\Desktop\C\java>java largest.java
Enter the first number:
20
Enter the second number:
50
Enter the third number:
10
The largest number is: 50

C:\Users\Saish\OneDrive\Desktop\C\java>
```

```
C:\Users\Saish\OneDrive\Desktop\C\java>java series.java
Enter the value of n:
5
The sum of the series is: 2.2833333333333333

C:\Users\Saish\OneDrive\Desktop\C\java>
```