Experiment No.1
Basic programming constructs like branching and looping
Date of Performance:
Date of Submission:

Aim :- To apply programming constructs of decision making and looping.

Objective :- To apply basic programming constructs like Branching and Looping for solving arithmetic problems like calculating factorial of a no entered by user at command prompt .

Theory :-

Programming constructs are basic building blocks that can be used to control computer programs. Most programs are built out of a fairly standard set of programming constructs. For example, to write a useful program, we need to be able to store values in variables, test these values against a condition, or loop through a set of instructions a certain number of times. Some of the basic program constructs include decision making and looping.

Decision Making in programming is similar to decision making in real life. In programming also, we face some situations where we want a certain block of code to be executed when some condition is fulfilled. A programming language uses control statements to control the flow of execution of a program based on certain conditions. These are used to cause the flow of execution to advance, and branch based on changes to the state of a program.

- if
- if-else
- nested-if
- if-else-if
- switch-case
- break, continue

These statements allow you to control the flow of your program's execution based upon conditions known only during run time.

A loop is a programming structure that repeats a sequence of instructions until a specific condition is met. Programmers use loops to cycle through values, add sums of numbers, repeat functions, and many other things. ... Two of the most common types of loops are the while loop and the for loop.

The different ways of looping in programming languages are

- while
- do-while

- for loop
- Some languages have modified for loops for more convenience eg :- Modified for loop in java.

For and while loop is entry-controlled loops. Do-while is an exit-controlled loop.

Code: -

- **If Condition**

```
import java.util.Scanner;

    class PositiveCheck

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

                         System.out.print("Enter a number: ");
                          int number = scanner.nextInt();

                         if (number > 0)
                         {
                        System.out.println("The number is positive.");
                         }

             }
   }
```

- **If – else Condition**

```
class Conditional
{
        public static void main (String args[])
        {
                int a=10;
                if(a<15)
                {
                        System.out.println("a is less number than ");
                }
                else
                {
                        System.out.println("a is not less than 15");
                }
        }
}
```

- **Nested – if / if-else-if**

```
        import java.util.Scanner;

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

```
              System.out.print("Enter a number: ");
             int number = scanner.nextInt();

               if (number > 0)
              {
                        System.out.println("The number is positive.");
               }
             else if (number < 0)
              {
                           System.out.println("The number is negative.");
             }
             else
             {
                        System.out.println("The number is zero.");
              }

        }
}
```

- **Switch , Continue , Break , While**

```
import java.util.Scanner;

public class SwitchExample
{
          public static void main(String[] args)
          {
           Scanner scanner = new Scanner(System.in);
          while (true)
          {
            System.out.print("Enter a number (1-7) for the day of the week (0 to exit): ");
            int day = scanner.nextInt();

            if (day == 0)
             {
                System.out.println("Exiting...");
                break;
            }

            switch (day)
            {
                case 1:
                case 2:
                case 3:
                case 4:
                case 5:
                    System.out.println("It's a weekday.");
                    break;
                case 6:
                case 7:
                    System.out.println("It's the weekend.");
                    break;
                default:
                    System.out.println("Invalid input, please enter a number between 1 and 7.");
                    continue;
            }
        }



    }
}
```

- **Do while loop**

```
import java.util.Scanner;

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

      do
      {
         System.out.print("Enter a number (0 to exit): ");
         number = scanner.nextInt();
         if (number != 0)
         {
            System.out.println("You entered: " + number);
         }
       }
         while (number != 0);
         // Loop continues until 0 is entered

      System.out.println("Exiting...");
      scanner.close();
   }
}
```

Conclusion:

When combined, branching and looping allow for powerful problem-solving capabilities:

- **Complex Logic**: Branching can be used within loops to create complex decision-making processes for each iteration.
- **Dynamic Algorithms**: Many algorithms, such as search and sort, rely on both branching and looping to navigate through data and make decisions efficiently.