Vidyavardhini's College of Engineering and Technology
Department of Artificial Intelligence & Data Science

Experiment No.1
Basic programming constructs like branching and looping
Date of Performance:18/07/24
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: -

NESTED IF-ELSE:

```
import java.io.*;  public class
NestedIfElse{  public static void
main(String args[]) {  int age = 15;
boolean isStudent = true;

        if (age >= 18) {
          if (isStudent) {
              System.out.println("Adult Student");
           } else {
              System.out.println("Adult Non-Student");
           }
        } else {  if
          (isStudent) {
              System.out.println("Minor Student");
           } else {
              System.out.println("Minor Non-Student");
           }
        }
    }
}
```

OUTPUT:

```
C:\Users\Shubham\Desktop\JAVA>javac  NestedIfElse.java

C:\Users\Shubham\Desktop\JAVA>javac  NestedIfElse.java

C:\Users\Shubham\Desktop\JAVA>java  NestedIfElse.java
Minor Student

C:\Users\Shubham\Desktop\JAVA>
```

SWITCH CASE:

```
import java.util.Scanner;


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


      System.out.print("Enter marks (0-100):
     ");  int marks = sc.nextInt();  if (marks < 0
     || marks > 100) {
         System.out.println("Invalid marks. Please enter a value between 0 and 100.");
      } else {  int
        gradeCategory;  if
        (marks >= 90) {
        gradeCategory = 1;
         } else if (marks >= 80) {
           gradeCategory = 2;
         } else if (marks >= 70) {
           gradeCategory = 3;
         } else if (marks >= 60) {
           gradeCategory = 4;
         } else {
```

```
            gradeCategory = 5;
        }
        switch (gradeCategory) {
            case 1:
                System.out.println("Grade: A");
                break;
            case 2:
                System.out.println("Grade: B");
                break;
            case 3:
                System.out.println("Grade: C");
                break;
            case 4:
                System.out.println("Grade: D");
                break;
            case 5:
                System.out.println("Grade: F");
                break;
            default:
                System.out.println("error");
                break;
        }
    }

    scanner.close();
  }
}
```

OUTPUT:

```
C:\Users\Shubham\Desktop\JAVA>javac Grade.java

C:\Users\Shubham\Desktop\JAVA>java Grade.java
Enter marks (0-100): 98
Grade: A

C:\Users\Shubham\Desktop\JAVA>
```

FOR LOOP:

```
import java.io.*;
public class ForLoop {
    public static void main(String args[]) {
      System.out.println("Numbers from 1 to 10:");
      for (int i = 1; i <= 10; i++) {
          System.out.println(i);
       }
    }
}
```

OUTPUT:

```
Numbers from 1 to 10:
1
2
3
4
5
6
7
8
9
10
```

DO WHILE LOOP:

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

```
public class DoWhile{  public static void
  main(String args[]) {  Scanner sc = new
  Scanner(System.in);  int num;  do {
        System.out.print("Enter a number (enter 0 to exit):
        ");  num = sc.nextInt();  if (number != 0) {
            System.out.println("You entered: " + number);
        }
    }  while (number != 0);
     System.out.println("Program has ended.");
    sc.close();
  }
}
```

Output:

```
Enter a number (enter 0 to exit): 2
You entered: 2
Enter a number (enter 0 to exit): 4
You entered: 4
Enter a number (enter 0 to exit): 0
Program has ended.
```

Conclusion:

Branching enables a program to choose different paths of execution based on certain conditions.It helps in managing different scenarios that might arise, such as handling invalid input or different error states.It directs the flow of control in a program.