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 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:**

-

1. If-Else:)IF-ELSE class Condition {

```
public static void main(String
args[]) {  int a = 10;   int b = 20;   if (a
== b) {
 System.out.println("This is my first program");
 } else {
 System.out.println("Invalid condition");
 }
 }
```

```
}
```

2.Nested-If:

NESTED-IF

```
Class Nestedif {
 Public static void main(String[] args) {
 Int age = 21;
 Int weight = 75;
 If (age >= 18) {
 If (weight > 50) {
 System.out.println("You can donate blood");
 }
 }
 Else
 System.out.println("You cannot donate blood");

 }
}
```

3.Switch Case:

```
public class SwitchExample {   public
static void main(String[] args) {
//Declaring a variable for switch
expression      int number=20;
//Switch expression      switch(number){
//Case statements      case 10:
System.out.println("10");      break;
case 20: System.out.println("20");

break;      case 30:

System.out.println("30");      break;

//Default case statement

default:System.out.println("Not in 10, 20 or

30");

   }

}

}
4. Break: class Break {   public static

void main(String args[]) {

 int i;   for (i = 0; i < 5;

i++) {   if (i == 3)

break;

 System.out.println(i);

 }

 }

}

5)CONTINUE    class Continue {

public static void main(String

args[]) {

 int i;   for (i = 0; i < 5;

i++) {   if (i == 3)

continue;

 System.out.println(i);
 }
 }
}
```

6)WHILE

```
Class While

{

 Public static void main(String[] args) {    Int a = 0;

 While (a < 10) {

 System.out.println(a);

 A++;

 }

 }
```

7)DO-WHILE

```
class Do {  public static void
main(String[] args) {  int a = 0;   do {
a
= a + 1;
 System.out.println(a);
 } while (a < 10);
 }
8FOR    class For {   public static

void main(String[] args) {   int a =

10;    int i;   for (i = 1; i <= a; i++) {

System.out.println(i);

 }

 }
```

}   Conclusion:

Branching and looping play a crucial role in problem-solving in Java. With branching, we can make decisions based on conditions, allowing our programs to take different paths. Looping enables us to repeat a block of code multiple times, which is useful when we need to iterate over data or perform tasks repeatedly. Together, these concepts provide us with the necessary tools to create efficient and effective solutions to various problems in Java programming.