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: -

Branching and Looping

1. IfElse

```
class IfElse
{
  public static void main(String args[])
     {

      int a=5;
      if(a>10)
       {
        System.out.println("\nNumber is smaller than 10");
       }
      else
       {
        System.out.println("\nNumber is smaller than 10");
        }


     }
}
```

2. NestedIfElse

```
class NestedIfElse
{
  public static void main(String args[])
     {

      int a=15;
      int b=10;
      int c= 26;
      if(a>b)
       {
        if(a>c)
```

```
        {
         System.out.println("\nA is Greater");
        }
       else
        {
         System.out.println("\nC is Greater");
        }
       }
      else
       {
        if(b>c)
        {
          System.out.println("\nB is Greater");
        }
        else
        {
         System.out.println("\nC is Greater");
        }
       }



    }
}
```

3. IfElseIf

```
class IfElseIf
{
  public static void main(String args[])
     {
int a=99;

if(a>=90)
{
System.out.println("\nGrade : A+");
}
else if(a>=80)
{
System.out.println("\nGrade : A");
}
else if(a>=60)
{
System.out.println("\nGrade : B");
}
else if(a>=40)
```

```
{
System.out.println("\nGrade : C");
}
else
{
System.out.println("\nFail");
}
}
}
```

4. Switch

```
class Switch
{
  public static void main(String args[])
    {
int a=1;
switch(a)
{
case 1:
System.out.println("\nMonday");
break;

case 2:
System.out.println("\nTuesday");
break;

case 3:
System.out.println("\nWednesday");
break;

case 4:
System.out.println("\nThursday");
break;

case 5:
System.out.println("\nFriday");
break;

case 6:
System.out.println("\nSaturday");
break;

case 7:
System.out.println("\nSunday");
```

```
break;

default:
System.out.println("\nInvalid");
}
}
}
```

5. class For

```
{
  public static void main(String args[])
     {
int a=0;
for(a=0;a<=50;a++)
  {
    if(a%2==0)
     {
       System.out.println(a);
     }
  }
}
}
```

6. While

class While

```
{
  public static void main(String args[])
     {
int a=0;
while(a<=25)
 {
   if(a%2!=0)
   {
   System.out.println(a);
   }
  a++;
}
}
}
```

7. DoWhile

```
public class doWhile {
public static void main(String[] args) {
   int i=1;
   do{
      System.out.println(i);
   i++;
   }while(i<=10);
}
```

Conclusion:

Comment on how branching and looping useful in solving problems.

Branching (if-else statements, switch statements, etc.):
Conditional Execution: Branching allows you to execute different code blocks based on specific conditions. This is crucial for making decisions in your program, such as handling different cases or responding to user input.
Decision-Making: You can use branching to make decisions, such as determining which path to take in an algorithm, processing different inputs, or responding to specific events.
Error Handling: Branching is vital for error handling. You can use it to check for errors and exceptions, providing graceful ways to handle unexpected situations and prevent program crashes.
Customization: You can customize the behavior of your program for different scenarios. For example, you can create software that adapts to user preferences or responds to changing data.

Looping (for loops, while loops, etc.):
Iteration: Loops enable you to repeat a set of instructions multiple times. This is invaluable for processing large datasets, performing repetitive tasks, or executing the same operation with slight variations.
Efficiency: Loops help optimize code by reducing redundancy. Instead of writing the same code multiple times, you can use loops to iterate through data or perform a task as many times as necessary.
Automation: They are essential for automation, allowing you to automate tasks, batch processing, and complex computations. This can significantly increase efficiency and reduce human error.
Data Processing: Loops are vital for processing data structures like arrays, lists, and collections. They enable you to perform operations on each element in the collection.
State Management: Loops are often used to manage program state. You can use them to maintain and update the state of your application, whether it's a game, simulation, or any interactive system.