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

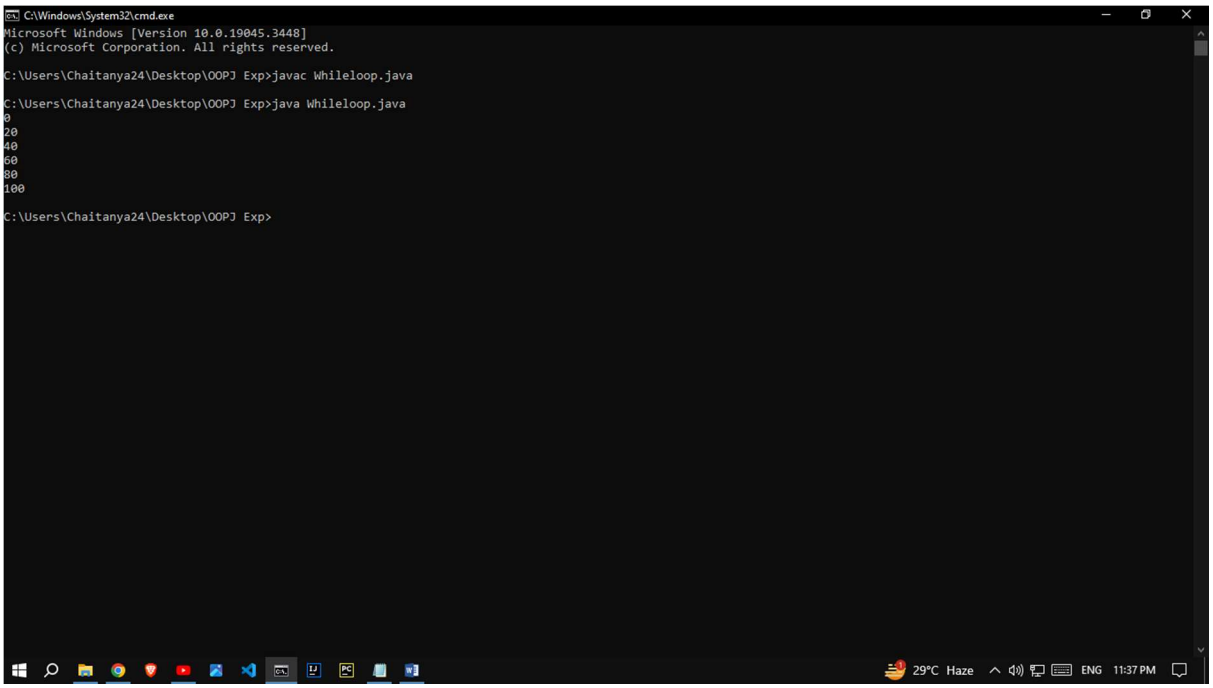
- for loop
- Some languages have modified for loops for more convenience eg :- Modified for loop in java.

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

Code: -

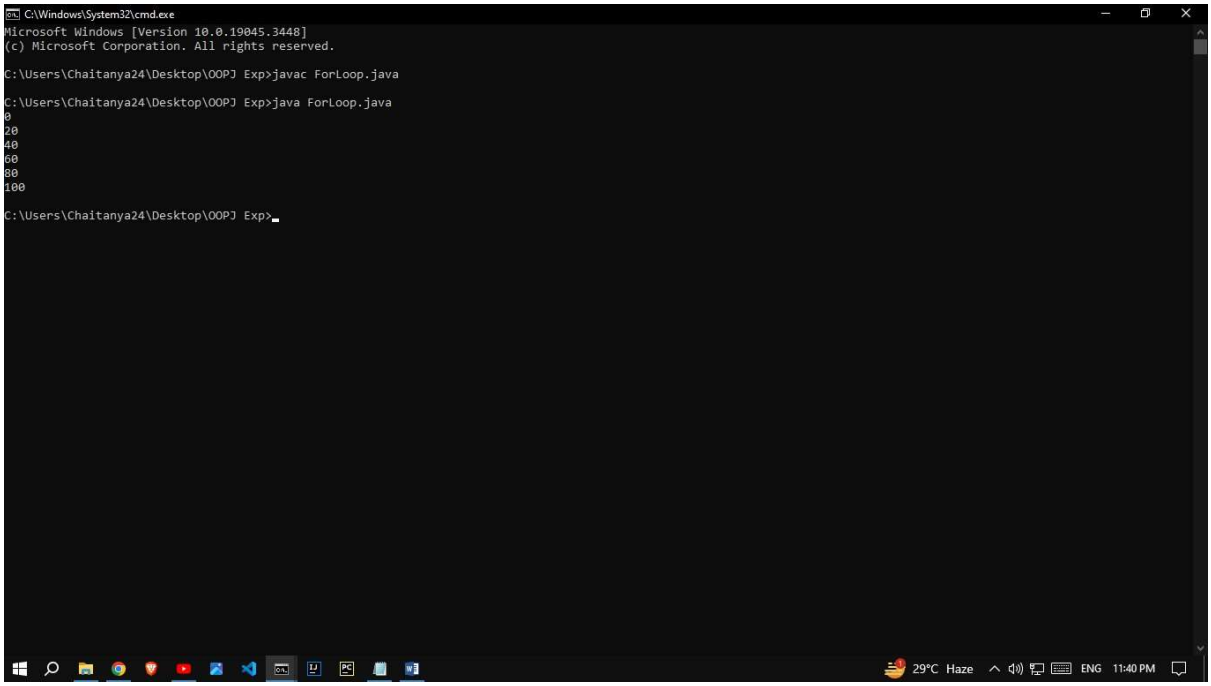
1} while loop

```
class Whileloop

{

    public static void main(String arg[])

  {

int a=0;

   while(a<=100)

 {

   if(a%20==0)

   {

    System.out.println(a);

    } a++;

 }

 }

}
```



2} for loop

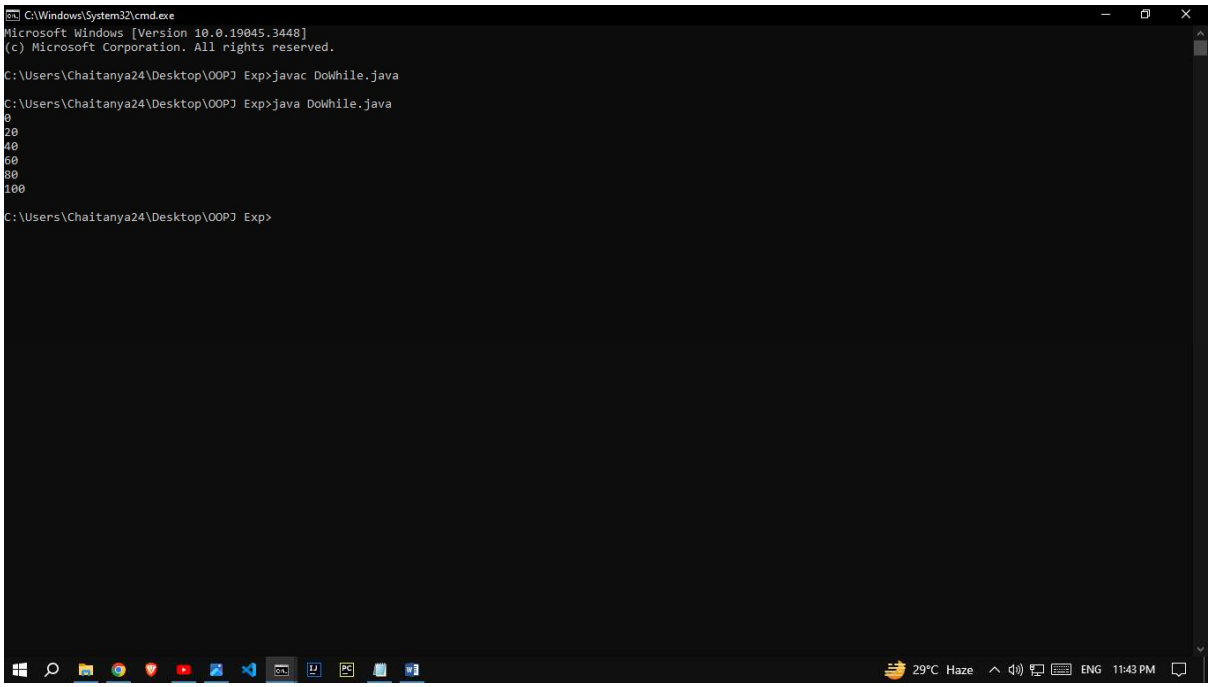
```
class Forloop

{

 public static void main(String arg[])

  {

    int a;

for(a=0;a<=100;a++)

  {

    if(a%20==0)

     {

       System.out.println(a);

      }

  }

 }

}
```



3} dowhile loop

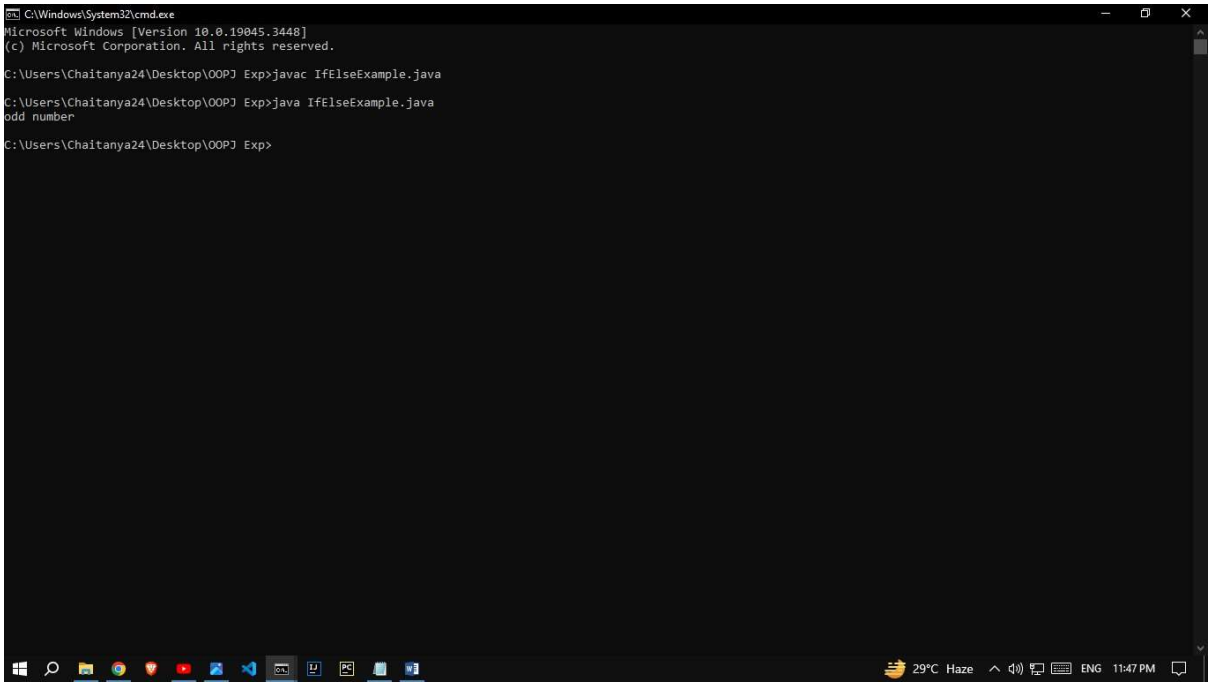
```
class Dowhileloop
{
    public static void main(String arg[])
   {
int a=0;
   do
  {
    if(a%20==0)
    {
     System.out.println(a);
     } a++;
  } while(a<=100);
  }
```

}



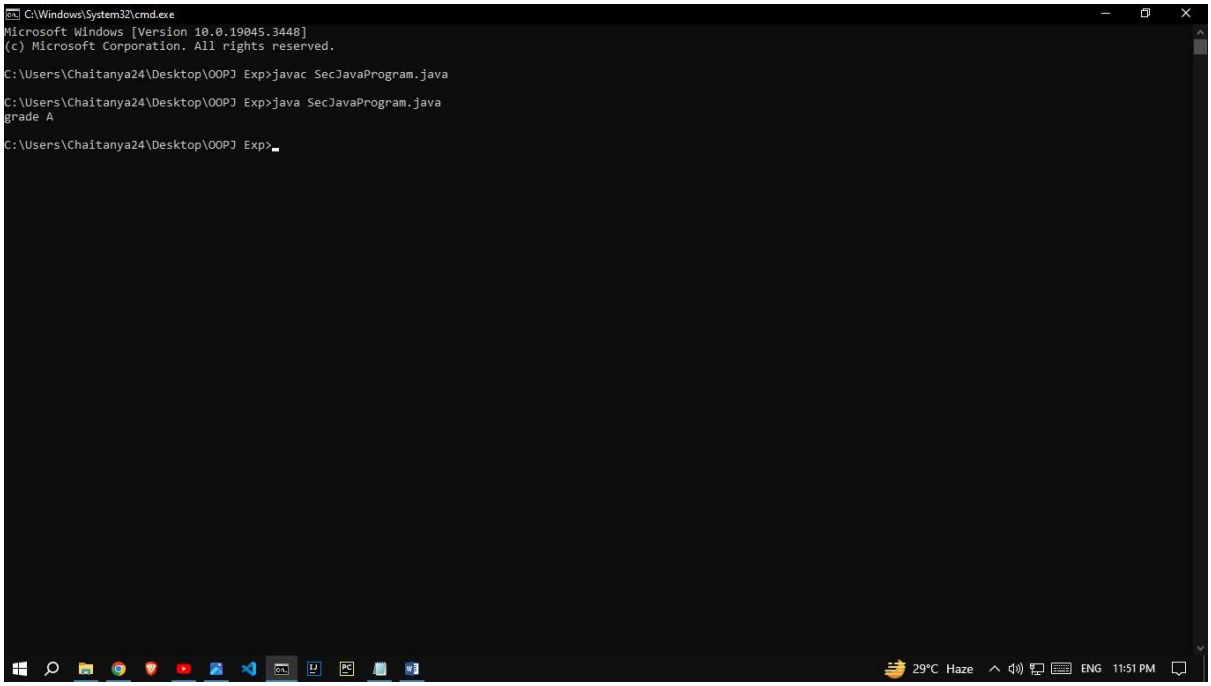
4}if else

```
public class IfElseExample {

public static void main(String[] args) {

    int number=13;

        if(number%2==0){

        System.out.println("even number");

    }else{

        System.out.println("odd number");

    }

}
```



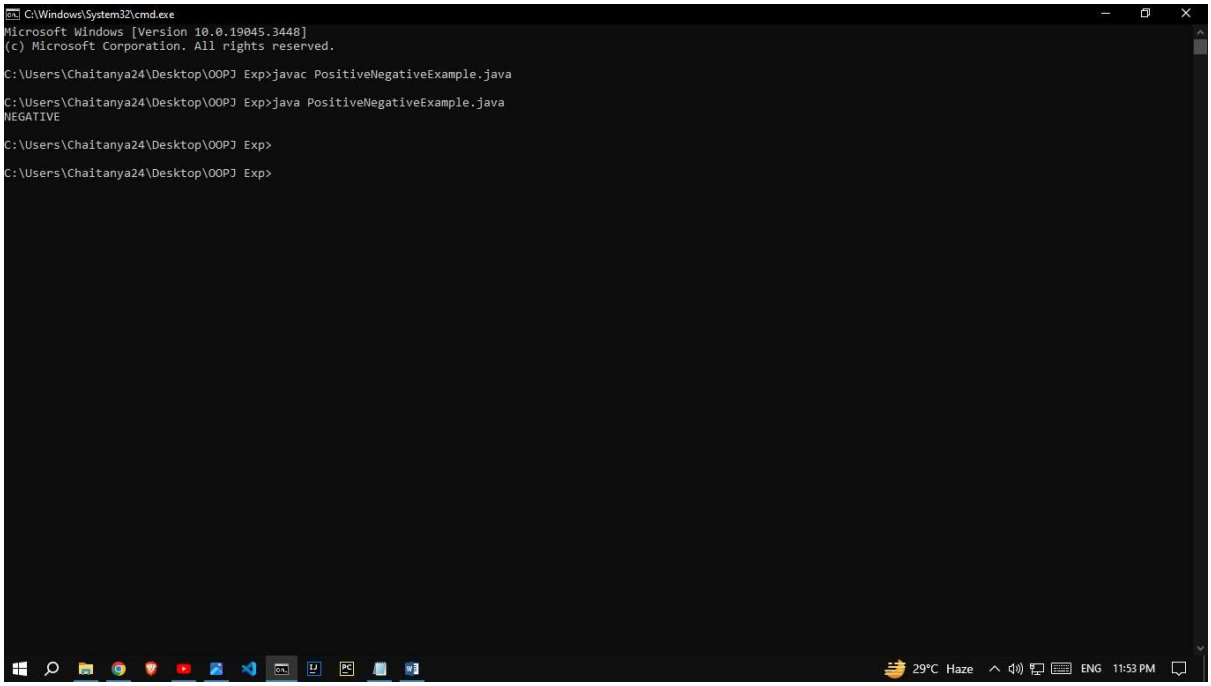
5} Ladder if else

```
class SecJavaProgram
{
 public static void main(String args[])
{
 int a=90;
if(a>=90)
{
System.out.println("grade A");
}
else if(a>=80)
{
System.out.println("grade B");
}
else if(a>=70)
{
System.out.println("grade c");
}
else if(a<70)
{
System.out.println("grade F");
}
}}
```



6} nested if else

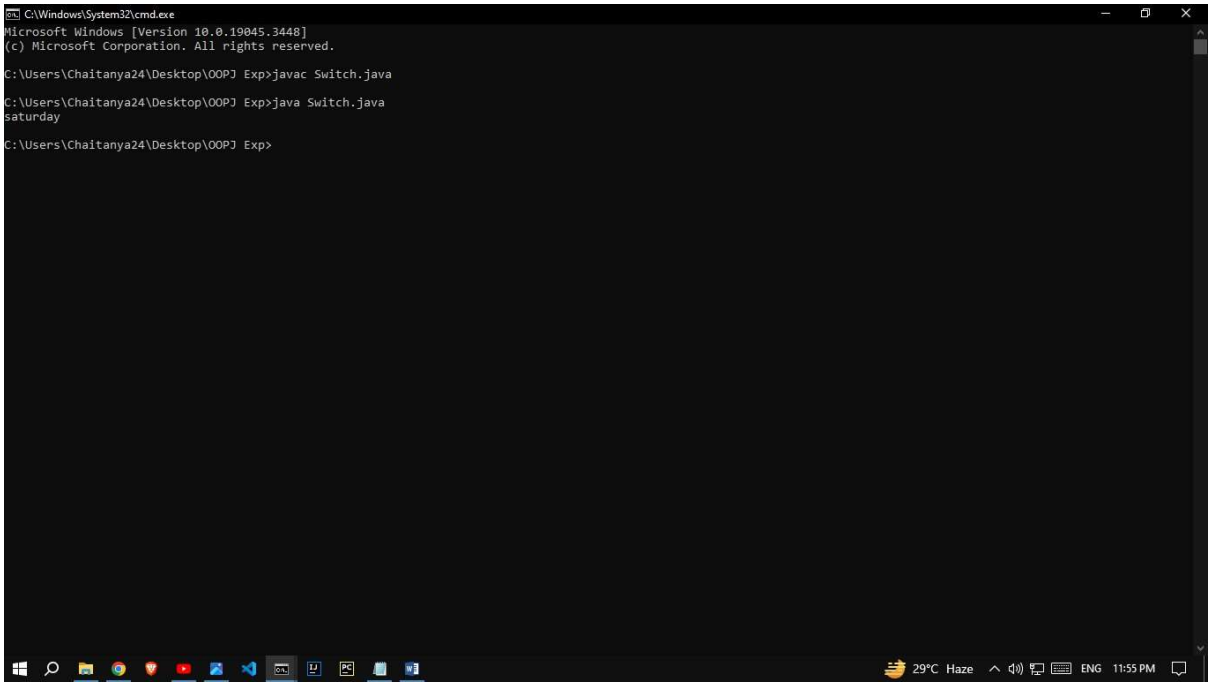
```
public class PositiveNegativeExample {
public static void main(String[] args) {
   int number=15;
   if(number>0){
   System.out.println("POSITIVE");
   }else if(number<0){
   System.out.println("NEGATIVE");
   }else{
   System.out.println("ZERO");
  }
}}
```



7} switch

```
class SwitchProgram
{
  public static void main(String args[])
        {
         int a = 6 ;
         switch(a)
         {
         case 1 :
            System.out.println("\n Monday");
            break;
         case 2 :
            System.out.println("\n Tuesday");
            break;
         case 3 :
            System.out.println("\n Wednesday");
            break;
         case 4 :
            System.out.println("\n Thursday");
            break;
         case 5 :
            System.out.println("\n Friday");
            break;
         case 6 :
            System.out.println("\n Saturday");
```

```
        break;
    case 7 :
        System.out.println("\n Sunday");
        break;
    default :
        System.out.println("\n Not Valid");
    }
} }
```



Conclusion:

1) Comment on how branching and looping useful in solving problems.

➢ **Branching (Conditional Statements):**

- Decision Making: Conditional statements like if, else if, and switch allow you to make decisions in your code. You can choose different paths based on the values of variables or conditions, making it possible to solve problems with multiple possible outcomes.

- Error Handling: Exception handling with try, catch, finally, and throw allows you to gracefully handle errors, ensuring that your program doesn't crash when

unexpected issues arise. This is crucial for robust problem-solving.

- Boolean Logic: You can use logical operators (e.g., &&, ||, !) to create complex conditions, which are often necessary for solving real-world problems that involve multiple criteria.

➢ **Looping (Iteration):**

- Repetition: Loops like for, while, and do-while enable you to execute a block of code repeatedly. This is invaluable for solving problems that involve processing a sequence of data or performing the same operation multiple times.
- Array and Collection Processing: When working with arrays, lists, or other collections, loops are essential for iterating through the elements and performing actions on each item. This is fundamental for data manipulation and analysis.
- Simplifying Code: Loops help avoid code duplication. Instead of writing the same code segment multiple times, you can encapsulate it within a loop and make your code more concise and maintainable.