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

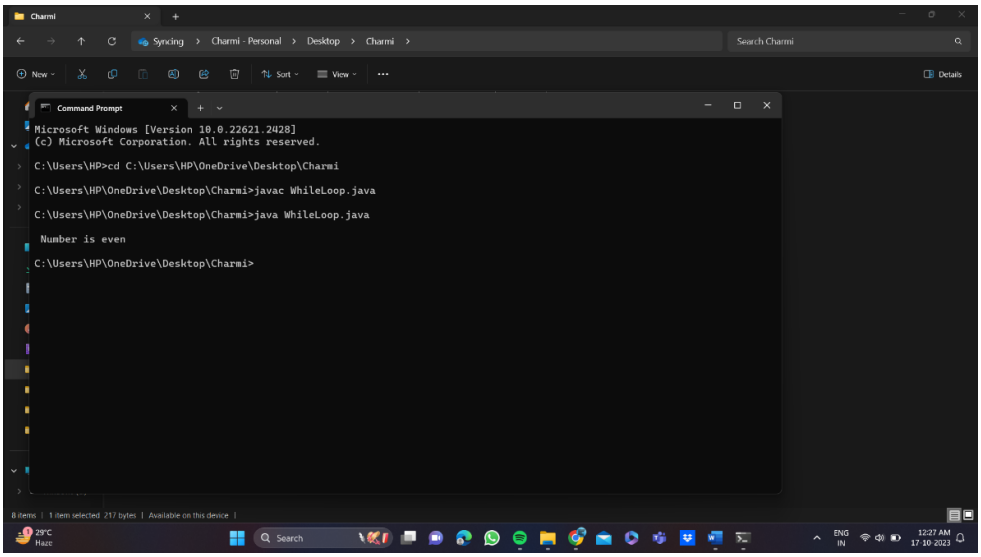
- for loop
- Some languages have modified for loops for more convenience eg :- Modified for loop in java.

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

Code: -

1} while loop

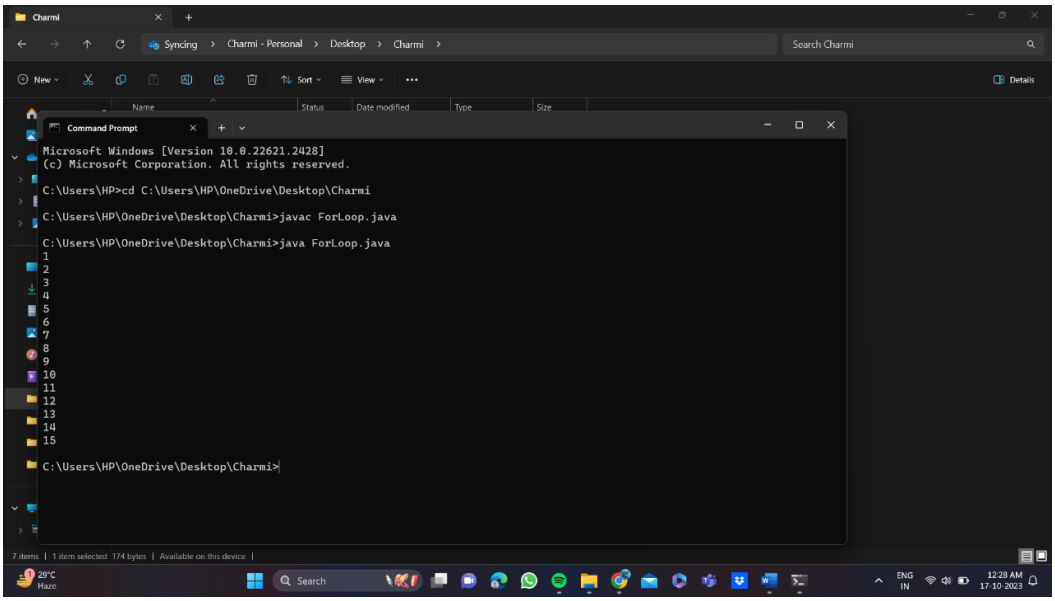
```
class Whileloop
{
  public static void main(String args[])
        {
          int a=4;
          while(a%2==0)
          {
            System.out.println("\n Number is even");
            break;
          }
        } }
```



2} for loop

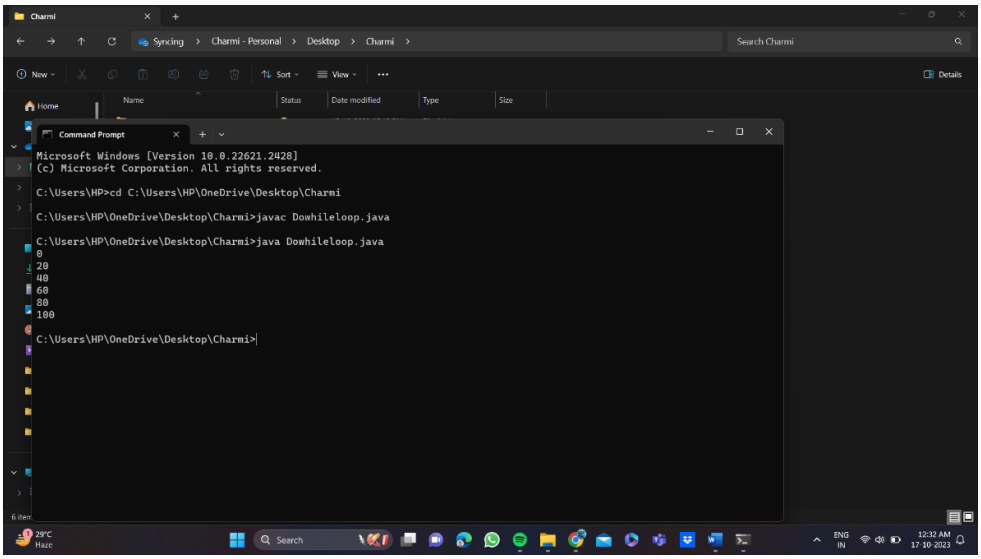
```
class Forloop
{
  public static void main(String args[])
    {
     int x;
     for(x=1;x<=10;x++)
     {
```

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



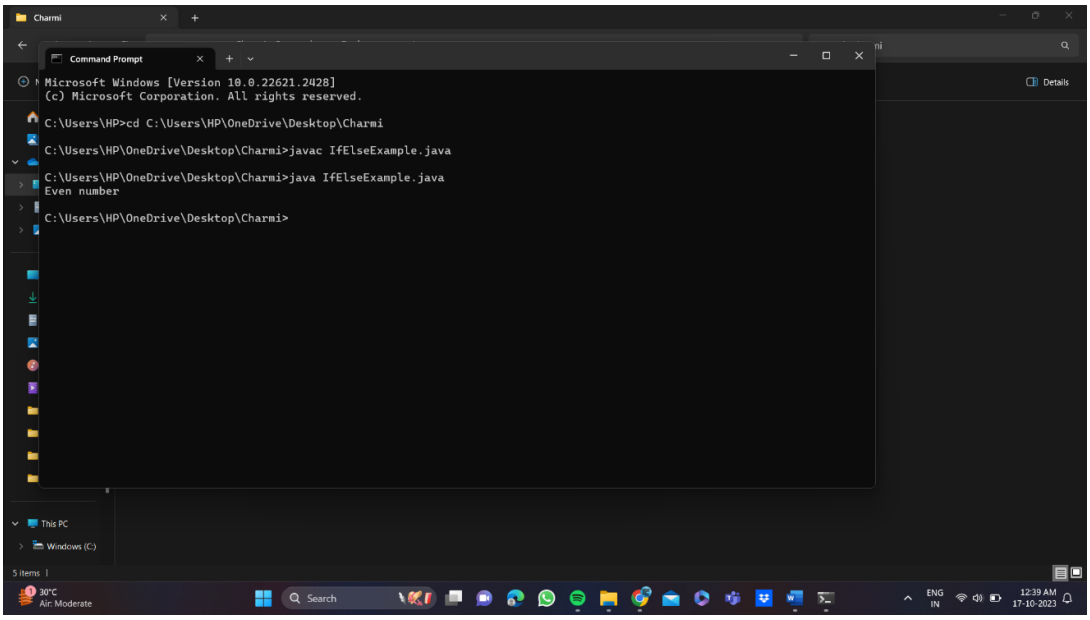
3} dowhile loop

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



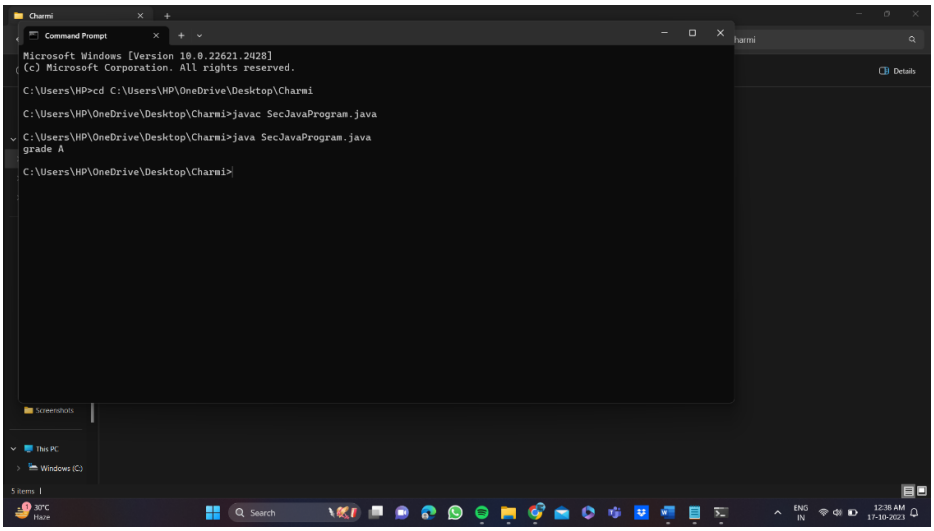
4}if else

```
public class IfElseExample {
public static void main(String[] args) {
    int number=10;
        if(number%2==0){
        System.out.println("Even number");
    }else{
        System.out.println("Odd number");
    }
}
}
```



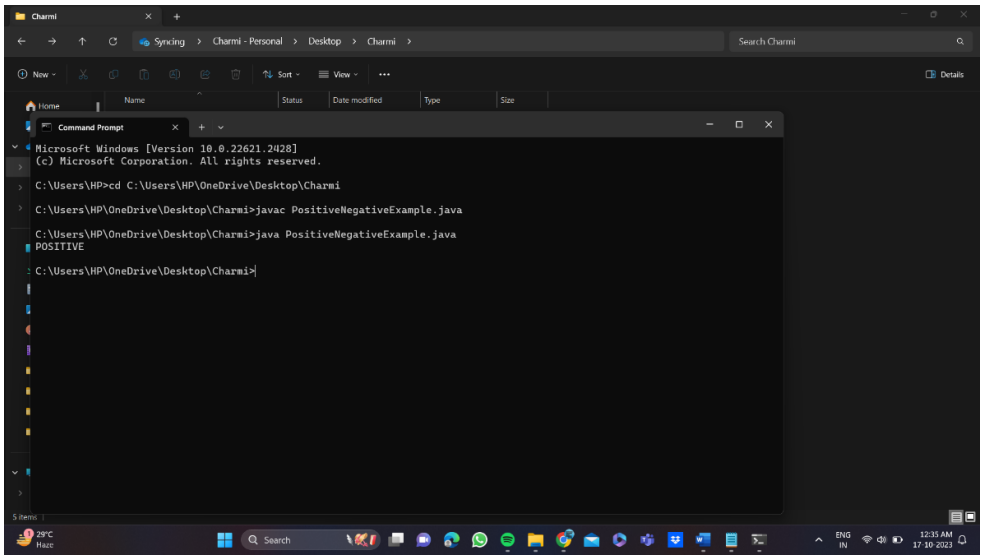
5} Ladder if else

```
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");
}
}}
```



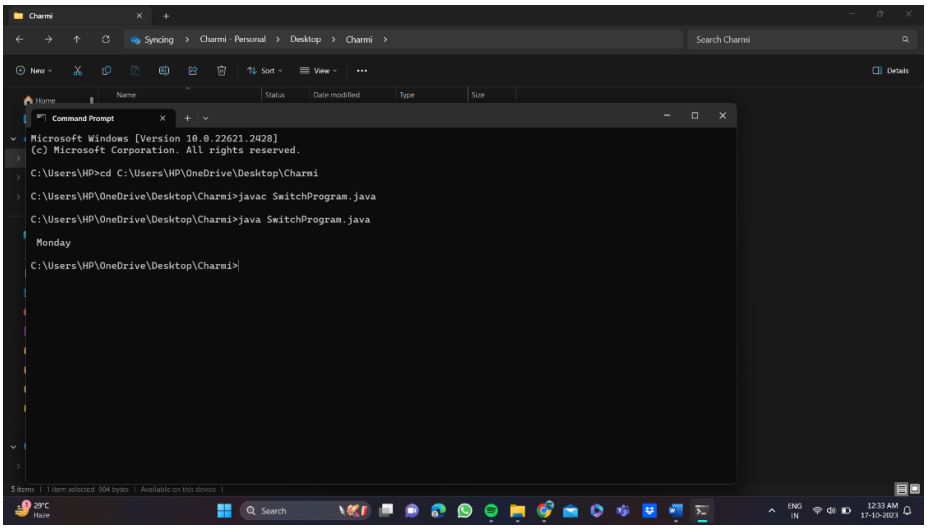
6} nested if else

```
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 = 1 ;
         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.

Java, along with many other programming languages, relies on fundamental control structures, branching, and looping, to tackle a diverse array of problems. These constructs empower your code with decision-making capabilities and the ability to perform repetitive tasks, enhancing its flexibility and adaptability.

Branching (if statements):

Decision Points: If statements are your tools for introducing decision points in your code. They allow you to choose different paths based on whether certain conditions hold true or false. In essence, they enable your code to adapt to various scenarios.

Looping:

Iterative Processes: Loops, available in different forms like for, while, and do-while, equip your code with the power of repetition. This is particularly handy when you need to perform the same block of code multiple times, be it for processing data structures like arrays and lists or conducting iterative computations.