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:
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

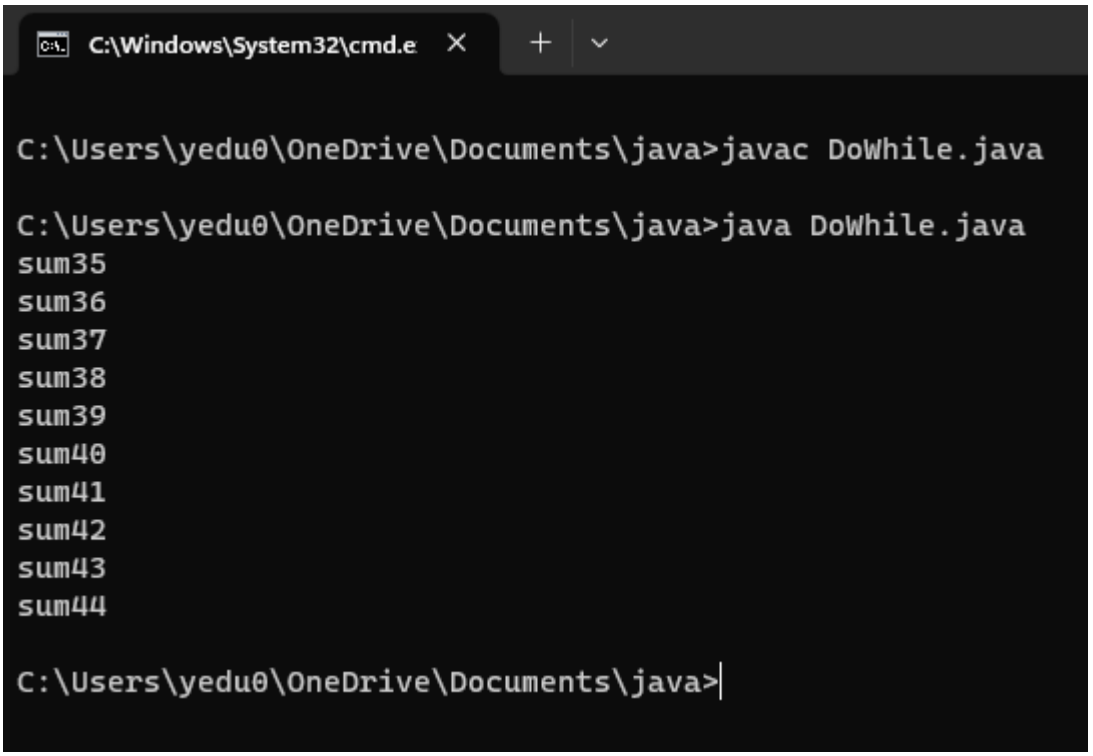
- for loop
- Some languages have modified for loops for more convenience eg :- Modified for loop in java.

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

Code: - DO-WHILE

```
class While
{
 public static void main(String args[])
{
 int n=0;
 int a=15;
 int b=20;
 int sum;
do
{
 sum=a+b+n;
 System.out.println("sum"+sum);
 n++;
}
while(n<10);
}
}
```

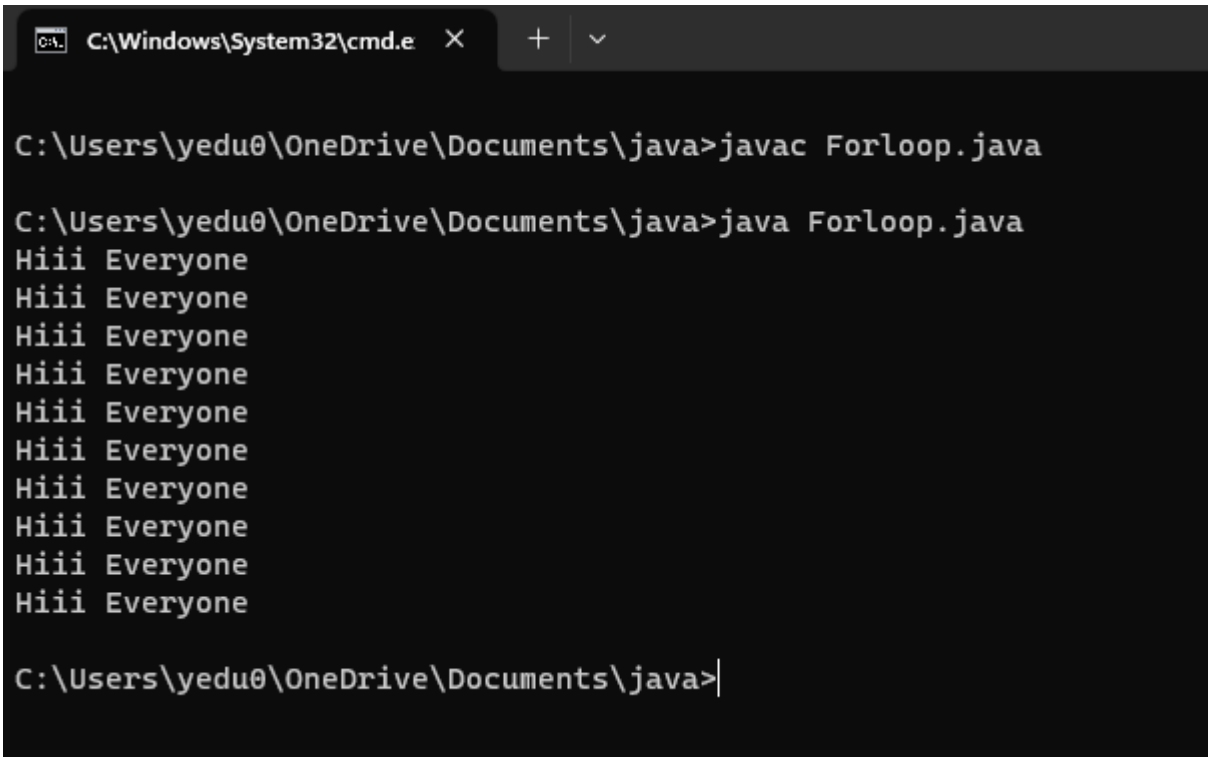
OUTPUT:



FOR

```
class Forloop
{
public static void main(String[]args)
{
 int i;
 int a=10;
for(i=0;i<10;i++)
 {
 System.out.println("Hiii Everyone");
 }
 }
}
```

OUTPUT



IF-FOR

```
class IfFor {
  public static void main(String args[]) {
    int a = 10;
    int b = 20;
    int i;
    for (i = 0; i < 5; i++) {
      if (a == b) {
        System.out.println("This is my first program");
      } else {
        System.out.println("Invalid condition");
      }
    }
  }
}
```

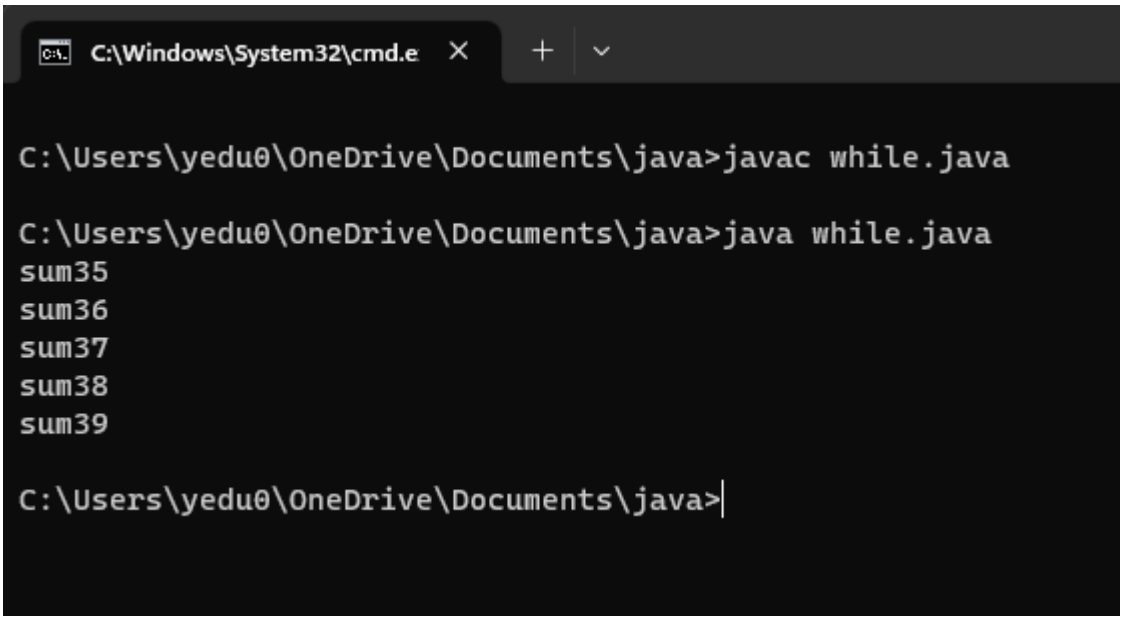
OUTPUT

```
if ($?) { javac IfFor.java } ; if ($?) { java IfFor }
Invalid condition
Invalid condition
Invalid condition
Invalid condition
Invalid condition
```

WHILE

```
class While
{
 public static void main(String args[])
{
 int n=0;
 int a=15;
 int b=20;
 int sum;
 while(n<5)
{
 sum=a+b+n;
 System.out.println("sum"+sum);
 n++;
}
}
}
```

OUTPUT



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);
    }
  }
}
```

OUTPUT

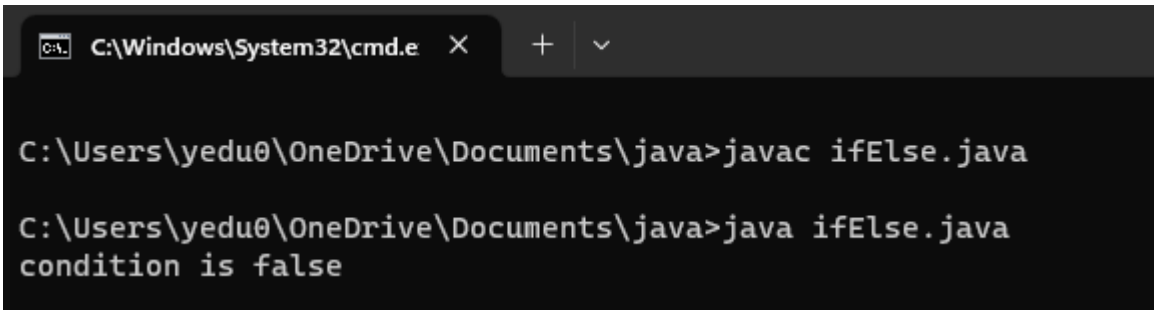
```
if ($?) { javac Break.java } ; if ($?) { java Break }
0
1
2
```

IF-ELSE

```
class IfElse
 {
 public static void main(String args[])
 {
 int a=25;
 int b=20;

 if(a<b)
 {
 System.out.println("condition is true");
 }
 else
 System.out.println("condition is false");
 }
 }
```

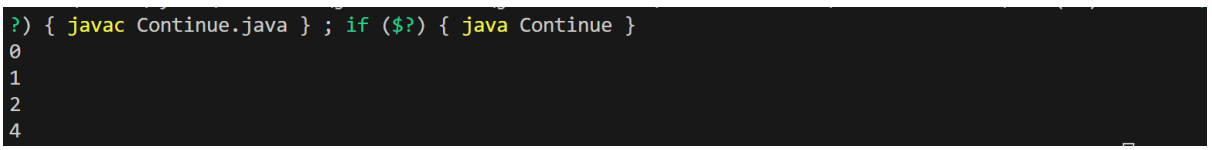
OUTPUT



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);
    }
  }
}
```

OUTPUT



IF-ELSE LADDER

```
class ifelseladder {
  public static void main(String args[]) {
    int a = 10;
    int b = 20;

    if (a == b) {
      System.out.println("a is equal to b");
    } else if (a > b) {
      System.out.println("A is greater then b");
    } else {
      System.out.println("b is greater than a");
    }

  }

}
```

OUTPUT

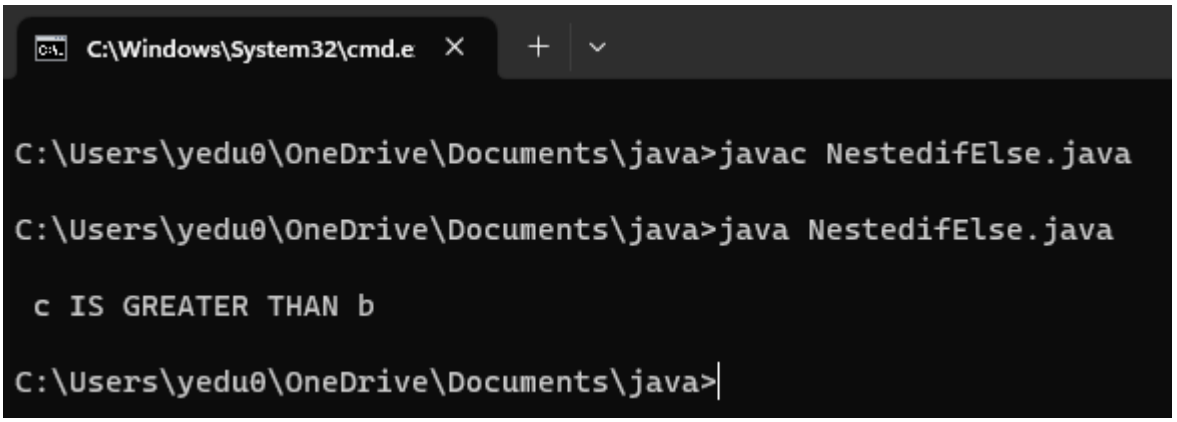
```
?) { javac ifelseladder.java } ; if ($?) { java ifelseladder }
b is greater than a
```

NESTED-IF

```
class NestedifelseProgram{
 public static void main(String args[])
 {
 int a=10;
int b=20;
int c=30;
if(a>b)
 {
 if(a>c)
 {
 System.out.println("\n a IS GREATER THAN c");
 }
 else
 {
System.out.println("\n c IS GREATER THAN a");
 }
 }
 else
 {
 if(b>c)
 {
 System.out.println("\n b IS GREATER THAN c");
 }
 else
 {
 System.out.println("\n c IS GREATER THAN b");
 }
```

```
 }
}
}
```

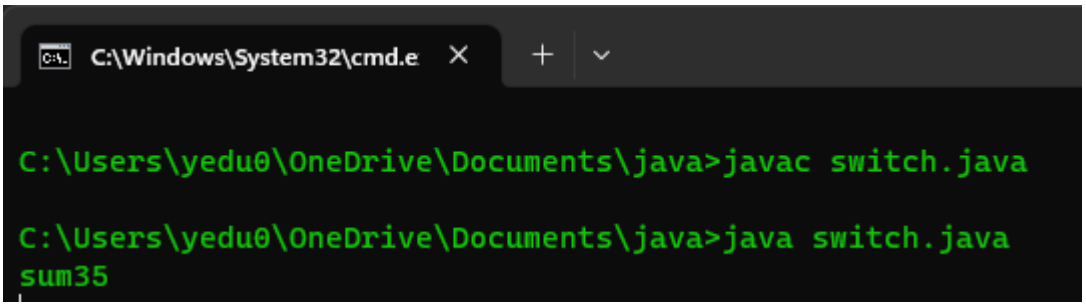
OUTPUT



SWITCH

```
class While
{
 public static void main(String args[])
{
 int n=0;
 int a=15;
 int b=20;
 int sum;
while(n<2)
{
 sum=a+b;
switch(n)
{
case 0:
System.out.println("sum"+sum);
n++;
break;
case 2:
System.out.println("invalid");
break;
}
}
}
}
```

OUTPUT:



Conclusion:

Comment on how branching and looping useful in solving problems.

Branching (if-else and switch statements): Branching is pivotal for decision-making within a program. It enables the execution of different blocks of code based on certain conditions. In Java, the if-else and switch statements are commonly used for branching. They help in handling various scenarios, such as handling user inputs, managing exceptions, and implementing logic based on different conditions.

if-else: It allows the program to execute different blocks of code based on a condition's evaluation. This is crucial for creating adaptive code that responds differently to different inputs or situations.

switch: The switch statement provides an efficient way to select from many alternatives. It is particularly useful when dealing with a large number of potential execution paths based on the value of a single variable.

Looping (for, while, and do-while loops): Loops are vital for executing a block of code repeatedly. They allow programmers to automate repetitive tasks, process collections of data, and perform iterative operations. In Java, the commonly used loops are:

for loop: It is used when the number of iterations is known beforehand. It provides a concise way to write the loop's initialization, condition, and increment/decrement in a single line.

while loop: This loop continues to execute a block of code as long as a specified condition is true. It is particularly useful when the number of iterations is not predetermined.

do-while loop: This loop is similar to the while loop, but it guarantees that the block of code will be executed at least once, even if the condition is initially false.