Decision making and looping are the concepts which works as a control statement in Java. Control statements in Java are the instructions which can control the flow of execution of a program. In control statement, there are some conditions which are specified by a programmer. These conditions are evaluated by the system and a particular block of statements are executed.

Following are the types of control statements:

**Decision-making statements**

**Looping statements**

## Decision-making statements

Decision-making statements are the conditional instructions in the program which are evaluated and according to the result of that instruction particular block of statements are executed.

Decision making in Java is pretty much similar to decision making in real life.

Following are the types of decision-making statements:

**Simple if statement**

**if … else statement**

**else … if ladder**

**nested if statements**

**switch case statement**

**nested switch case**

### 1. Simple if statement

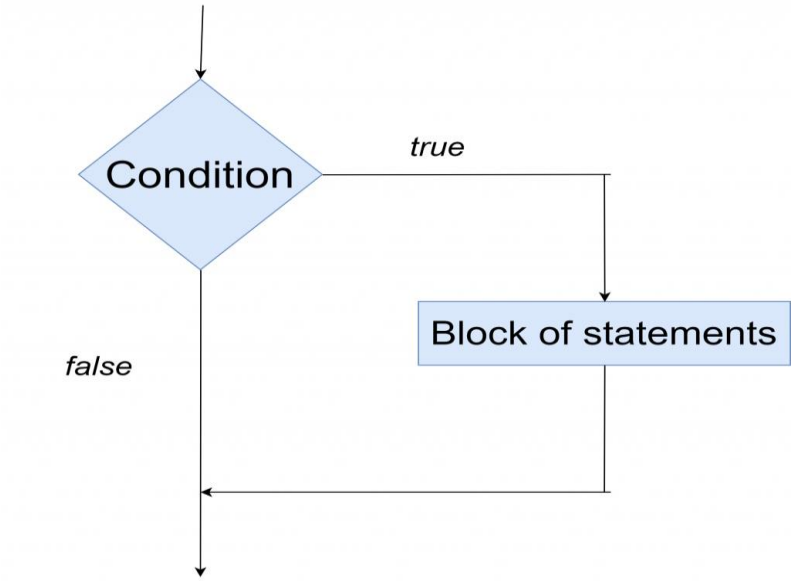
Simple if statement is used to execute a particular block of statements if the condition is true otherwise program continues its execution excepting the block of statement inside if block.

Syntax :

```
if(condition){


//block of statements which will be executed if condition is true.
```

}

Following is the flowchart of simple if statement




**Simple if statement**

Example :

```
class SimpleIf{
        public static void main(String args[]){
                int i=3;
                if(i<5)      //i<5 is a condition
                {
                   //this block is executed only if condition is true
                        System.out.println("If block executed");
                }
        }
}
```

Output :

C:\workspace>javac SimpleIf.java

C:\workspace>java SimpleIf

If block executed

C:\workspace>

In above example, i is an integer variable which is assigned by value 5, in above program simple if statement is checking the value of i. If it is less than 5, then statements inside if the block is executed.

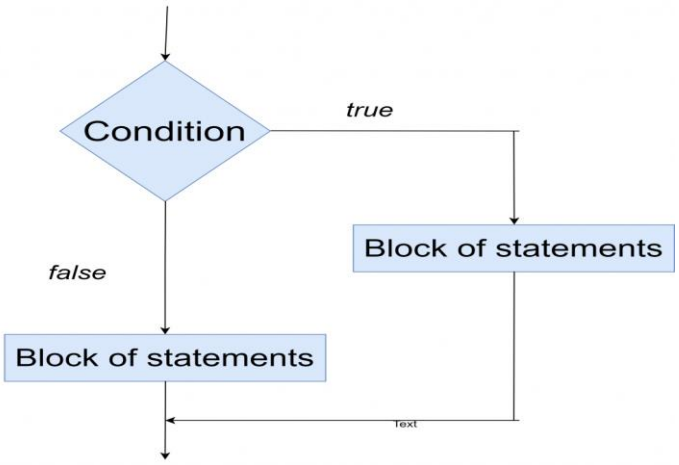
## 2. If … else statement

If else statement is a decision-making statement which is having true block as well as false block. In this type of statement, the condition is evaluated, if the condition is true then the true block of the statement is executed other false block.

Syntax :

```
if(condition){

//true block

}

else{

//false block

}
```

Following is the flowchart of if … else statement



## if … else statement

Example :

```
class Ifelse{
        public static void main(String args[]){
                int i=3;
                if(i<5)
                {
                        System.out.println("Value of i is less than 5");
                }
                else
                {
                        System.out.println("Value of i is not less than 5");
                }
        }
}
```

**Output :**

```
C:\workspace>javac Ifelse.java
C:\workspace>java Ifelse
Value of i is less than 5
C:\workspace>
```

In above example, value of integer variable i is being checked with integer value 5, if the value of i is less than 5, then true block will be executed, otherwise false block.

You can copy and run the program and change the value of i so that it is greater than 5 and execute it. The output will be “Value of i is not less than 5”.

### 3. Else … if ladder

Else … if ladder is a decision-making statement which is used to test sequence of test conditions.

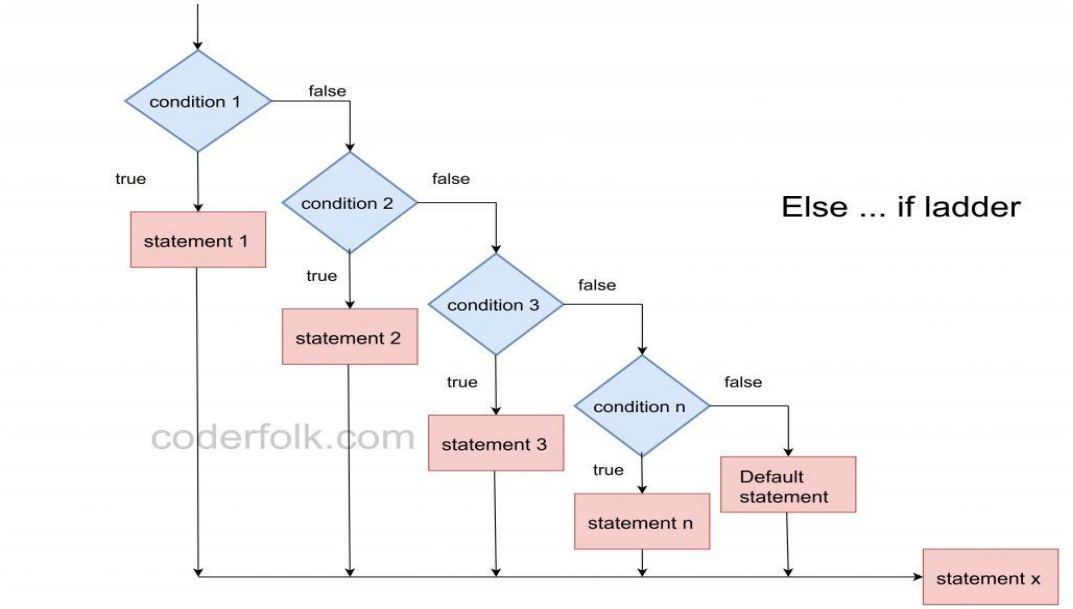
In this type of statement it checks for the condition and if the condition is satisfied then associated block of the statement will be executed.

If the condition is false then next condition in the Else … if ladder will be checked. If none of the conditions is true, the default block is executed.

Syntax :

```
if(condition 1){

//first block of statements

}

else if(condition 2){

//second block of statements

}

else if(condition 3){

//third block of statement

}

else if(condition n){

//nth block of statement

}

else{

//Default block of statement

}
```

**Flowchart :**




**Else if ladder**

Example :

```
class IfelseLadder{
	public static void main(String args[]){
		int marks=50;
		if(marks>75 && marks<=100)
		{
			System.out.println("A grade");
		}
		else if(marks>65 && marks<=75)
		{
			System.out.println("B grade");
		}
		else if(marks>55 && marks<=65)
		{
			System.out.println("C grade");
		}
```

```
            else if(marks>39 && marks<=55)
            {
                System.out.println("D grade");
            }
            else
            {
                System.out.println("FAIL");
            }
        }
}
```

**Output :**

```
C:\workspace>javac IfelseLadder.java

C:\workspace>java IfelseLadder

D grade

C:\workspace>
```

In above Else … if ladder example, marks is an integer variable which is having value 50. This variable is being checked with various conditions. If the first block is true, then first true block will be executed, if the second condition is true then the second block will be executed and it goes on.

If none of the conditions is true, then default block of statements is executed.

### 4. Nested if statement

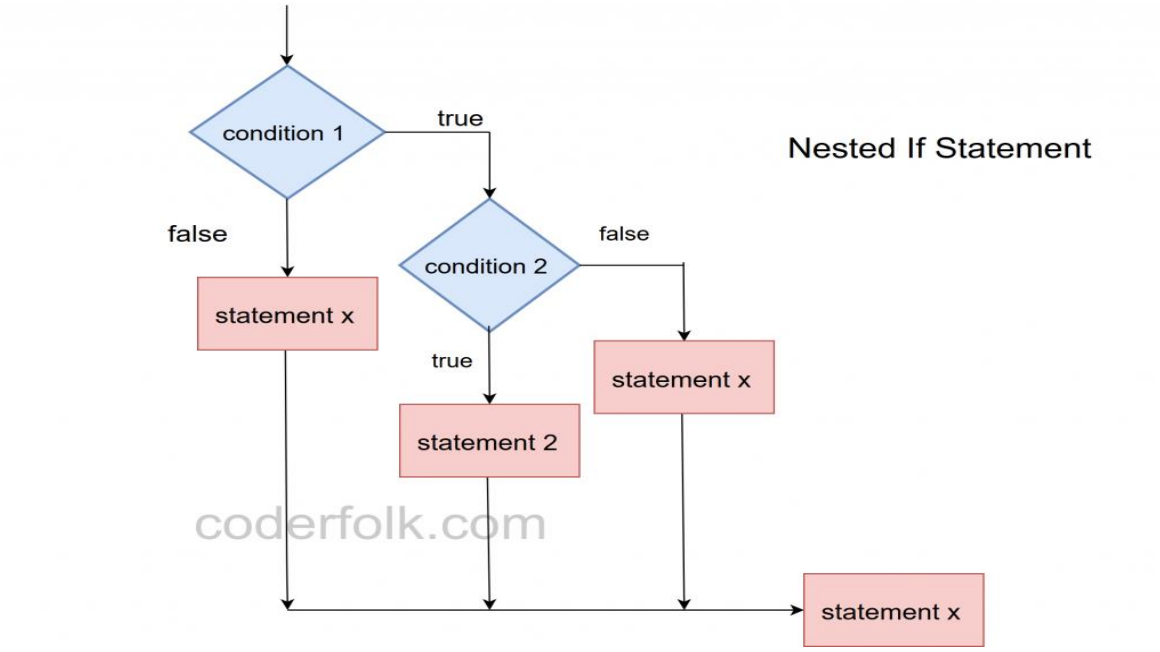
If statement inside another if statement is known as nested if statement. Nested means a block inside another block.

Syntax :

```
if(condition){
```

```
    //body of parent if
    if(condition){
     //body of nested if
    }
}
```

Flowchart :




**nested if**

Example :

```
class NestedIf{
        public static void main(String args[]){
                int i=16;
                if(i>10)
                {
                        System.out.println("Value of i is greater than 10");
                        if(i>15){
                                System.out.println("Value of i is greater than 15 as well");
```

```
                }
            }
        }
}
```

Output :

```
C:\workspace>javac NestedIf.java

C:\workspace>java NestedIf

Value of i is greater than 10

Value of i is greater than 15 as well

C:\workspace>
```

In above program, there are two if statements, the body of first if statement which is checking the value of i greater than 10 consists of another if statement which is checking the value of i greater than 15.

### 5. Switch Case Statement

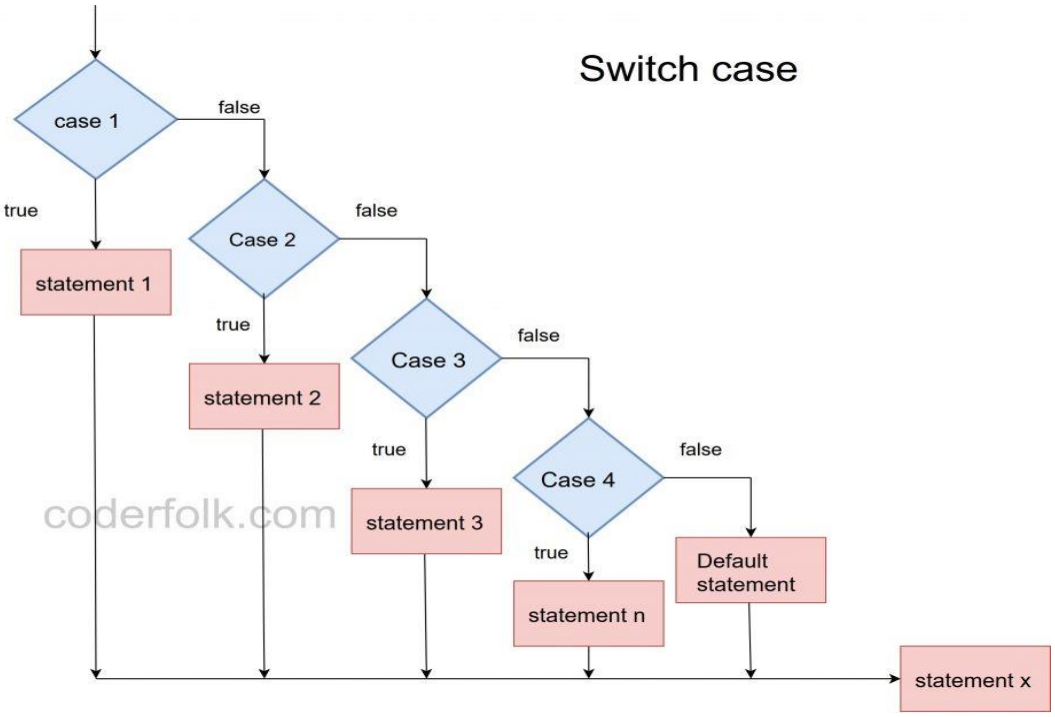
Switch case statement works same as else … if ladder but it checks only one variable. Else … if ladder can check multiple variables. In switch single variable is checked for multiple values.

Syntax :

```
switch(variable){
  case constant1:
        //body of case 1
        break;
  case constant2:
        //body of case 2
        break;
  case constant3:
```

```
        //body of case 3
        break;
    case constantn:
        //body of case n
        break;
    default:
        //body of default block (This will be executed if none of the above cases are true)
}
```

Flowchart :




**switch case**

Example :

```
class SwitchCase{
    public static void main(String args[]){
        int i=4;
        switch(i){
```

```
                case 1:
                        System.out.println("one");
                 break;
            case 2:
                        System.out.println("two");
                        break;
                case 3:
                        System.out.println("three");
                        break;
            case 4:
                        System.out.println("four");
                        break;
            case 5:
                        System.out.println("five");
                        break;
            default:
                        System.out.println("Invalid");
        }
    }
}
```

**Output :**

```
C:\workspace>javac SwitchCase.java
C:\workspace>java SwitchCase
four
```

C:\workspace>

In above program integer variable, i is a single variable which is checked inside condition. A single variable is tested with all the cases.

Note : You must use break keyword to break the flow of execution after execution of statement in true block. If you dont use break then next case will also be executed.

### 6. Nested Switch case

A switch case statement which consists of another switch case within its block is known as nested switch case statement.

Syntax :

```
switch(variable 1){
    case constant1:
      //body of case 1
       switch(variable 2){
            //body of nested switch
             case constant1:
                  //body of case 1
                  break;
             case constant2:
                  //body of case 2
                  break;
             case constantn:
                  //body of case n
                  break;
             default:
```

```
                    //default block of statement
                    //this block will be executed if none of the conditions are true
            }
        break;
    case constant2:
        //body of case 2
        break;
    case constantn:
        //body of case n
         break;
    default:
       //body of default block (This will be executed if none of the above cases are true)
}
```

Example :

```
class NestedSwitchCase{
    public static void main(String args[]){
            int i=1;
            switch(i){
                    case 1:
                            System.out.println("one");
                            char c='a';
                            switch(c){
                            //nested switch statement
                                    case 'a':
```

```
                    System.out.println("Character a");
                    break;
                case 'b':
                    System.out.println("character b");
                    break;
                default:
                    System.out.println("Invalid character");
            }
        break;
case 2:
            System.out.println("two");
            break;
      case 3:
            System.out.println("three");
            break;
case 4:
            System.out.println("four");
            break;
case 5:
            System.out.println("five");
            break;
default:
            System.out.println("Invalid");
```

```
            }
        }
}
```

**Output :**

```
C:\workspace>javac NestedSwitchCase.java

C:\workspace>java NestedSwitchCase
one
Character a

C:\workspace>
```

In above program, the first case of switch case contains another switch case which is checking character variable.

Note : No need to use break keyword in default block.

Note : You must put character literal between single quotes.

## Looping statements

Looping statements allow the programmer to execute some group of statement repetitively multiple times.

Control goes inside the body of the loop if the condition is true otherwise goes outside of looping block. The main difference between decision making statements and looping statement is decision making statement execute once alike looping statements executes several times.

There are two types of loop

**Entry control loop (while loop and for loop)**

**Exit control loop (do … while loop)**

Entry control loops are the loop in which condition is tasted at the beginning of the loop.

Exit control loops are the loop which condition is tasted at the end of the loop.

Do … while is an exit control loop. For and while are entry control loop.

Note : Exit control loops are executed at least once in their lifetime.

**Advantages of looping :**

Reduce the memory consumption.

Reduce the length of the code.

No need to write same code again to execute several times

**Following are the types of looping statements**

**while loop**

**do … while loop**

**for loop**

**nested loops**

## 1. While loop

While loop is an entry control looping statement that allows code to be executed until the boolean condition is true.

You can also call while loop as a repeating if statement.

Syntax :

...Assignment of counter variable

while(condition)

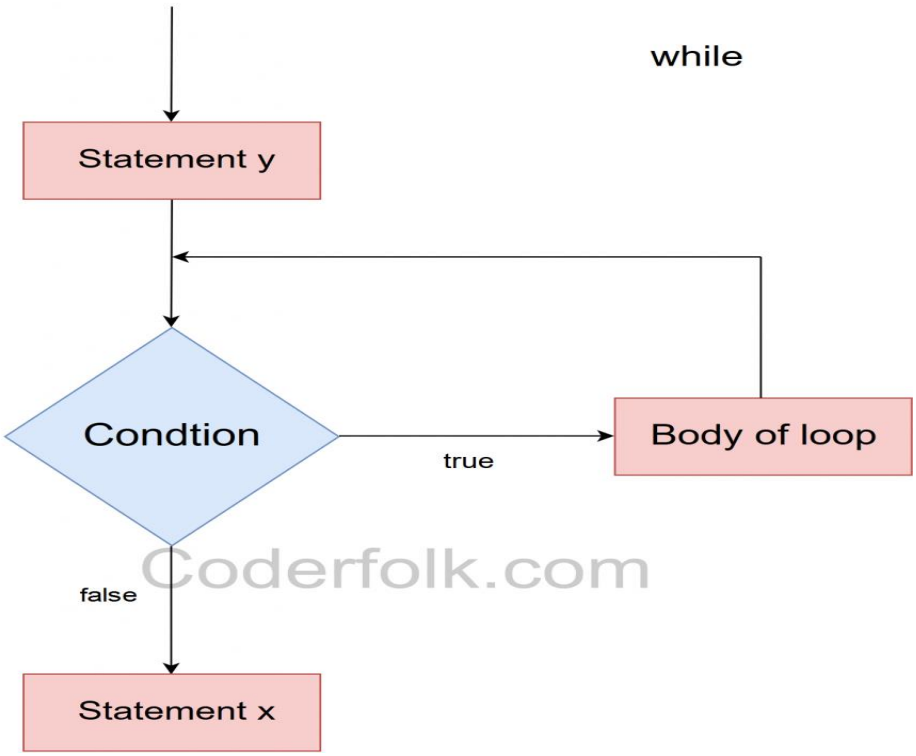
{

//body of loop

//increment or decrement of variable

}

Flowchart :




**while loop**

Example :

```
class WhileLoop{
	public static void main(String args[]){
		int i=0; // Assignment of counter variable
		while(i<10){
			//This loop will be executed 10 times.
			System.out.println(i);
			i++; //increment of counter variable
		}
	}
}
```

**Output :**

C:\workspace>notepad WhileLoop.java

```
C:\workspace>javac WhileLoop.java

C:\workspace>java WhileLoop

0

1

2

3

4

5

6

7

8

9

C:\workspace>
```

In above example, while loop is executed 10 times. Initially counter variable i is assigned by value 0 then the loop is executed 10 times and each time counter variable i is incremented by 1.

### 2. Do … while loop

Do … while is an exit control looping statement which is also used to execute a block of instructions several times. This loop is executed at least once in its lifetime whether the condition is true or false.

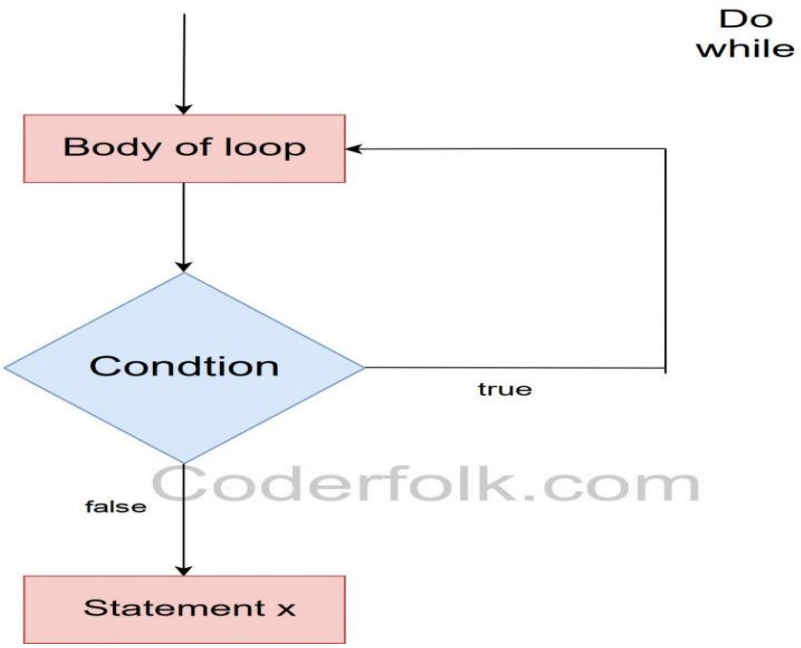
Syntax :

```
//Assignment of counter variable

do{

//body of statement

//increment or decrement of counter variable
```

}

while(condition); //You must put semicolon here

Flowchart :




**do while loop**

Example :

```
class DoWhileLoop{
        public static void main(String args[]){
                int i=0; // Assignment of counter variable
                do{
                        //this body will be executed at least once
                        System.out.println(i);
                        i++; //increment of counter variable
                }
                while(i<10);
        }
}
```

**Output :**

```
C:\workspace>javac DoWhileLoop.java

C:\workspace>java DoWhileLoop
0
1
2
3
4
5
6
7
8
9
C:\workspace>
```

In above example, Condition is tested at the end of the loop.

Note : You need to put a semicolon at the end of the do … while statement.

### 3. For loop

For loop is an entry control loop which works same as the other two loop but its syntax is different. In other two loop counter variable is declared outside the body of the loop.

Here, the counter variable is declared within for loop itself but the counter variable is declared and assigned by initial value only once.

Increment and condition checks will be done multiple times.

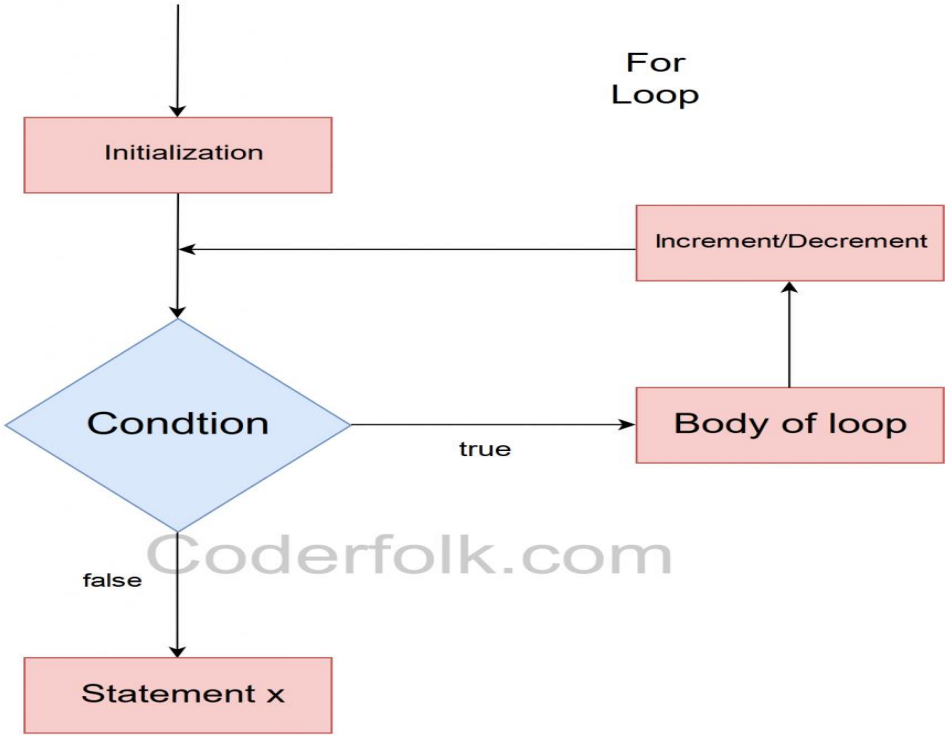
Syntax :

```
for(init; condition; increment or decrement){
```

//body of for loop

}

Flowchart :




**for loop**

Example :

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

**Output** :

C:\workspace>javac ForLoop.java

```
C:\workspace>java ForLoop
0
1
2
3
4
5
6
7
8
9
C:\workspace>
```

For loop is very easy to implement. Length of code is reduced than other loops :). In above program counter variable, i is declared and initialised only once. Body is executed 10 times.

For loop is also a kind of repetitive if statement in decision making.

### 4. Nested loops

The concept of a nested loop is as same as nested decision-making statement. One loop can contain another looping statement.

Example :

```
class NestedLoop{
    public static void main(String args[]){
        int j;
        for(int i=0;i<5;i++)
```

```
            {
                j=1;
                while(j<=i)
                {
                    System.out.print("*");
                    j++;
                }
                System.out.println();
            }
        }
}
```

**Output :**

```
C:\workspace>javac NestedLoop.java
C:\workspace>java NestedLoop
*
**
***
****
C:\workspace>
```