Loop in Java

For loop - Loops are used to execute a set of statements repeatedly until a particular condition is satisfied. In Java we have three types of basic loops: for, while and do-while. Let's see how to use "for loop" in Java.

Syntax of for loop:

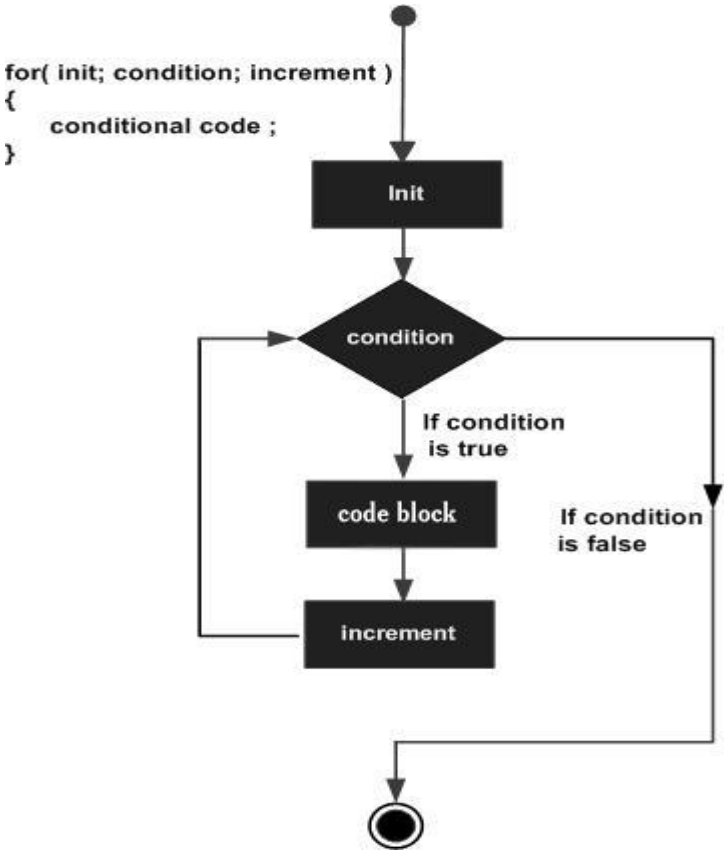
```
for(initialization; condition ; increment/decrement)
{
  statement(s);
}
```

```
for(int i=0; i< 5; i++ )
{
System.out.println(i); //
}
```

```
0
1
2
3
4
```

Flow of Execution of the for Loop

As a program executes, the interpreter always keeps track of which statement is about to be executed. We call this the control flow, or the flow of execution of the program.



First step: In for loop, initialization happens first and only one time, which means that the initialization part of for loop only executes once.

Second step: Condition in for loop is evaluated on each iteration, if the condition is true then the statements inside for loop body gets executed. Once the condition returns false, the statements in for loop does not execute and the control gets transferred to the next statement in the program after for loop.

Third step: After every execution of for loop's body,

Fourth step: The increment/decrement part of for loop executes that updates the loop counter. After third step, the control jumps to second step and condition is re-evaluated.

Example of Simple For loop

```
class ForLoopExample {
   public static void main(String args[]){
       for(int i=5; i>1; i--){
//int i=0;
 //for(;;){
            System.out.println("The value of i is: "+i);
       }
   }
}
```

The output of this program is:

```
The value of i is: 5
The value of i is: 4
The value of i is: 3
The value of i is: 2
```

In the above program:
int i=1 is initialization expression
i>1 is condition(Boolean expression)
i– Decrement operation

```
class ForLoopExample {
   public static void main(String args[]){
      int i=0;
       for(;;){
            System.out.println("The value of i is: "+i);
       }
   }
}
```

Infinite for loop - The importance of Boolean expression and increment/decrement operation co-ordination:

```
class ForLoopExample2 {
   public static void main(String args[]){
       for(int i=1; i>=1; i++){
            System.out.println("The value of i is: "+i);

       }
```

```
    }
}
```
This is an infinite loop as the condition would never return false. The initialization step is setting up the value of variable i to 1, since we are incrementing the value of i, it would always be greater than 1 (the Boolean expression: i>1) so it would never return false. This would eventually lead to the infinite loop condition. Thus it is important to see the co-ordination between Boolean expression and increment/decrement operation to determine whether the loop would terminate at some point of time or not.

Here is another example of infinite for loop:

```
// infinite loop
for ( ; ; ) {
    // statement(s)
}
```

For loop example to iterate an array:

Here we are iterating and displaying array elements using the for loop.

```
class ForLoopExample3 {
    public static void main(String args[]){
        int arr[]={2,11,45,9};
        //i starts with 0 as array index starts with 0 too
        for(int i=0; i<arr.length; i++){
            System.out.println(arr[i]);
        }
    }
}
```
Output:
```
2
11
45
9
```

Enhanced For loop

Enhanced for loop is useful when you want to iterate Array/Collections, it is easy to write and understand.

```
class ForLoopExample3 {
   public static void main(String args[]){
      int arr[]={2,11,45,9};
      for (int num : arr) {
         System.out.println(num);
      }
   }
}
```
Output:
```
2
11
45
9
```
Note: In the above example, I have declared the num as int in the enhanced for loop. This will change depending on the data type of array. For example, the enhanced for loop for string type would look like this:

```
String arr[]={"hi","hello","bye"};
```

```
for (String str : arr) {
        System.out.println(str);
}
```

Assignments-

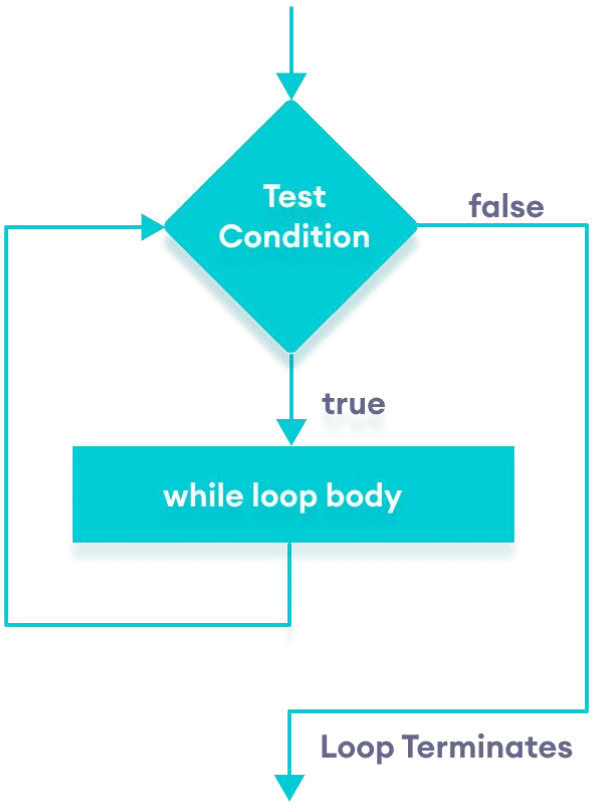
Java Program to find sum of natural numbers using for loop
Java Program to find factorial of a number using loops
Java Program to print Fibonacci Series using for loop

While loop- As we know, loops are used to execute a set of statements repeatedly until a particular condition is satisfied.
Syntax of while loop

```
while(condition)
{
  statement(s);
}
```

How while Loop works?

In while loop, condition is evaluated first and if it returns true then the statements inside while loop execute. When condition returns false, the control comes out of loop and jumps to the next statement after while loop.
Note: The important point to note when using while loop is that we need to use increment or decrement statement inside while loop so that the loop variable gets changed on each iteration, and at some point condition returns false. This way we can end the execution of while loop otherwise the loop would execute indefinitely.



while loop example

```
class WhileLoopExample {
   public static void main(String args[]){
       int i=10;
       while(i>5){
           System.out.println(i);
           i--;
       }
   }
}
```

Output:

```
10
9
8
7
6
```

Infinite while loop

```
class WhileLoopExample2 {
   public static void main(String args[]){
       int i=10;
       while(i>1)
       {
          System.out.println(i);
           i++;
       }
   }
}
```

This loop would never end, its an infinite while loop. This is because condition is i>1 which would always be true as we are incrementing the value of i inside while loop.
Here is another example of infinite while loop:

```
while (true){
   statement(s);
}
```

Iterating an array using while loop

Here we are iterating and displaying array elements using while loop.

```
class WhileLoopExample3 {
   public static void main(String args[]){
       int arr[]={2,11,45,9};
       //i starts with 0 as array index starts with 0 too
       int i=0;
       while(i<4){
           System.out.println(arr[i]);
           i++;
       }
   }
```

```
}
```
Output:
2
11
45
9

Assignments:

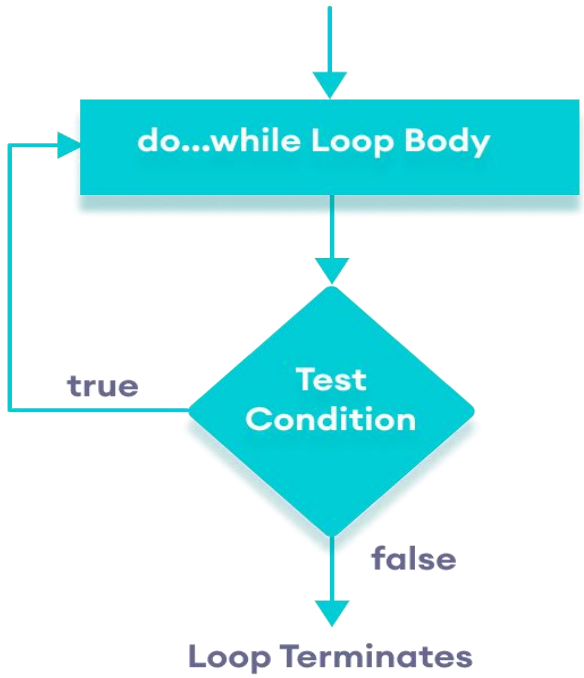
Java Program to display Fibonacci Series using while loop
Java Program to find factorial using while loop

do-while loop - do-while loop is similar to while loop, however there is a difference between them: In while loop, condition is evaluated before the execution of loop's body but in do-while loop condition is evaluated after the execution of loop's body.
Syntax of do-while loop:

```
do
{
  statement(s);
} while(condition);
```

How do-while loop works?

First, the statements inside loop execute and then the condition gets evaluated, if the condition returns true then the control gets transferred to the "do" else it jumps to the next statement after do-while.



do-while loop example

```
class DoWhileLoopExample {
    public static void main(String args[]){
        int i=6;
```

```
        do{
            System.out.println(i);
            i--;
        }while(i>1);
    }
}
```

Output:

```
6
5
4
3
2
```

Iterating array using do-while loop

Here we have an integer array and we are iterating the array and displaying each element using do-while loop.

```
class DoWhileLoopExample2 {
    public static void main(String args[]){
        int arr[]={2,11,45,9};
        //i starts with 0 as array index starts with 0
        int i=0;
        do{
            System.out.println(arr[i]);
            i++;
        }while(i<4);
    }
}
```

Output:

```
2
11
45
9
```

Nested Loop-

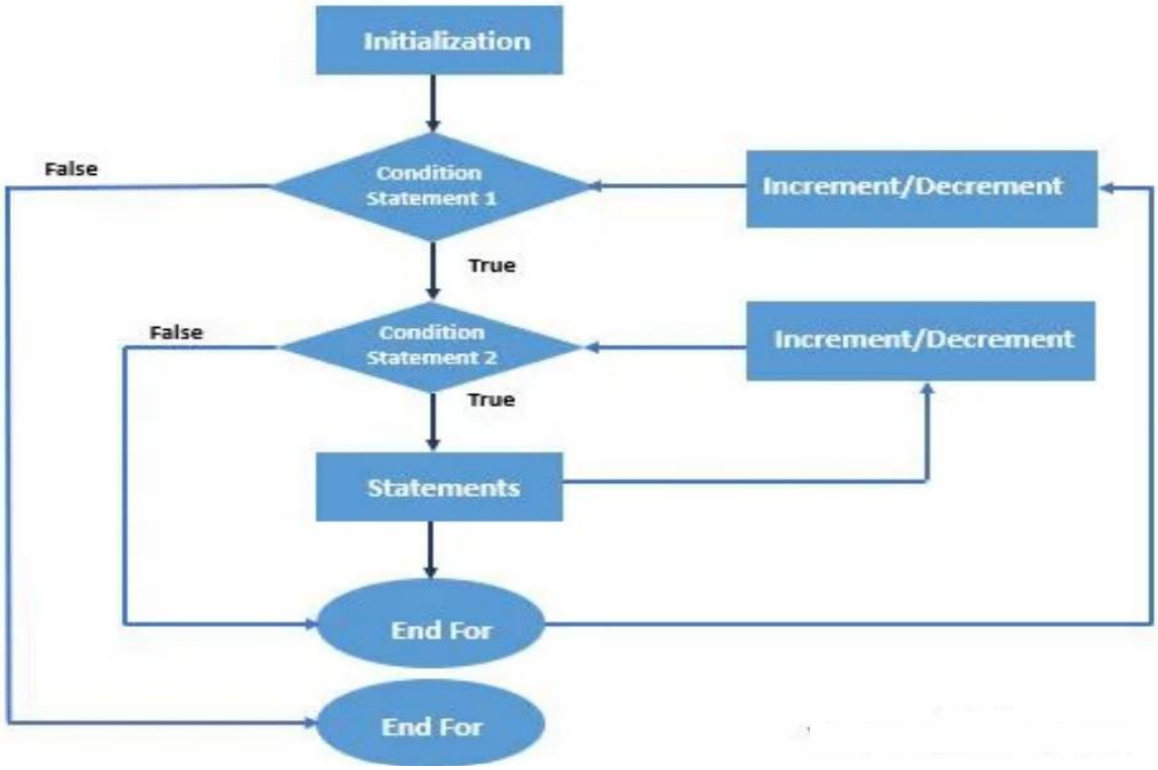
```
    public class NestedForExample {
    public static void main(String[] args) {
    //loop of i
    for(int i=1;i<=3;i++){
    //loop of j
    for(int j=1;j<=3; j++){
                    System.out.println(i+" "+j);
    }//end of j
    }//end of i
    }
    }
```

Output-

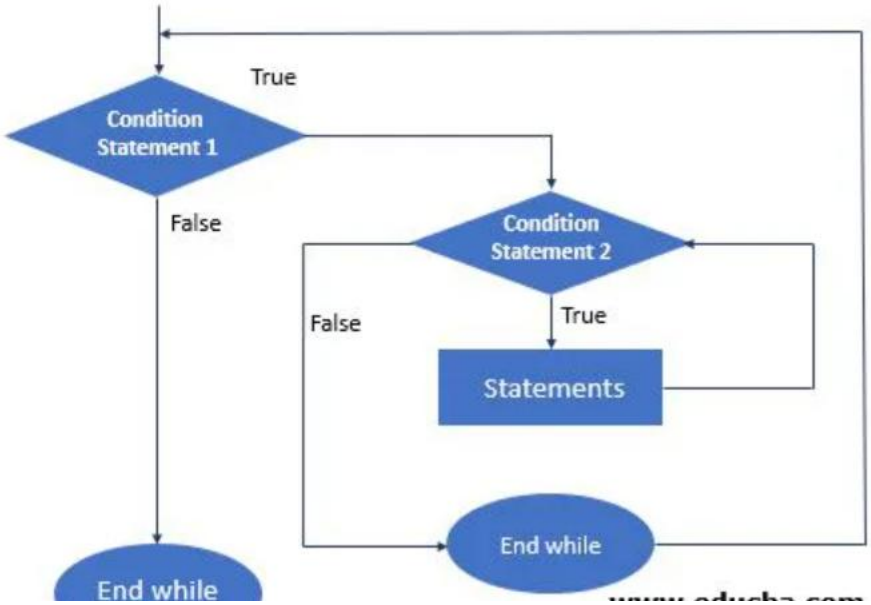
```
    1 1
    1 2
    1 3
```

2 1
2 2
2 3
3 1
3 2
3 3

Flow char of Nested For Loop



Flow chart of nested while loop



Flow Chart of Nested do while Loop