Loops in Java

In programming languages, loops are used to execute a set of instructions/functions repeatedly when some conditions become true. There are three types of loops in Java.

- for loop
- while loop
- do-while loop

Java For Loop

The Java *for loop* is used to iterate a part of the program several times. If the number of iteration is fixed, it is recommended to use for loop.

There are three types of for loops in java.

- Simple For Loop
- For-each or Enhanced For Loop
- Labeled For Loop

Java Simple For Loop

A simple for loop is the same as C/C++. We can initialize the variable, check condition and increment/decrement value. It consists of four parts:

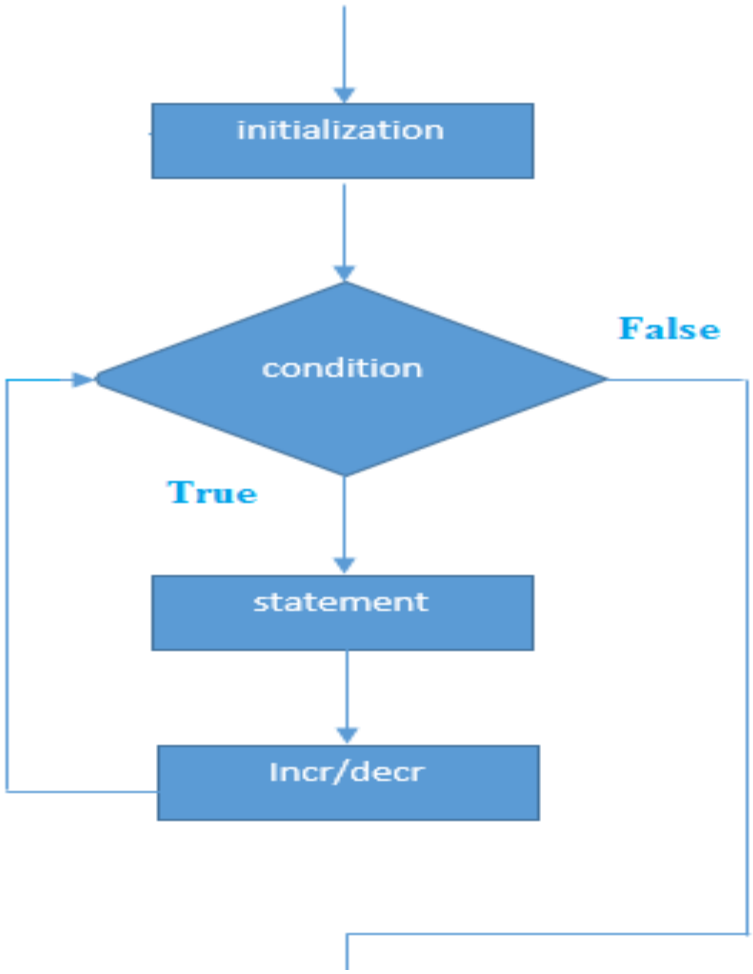
1. **Initialization**: It is the initial condition which is executed once when the loop starts. Here, we can initialize the variable, or we can use an already initialized variable. It is an optional condition.
2. **Condition**: It is the second condition which is executed each time to test the condition of the loop. It continues execution until the condition is false. It must return boolean value either true or false. It is an optional condition.
3. **Statement**: The statement of the loop is executed each time until the second condition is false.

4. **Increment/Decrement**: It increments or decrements the variable value. It is an optional condition.

Syntax:

```
for(initialization;condition;incr/decr){
//statement or code to be executed
}
```

Flowchart:



Example:

```
//Java Program to demonstrate the example of for loop
```

```
//which prints table of 1
public class ForExample {
public static void main(String[] args) {
   //Code of Java for loop
   for(int i=1;i<=10;i++){
      System.out.println(i);
   }
}
}
```

Output:

```
1
2
3
4
5
6
7
8
9
10
```

Java Nested For Loop

If we have a for loop inside the another loop, it is known as nested for loop. The inner loop executes completely whenever outer loop executes.

Example:

```
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 i
```

```
}//end of j

}

}
```

Output:

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

Pyramid Example 1:

```
public class PyramidExample {

public static void main(String[] args) {

for(int i=1;i<=5;i++){

for(int j=1;j<=i;j++){

        System.out.print("* ");

}

System.out.println();//new line

}

}

}
```

Output:

```
*
* *
* * *
* * * *
* * * * *
```

Pyramid Example 2:

```
public class PyramidExample2 {
```

```
public static void main(String[] args) {
int term=6;
for(int i=1;i<=term;i++){
for(int j=term;j>=i;j--){
        System.out.print("* ");
}
System.out.println();//new line
}
}
}
```

Output:

```
* * * * * *
* * * * *
* * * *
* * *
* *
*
```

Java for-each Loop

The for-each loop is used to traverse array or collection in java. It is easier to use than simple for loop because we don't need to increment value and use subscript notation.

It works on elements basis not index. It returns element one by one in the defined variable.

Syntax:

```
for(Type var:array){
//code to be executed
}
```

Example:

```
//Java For-each loop example which prints the
//elements of the array
```

```
public class ForEachExample {
public static void main(String[] args) {
    //Declaring an array
    int arr[]={12,23,44,56,78};
    //Printing array using for-each loop
    for(int i:arr){
        System.out.println(i);
    }
}
}
```

Output:

```
12
23
44
56
78
```

Java Infinitive For Loop

If you use two semicolons ;; in the for loop, it will be infinitive for loop.

Syntax:

```
for(;;){
//code to be executed
}
```

Example:

```
//Java program to demonstrate the use of infinite for loop
//which prints an statement
public class ForExample {
public static void main(String[] args) {
   //Using no condition in for loop
   for(;;){
```

```
      System.out.println("infinitive loop");
   }
}
}
```

Output:

```
infinitive loop
infinitive loop
infinitive loop
infinitive loop
infinitive loop
ctrl+c
```

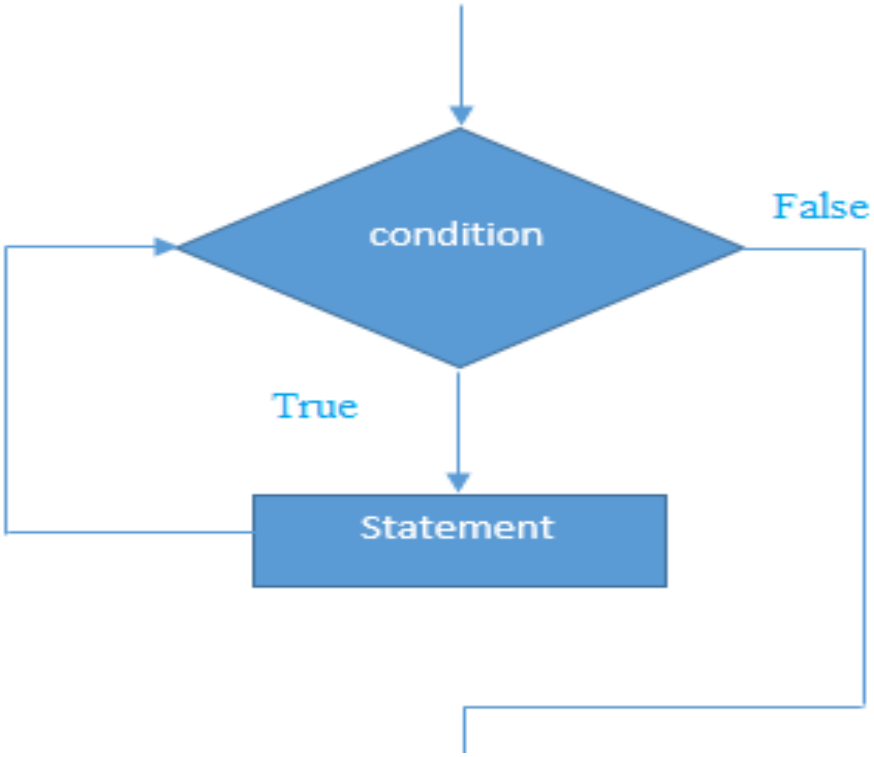
Now, you need to press ctrl+c to exit from the program

Java While Loop

The Java *while loop* is used to iterate a part of the program several times. If the number of iteration is not fixed, it is recommended to use while loop.

Syntax:

```
while(condition){
//code to be executed
}
```



Example:

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

Output:

```
1
2
3
4
5
6
```

```
7
8
9
10
```

Java Infinitive While Loop

If you pass **true** in the while loop, it will be infinitive while loop.

Syntax:

```
while(true){
//code to be executed
}
```

Example:

```
public class WhileExample2 {
public static void main(String[] args) {
   while(true){
      System.out.println("infinitive while loop");
   }
}
}
```

Output:

```
infinitive while loop
infinitive while loop
infinitive while loop
infinitive while loop
infinitive while loop
ctrl+c
```

Now, you need to press ctrl+c to exit from the program.

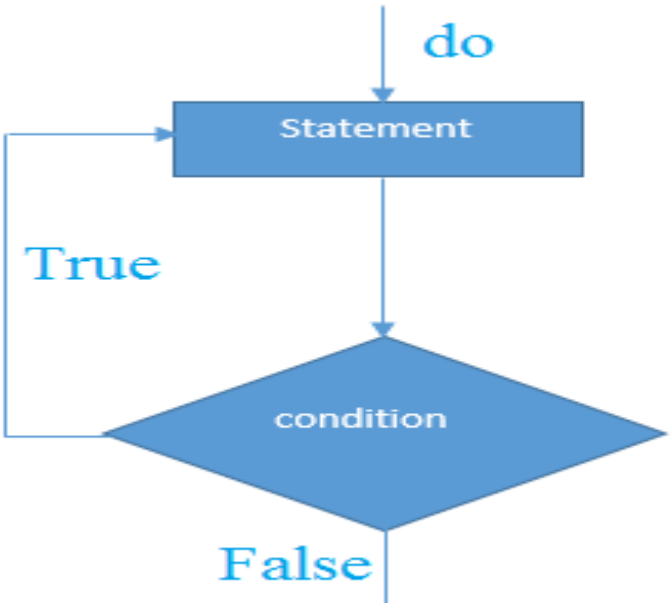
Java do-while Loop

The Java *do-while loop* is used to iterate a part of the program several times. If the number of iteration is not fixed and you must have to execute the loop at least once, it is recommended to use do-while loop.

The Java *do-while loop* is executed at least once because condition is checked after loop body.

Syntax:

```
do{
//code to be executed
}while(condition);
```



Example:

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

Output:

```
1
2
3
4
5
6
7
8
9
10
```

Java Infinitive do-while Loop

If you pass **true** in the do-while loop, it will be infinitive do-while loop.

Syntax:

```
do{
//code to be executed
}while(true);
```

Example:

```
public class DoWhileExample2 {
public static void main(String[] args) {
   do{
       System.out.println("infinitive do while loop");
  }while(true);
}
}
```

Output:

```
infinitive do while loop
infinitive do while loop
infinitive do while loop
ctrl+c
```

Now, you need to press ctrl+c to exit from the program.

Java Break Statement

When a break statement is encountered inside a loop, the loop is immediately terminated and the program control resumes at the next statement following the loop.

The Java *break* statement is used to break loop or switch statement. It breaks the current flow of the program at specified condition. In case of inner loop, it breaks only inner loop.

We can use Java break statement in all types of loops such as for loop, while loop and do-while loop.

Syntax:

```
jump-statement;
break;
```

Java Break Statement with Loop

Example:

```
//Java Program to demonstrate the use of break statement
//inside the for loop.
public class BreakExample {
public static void main(String[] args) {
   //using for loop
   for(int i=1;i<=10;i++){
      if(i==5){
         //breaking the loop
         break;
      }
      System.out.println(i);
   }
}
```

```
}
```

Output:

```
1
2
3
4
```

Java Break Statement with Inner Loop

It breaks inner loop only if you use break statement inside the inner loop.

Example:

```
//Java Program to illustrate the use of break statement
//inside an inner loop
public class BreakExample2 {
public static void main(String[] args) {
        //outer loop
        for(int i=1;i<=3;i++){
            //inner loop
            for(int j=1;j<=3;j++){
               if(i==2&&j==2){
                  //using break statement inside the inner loop
                  break;
               }
               System.out.println(i+" "+j);
            }
        }
}
}
```

Output:

```
1 1
1 2
```

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

Java Break Statement in while loop

Example:

```
//Java Program to demonstrate the use of break statement
//inside the while loop.
public class BreakWhileExample {
public static void main(String[] args) {
   //while loop
   int i=1;
   while(i<=10){
      if(i==5){
         //using break statement
         i++;
       break;//it will break the loop
      }
      System.out.println(i);
      i++;
   }
}
}
```

Output:

```
1
2
3
4
```

Java Break Statement in do-while loop

Example:

```
//Java Program to demonstrate the use of break statement
//inside the Java do-while loop.
public class BreakDoWhileExample {
public static void main(String[] args) {
    //declaring variable
    int i=1;
    //do-while loop
    do{
        if(i==5){
          //using break statement
          i++;
          break;//it will break the loop
        }
        System.out.println(i);
        i++;
  }while(i<=10);
}
}
```

Output:

```
1
2
3
4
```

Java Continue Statement

The continue statement is used in loop control structure when you need to jump to the next iteration of the loop immediately. It can be used with for loop or while loop.

The Java *continue statement* is used to continue the loop. It continues the current flow of the program and skips the remaining code at the specified condition. In case of an inner loop, it continues the inner loop only.

We can use Java continue statement in all types of loops such as for loop, while loop and do-while loop.

Syntax:

```
1.        jump-statement;
2.        continue;
```

Java Continue Statement Example

Example:

```
//Java Program to demonstrate the use of continue statement
//inside the for loop.
public class ContinueExample {
public static void main(String[] args) {
   //for loop
   for(int i=1;i<=10;i++){
     if(i==5){
         //using continue statement
         continue;//it will skip the rest statement
      }
      System.out.println(i);
   }
}
}
```

Output:

```
1
2
3
4
6
7
8
9
10
```

Java Continue Statement in while loop

Example:

```
//Java Program to demonstrate the use of continue statement
//inside the while loop.
public class ContinueWhileExample {
public static void main(String[] args) {
   //while loop
   int i=1;
   while(i<=10){
      if(i==5){
         //using continue statement
         i++;
         continue;//it will skip the rest statement
     }
      System.out.println(i);
      i++;
   }
}
}
```

Output:

```
1
2
3
4
6
7
8
9
10
```

Java Continue Statement in do-while loop

Example:

```java
//Java Program to demonstrate the use of continue statement
//inside the Java do-while loop.
public class ContinueDoWhileExample {
public static void main(String[] args) {
    //declaring variable
    int i=1;
    //do-while loop
    do{
        if(i==5){
                //using continue statement
                 i++;
            continue;//it will skip the rest statement
        }
       System.out.println(i);
        i++;
    }while(i<=10);
}
}
```

Output:

```
1
2
3
4
6
7
8
9
10
```