# Loops in Java

In programming languages, loops are used to execute a set of instructions/functions repeatedly until the stated condition becomes false.

## Need of Loops in Java

Imagine a program which is required to output a particular value of a variable 800 times. Now we all know that the code for writing output is System.out.println(“Text”); But in order to print this 800 times we will need to write the same line 800 times in the code. That would take up a lot of effort which is particularly just copy pasting the same sentence 800 times. Let us say that you have managed to copy paste the entire thing easily. Now if there is a different program which requires you to print the first 800 natural numbers. The copy paste method would not work because you still would have to go to all these lines and fit a number. That’s where loops come in to play. Loops make it very easy to group all the code that’s needed to be repetitively processed and throw it under scope. The loop does the remaining job.

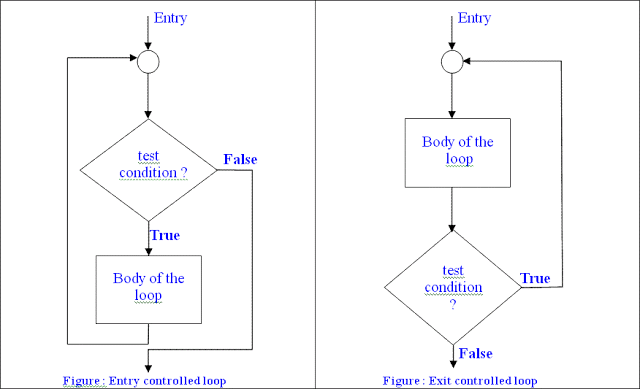
Depending upon the position of a control statement in a program, looping is classified into two types

1. Entry controlled loop

2. Exit controlled loop

In an **entry controlled loop,** a condition is checked before executing the body of a loop. It is also called as a pre-checking loop.

In an **exit controlled loop**, a condition is checked after executing the body of a loop. It is also called as a post-checking loop.



Comparison Between Entry Controlled And Exit Controlled Loop

|  |  |  |
| --- | --- | --- |
| **BASIS OF COMPARISON** | **ENTRY CONTROLLED LOOP** | **EXIT CONTROLLED LOOP** |
| **Description** | Entry controlled loop is a loop in which the test condition is checked first, and then loop body will be executed. | Exit controlled loop is a loop in which the loop body is executed first and then the given condition is checked afterwards. |
| **False Test Condition** | If the test condition is false, loop body will not be executed. | If the test condition is false, loop body will be executed at least once. |
| **Example** | Examples of exit controlled loop include, **For Loop and While Loop.** | An example of exit controlled loop is **Do While Loop.** |
| **Use** | Entry controlled loop are used when checking of test condition is mandatory before executing loop body. | Entry controlled loop is used when checking of test condition is mandatory after executing. |

The types of loops in Java.

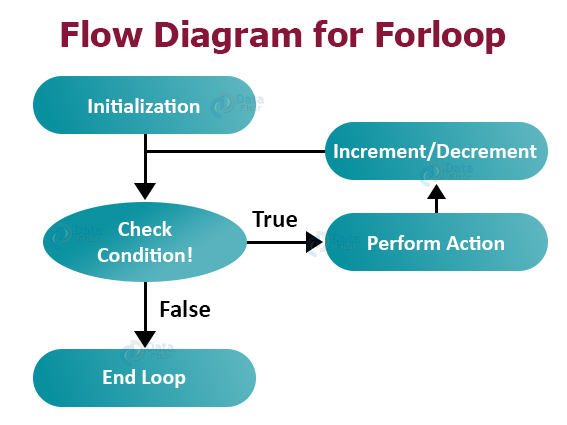
* for loop
* Enhanced For Loop
* [while loop](https://www.javatpoint.com/java-while-loop)
* [do-while loop](https://www.javatpoint.com/java-do-while-loop)
* Nested for loop

### 1. For loop in Java

Java for loop consists of 3 primary factors which define the loop itself. These are the initialization statement, a testing condition, an increment or decrement part for incrementing/decrementing the control variable.

**The basic syntax of java for loop :**

for(initializing statement;testing condition;increment/decrement)  
{  
//code to be iterated  
}



Here

1.The **initialExpression** initializes and/or declares variables and executes only once.

2.The **condition** is evaluated. If the **condition** is true, the body of the for loop is executed

3.The **updateExpression(increment / decrement)** updates the value of **initialExpression**

4.The **condition** is evaluated again. The process continues until the **condition** is false

# 2.Java For-each Loop | Enhanced For Loop

The Java for-each loop or enhanced for loop is introduced since Java 2 Special Edition 5.0. It provides an alternative approach to traverse the array or collection in Java. It is mainly used to traverse the array or collection elements.

The advantage of the for-each loop is that it eliminates the possibility of bugs and makes the code more readable. It is known as the for-each loop because it traverses each element one by one.

The drawback of the enhanced for loop is that it cannot traverse the elements in reverse order. Here, you do not have the option to skip any element because it does not work on an index basis. Moreover, you cannot traverse the odd or even elements only.

But, it is recommended to use the Java for-each loop for traversing the elements of array and collection because it makes the code readable.

**Syntax of Enhanced for loop:**

**for**(data\_type variable : array | collection){

//body of for-each loop

}

### 3.While loop in Java

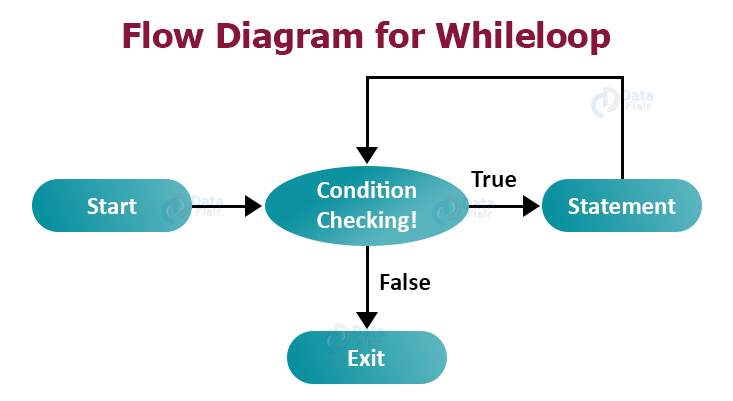
While loops are very important as we cannot know the extent of a loop everytime we define one. For example if we are asked to take a dynamic collection and asked to iterate through every element, for loops would be impossible to use because we do not know the size of the collection. Then we would have to use an enhanced for loop or a while loop.

A while loop iterates through a set of statements till its boolean condition returns false. As long as the condition given evaluates to true, the loop iterates.

The condition of the loop structure is checked at first and then the control proceeds into the loop structure only if the condition evaluates to true. Hence it is called an entry-controlled loop. The body of the loop generally contains a variable which controls the boolean condition mentioned.

**The basic syntax of Java while loop is:**

while(boolean condition)  
{  
//statements;  
}



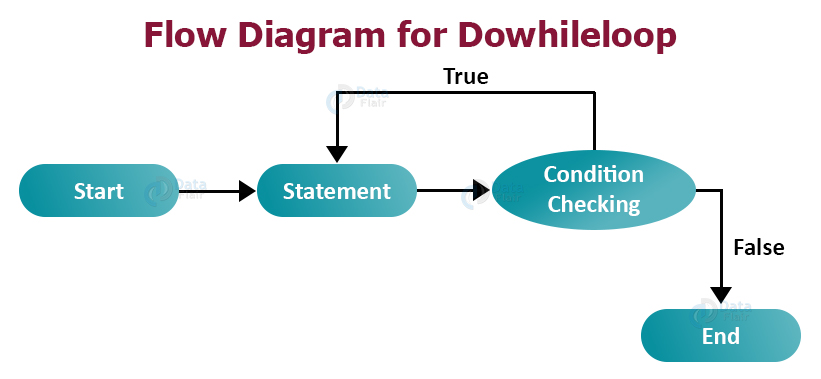
Here,

1. A while loop evaluates the **textExpression** inside the parenthesis ().
2. If the **textExpression** evaluates to true, the code inside the while loop is executed.
3. The **textExpression** is evaluated again.
4. This process continues until the **textExpression** is false.
5. When the **textExpression** evaluates to false, the loop stops.

### 4. do while loop in Java

Java do while loop executes the statement first and then checks for the condition. Other than that it is similar to the while loop. The difference lies in the fact that if the condition is true at the starting of the loop the statements would still be executed, however in case of while loop it would not be executed at all.

This is an exit-controlled loop because of the fact that it checks the condition after the statements inside it are executed.



#### Syntax of do-while loop:

do

{

statement(s);

} while(condition);

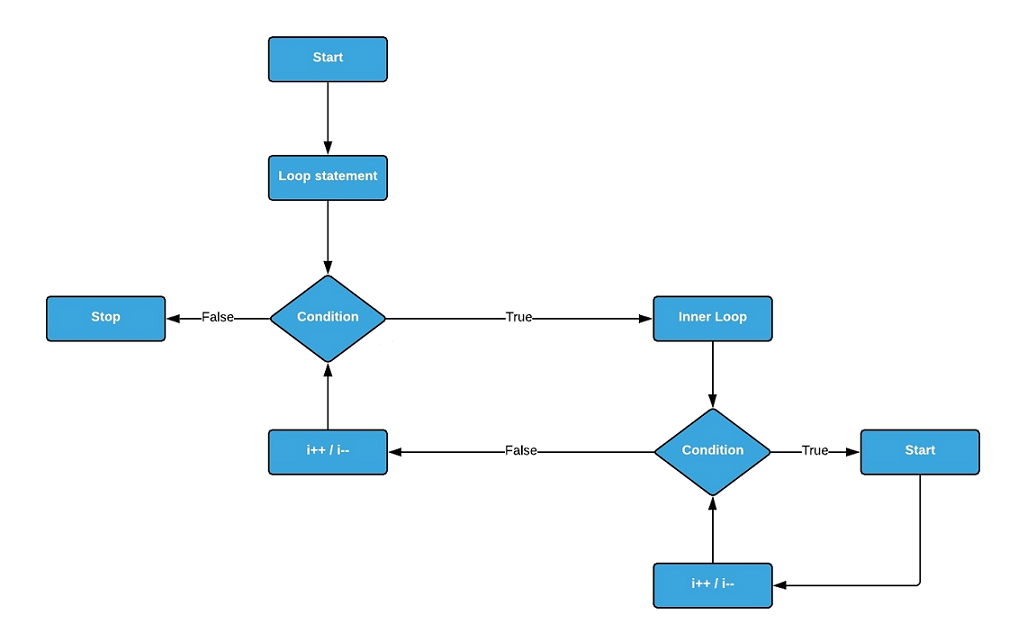
5.Java Nested For Loop

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.

As the name suggests, nested loops are basically one loop functioning inside another one. After the first iteration of the outer loop starts, the inner loop starts. As soon as the innerloop finishes it’s iterations and exits, the first iteration of the outer loop completes and then it goes for the second iteration. This keeps on repeating till the outermost loop finishes its iterations.

However nested loops doesn’t necessarily mean two loops. You can include as many loops as you want inside one another. If there are two loops one inside another, one of them having **N** iterations and the other one having **M** iterations. Then the total number of iterations would be **M x N.**

**Basic syntax for a for loop inside a for loop:**

for(initializer;condition;increment/decrement)  
{  
for(initializer;condition;increment/decrement)  
{  
//code to be nested  
}  
}

**Comparison Between for loop ,while loop, and do-while loop:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Comparison** | **for loop** | **While loop** | **do while loop** |
| **Initialization** | Variable Initialization may be either in loop statement or outside the loop. | Variable in condition is initialized before the execution of loop. | variable may be initialized before or within the loop. |
| **Condition** | Condition is checked first then statement(s) is executed.. | Condition is checked first then statement(s) is executed. | Statement(s) is executed atleast once, there after condition is checked. |
| Condition is a relational expression. | Condition may be expression or non-zero value. | Condition may be expression or non-zero value. |
| **Brackets** | If there is a single statement, brackets are not required. | If there is a single statement, brackets are not required. | If there is a single statement, brackets are not required. |
| **Semicolon** | No semicolon at the end of for. for(condition) | No semicolon at the end of while. while(condition) | Semicolon at the end of while. while(condition)**;** |
| **When to use** | If the number of iteration is fixed, it is recommended to use for loop. | If the number of iteration is not fixed, it is recommended to use while loop. | If the number of iteration is not fixed and you must have to execute the loop at least once, it is recommended to use the do-while loop. |
| **looping** | for loop is entry controlled loop | while loop is entry controlled loop | . do-while is exit controlled loop. |
| **Syntax** | for ( init ; condition ; iteration ) { statement (s); } | while(condition) { statement(s); } | do { statement(s); } while (condition); |
| **Example** | //for loop  for(int i=1;i<=10;i++){  System.out.println(i);  } | //while loop  int i=1;  while(i<=10){  System.out.println(i);  i++;  } | //do-while loop  int i=1;  do{  System.out.println(i);  i++;  }while(i<=10); |
| **Syntax for infinite loop** | for(;;){  //code to be executed  } | while(true){  //code to be executed  } | do{  //code to be executed  }while(true); |