

[Write an Article](#)[Login](#)

Loops in Java

Looping in programming languages is a feature which facilitates the execution of a set of instructions/functions repeatedly while some condition evaluates to true.

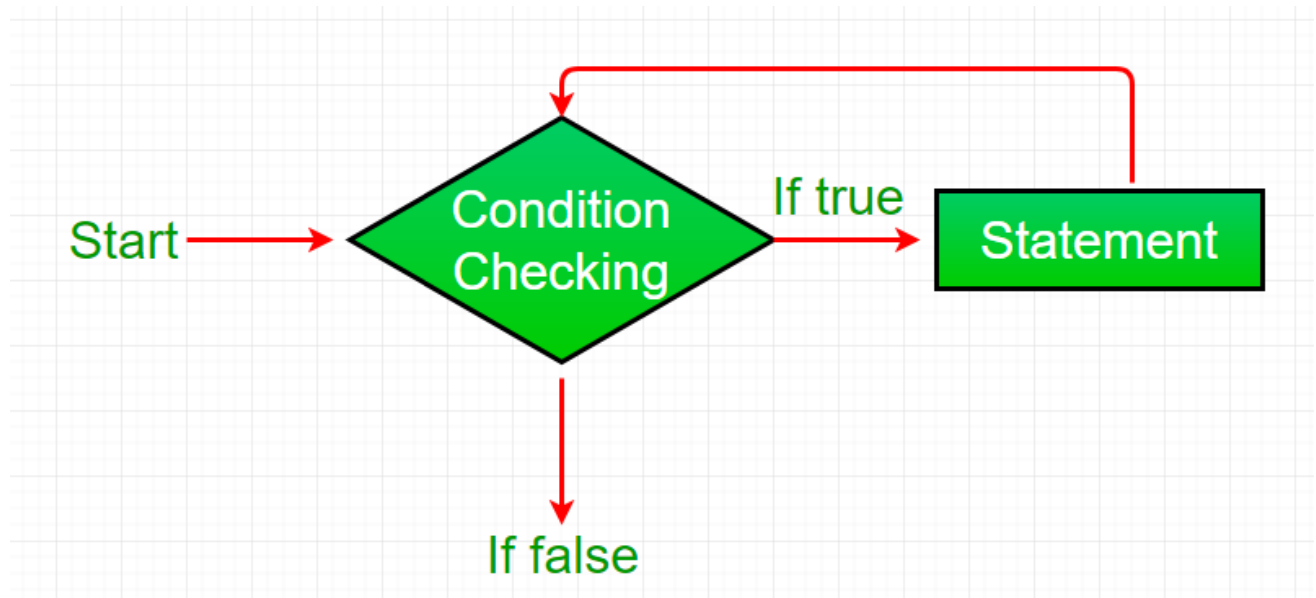
Java provides three ways for executing the loops. While all the ways provide similar basic functionality, they differ in their syntax and condition checking time.

1. **while loop:** A while loop is a control flow statement that allows code to be executed repeatedly based on a given Boolean condition. The while loop can be thought of as a repeating if statement.

Syntax :

```
while (boolean condition)
{
    loop statements...
}
```

Flowchart:



- While loop starts with the checking of condition. If it evaluated to true, then the loop body statements are executed otherwise first statement following the loop is executed. For this reason it is also called **Entry control loop**
- Once the condition is evaluated to true, the statements in the loop body are executed. Normally the statements contain an update value for the variable being processed for the

next iteration.

- When the condition becomes false, the loop terminates which marks the end of its life cycle.

```
// Java program to illustrate while loop
class whileLoopDemo
{
    public static void main(String args[])
    {
        int x = 1;

        // Exit when x becomes greater than 4
        while (x <= 4)
        {
            System.out.println("Value of x:" + x);

            // Increment the value of x for
            // next iteration
            x++;
        }
    }
}
```

[Run on IDE](#)

Output:

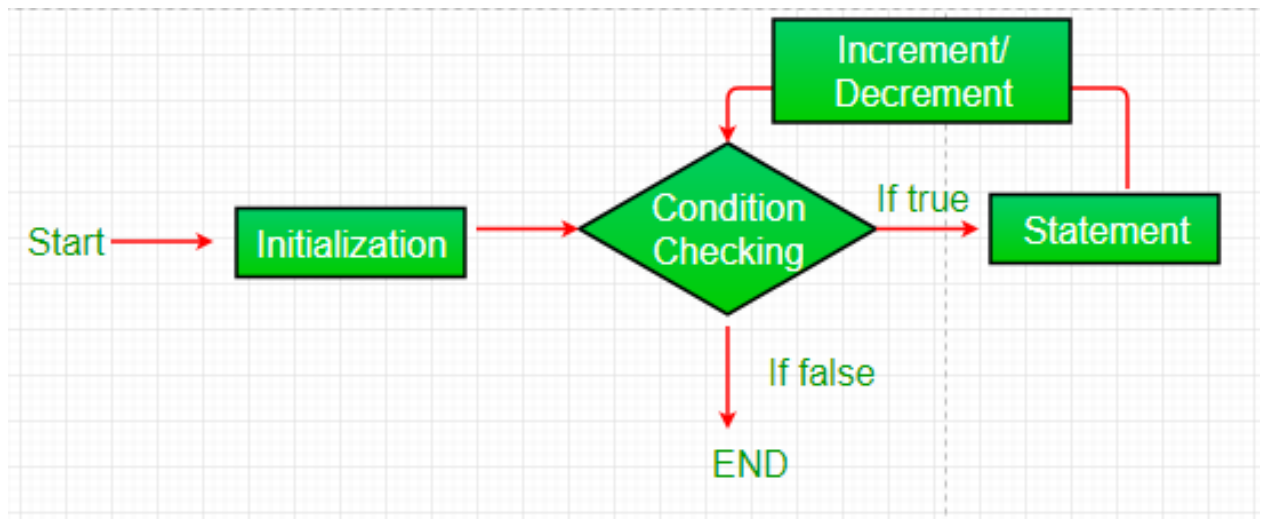
```
Value of x:1
Value of x:2
Value of x:3
Value of x:4
```

2. **for loop:** for loop provides a concise way of writing the loop structure. Unlike a while loop, a for statement consumes the initialization, condition and increment/decrement in one line thereby providing a shorter, easy to debug structure of looping.

Syntax:

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

Flowchart:



1. **Initialization condition:** Here, we initialize the variable in use. It marks the start of a for loop. An already declared variable can be used or a variable can be declared, local to loop only.
2. **Testing Condition:** It is used for testing the exit condition for a loop. It must return a boolean value. It is also an **Entry Control Loop** as the condition is checked prior to the execution of the loop statements.
3. **Statement execution:** Once the condition is evaluated to true, the statements in the loop body are executed.
4. **Increment/ Decrement:** It is used for updating the variable for next iteration.
5. **Loop termination:** When the condition becomes false, the loop terminates marking the end of its life cycle.

```
// Java program to illustrate for loop.
class forLoopDemo
{
    public static void main(String args[])
    {
        // for loop begins when x=2
        // and runs till x <=4
        for (int x = 2; x <= 4; x++)
            System.out.println("Value of x:" + x);
    }
}
```

[Run on IDE](#)

Output:

```
Value of x:2
Value of x:3
Value of x:4
```

Enhanced For loop

Java also includes another version of for loop introduced in Java 5. Enhanced for loop provides a simpler way to iterate through the elements of a collection or array. It is inflexible and should be used only when there is a need to iterate through the elements in sequential manner without knowing the index of currently processed element.

Syntax:

```
for (T element:Collection obj/array)
{
    statement(s)
}
```

Lets take an example to demonstrate how enhanced for loop can be used to simplify the work. Suppose there is an array of names and we want to print all the names in that array. Let's see the difference with these two examples

Enhanced for loop simplifies the work as follows-

```
// Java program to illustrate enhanced for loop
public class enhancedforloop
{
    public static void main(String args[])
    {
        String array[] = {"Ron", "Harry", "Hermoine"};

        //enhanced for loop
        for (String x:array)
        {
            System.out.println(x);
        }

        /* for loop for same function
        for (int i = 0; i < array.length; i++)
        {
            System.out.println(array[i]);
        }
        */
    }
}
```

[Run on IDE](#)

Output:

```
Ron
Harry
Hermoine
```

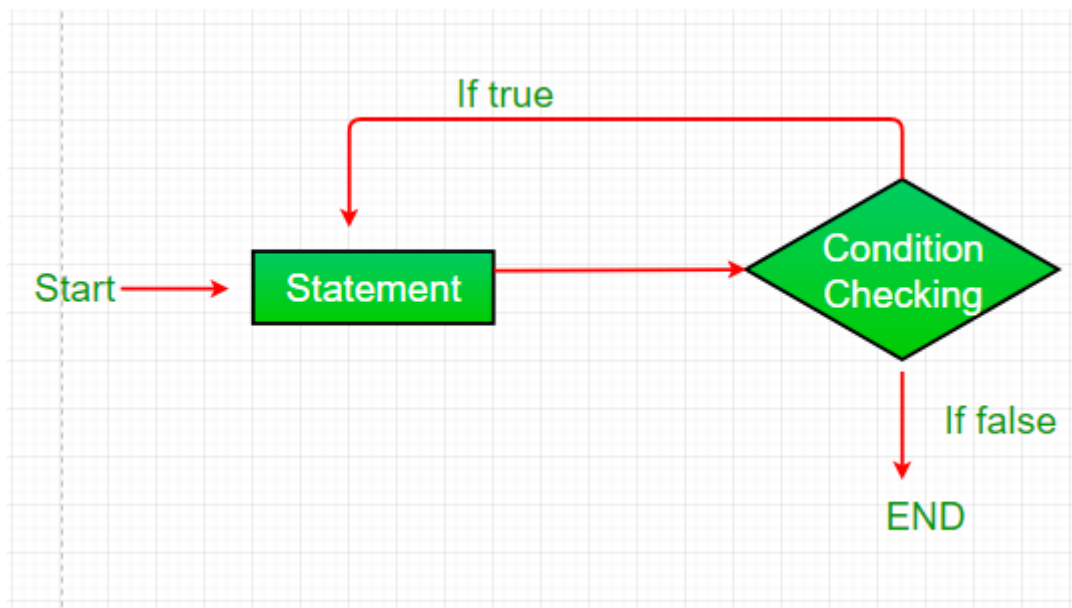
3. **do while:** do while loop is similar to while loop with only difference that it checks for condition after executing the statements, and therefore is an example of **Exit Control Loop**.

Syntax:

```
do
{
    statements..
}
```

```
}  
while (condition);
```

Flowchart:



1. do while loop starts with the execution of the statement(s). There is no checking of any condition for the first time.
2. After the execution of the statements, and update of the variable value, the condition is checked for true or false value. If it is evaluated to true, next iteration of loop starts.
3. When the condition becomes false, the loop terminates which marks the end of its life cycle.
4. It is important to note that the do-while loop will execute its statements atleast once before any condition is checked, and therefore is an example of exit control loop.

```
// Java program to illustrate do-while loop  
class dowhileloopDemo  
{  
    public static void main(String args[])  
    {  
        int x = 21;  
        do  
        {  
            // The line will be printed even  
            // if the condition is false  
            System.out.println("Value of x:" + x);  
            x++;  
        }  
        while (x < 20);  
    }  
}
```

[Run on IDE](#)

Output:

```
Value of x: 21
```

Pitfalls of Loops

1. **Infinite loop:** One of the most common mistakes while implementing any sort of looping is that that it may not ever exit, that is the loop runs for infinite time. This happens when the condition fails for some reason.

Examples:

```
//Java program to illustrate various pitfalls.
public class LooppitfallsDemo
{
    public static void main(String[] args)
    {
        // infinite loop because condition is not apt
        // condition should have been i>0.
        for (int i = 5; i != 0; i -= 2)
        {
            System.out.println(i);
        }
        int x = 5;

        // infinite loop because update statement
        // is not provided.
        while (x == 5)
        {
            System.out.println("In the loop");
        }
    }
}
```

[Run on IDE](#)

2. Another pitfall is that you might be adding something into you collection object through loop and you can run out of memory. If you try and execute the below program, after some time, out of memory exception will be thrown.

```
//Java program for out of memory exception.
import java.util.ArrayList;
public class Integer1
{
    public static void main(String[] args)
    {
        ArrayList<Integer> ar = new ArrayList<>();
        for (int i = 0; i < Integer.MAX_VALUE; i++)
        {
            ar.add(i);
        }
    }
}
```

[Run on IDE](#)

Output:

```
Exception in thread "main" java.lang.OutOfMemoryError: Java heap space
at java.util.Arrays.copyOf(Unknown Source)
at java.util.Arrays.copyOf(Unknown Source)
at java.util.ArrayList.grow(Unknown Source)
at java.util.ArrayList.ensureCapacityInternal(Unknown Source)
```

```
at java.util.ArrayList.add(Unknown Source)
at article.Integer1.main(Integer1.java:9)
```

This article is contributed by **Rishabh Mahrsee**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.



Java School Programming java-basics

[Login to Improve this Article](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Recommended Posts:

[For-each loop in Java](#)

[Decision Making in Java \(if, if-else, switch, break, continue, jump\)](#)

[Scope of Variables In Java](#)

[For Loop in Java | Important points](#)

[Variables in Java](#)

[Java Math copySign\(\) method with Examples](#)

[Java | Converting an Image into Grayscale using cvtColor\(\)](#)

[Java | Removing whitespaces using Regex](#)

[Java System.nanoTime\(\) vs System.currentTimeMillis](#)

[Java toDegrees\(\) method with Example](#)

(Login to Rate)

1.7

Average Difficulty : **1.7/5.0**
Based on **26** vote(s)

☐

Add to TODO List

☐

Mark as DONE

Basic

Easy

Medium

Hard

Expert

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

Share this post!

A computer science portal for geeks

710-B, Advant Navis Business Park,
Sector-142, Noida, Uttar Pradesh - 201305
feedback@geeksforgeeks.org

COMPANY

[About Us](#)
[Careers](#)
[Privacy Policy](#)
[Contact Us](#)

PRACTICE

[Company-wise](#)
[Topic-wise](#)
[Contests](#)
[Subjective Questions](#)

LEARN

[Algorithms](#)
[Data Structures](#)
[Languages](#)
[CS Subjects](#)
[Video Tutorials](#)

CONTRIBUTE

[Write an Article](#)
[GBlog](#)
[Videos](#)

@geeksforgeeks, Some rights reserved