

Control Structures - Loops

A loop causes a program section to be executed a number of times depending on a specified condition. The repetition continues till a specified condition is true and terminates as soon as the specified condition turns false. At the termination of the loop, the control of the execution is passed on to the statement following the loop.

There are three types of loops in Java

- **for**
- **while**
- **do...while**

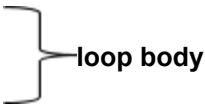
for Loop

The *for* loop is used to execute a program section a fixed number of times. The condition is always checked first before the loop is executed.

Syntax:

for(<initialization>; <condition>; <update expression>)

```
{
  statement 1;
  statement 2;
  statement 3;
  .
  .
}
```



Initialization: It is used to initialize the loop variable and is executed only once at the start of the loop. As shown in the example **1 (a)** - loop variable **a** is initialized to **1**.

Condition: It is evaluated every time before the execution of the statements in the loop body. The loop is executed if the condition evaluates to true, if evaluated to false the control of execution is passed to the statement immediately after the loop. As shown in the example **1 (a)** - the loop continues if the value of **a** ≤ **10** and stops the moment the value becomes **11**.

Update expression: This statement is executed after each execution of the statements in the loop body and therefore makes the loop variable attain the final value, after which the condition turns false and the loop execution stops.

In the following example **1(a)** the value of **a** is incremented after the execution of the loop body and keeps on getting incremented by one till it attains the value **11**, at which the condition turns false and the loop stops. Similarly, in the example in **1 (b)** the value of **a** is decremented by 1 each time and loop stops when the value of **a** becomes **0**, at which the condition turns false.

Examples:

1 (a)	1 (b)
for (int a=1;a<=10;a++) System.out.println(a + " ");	for (int a=10;a>=1;a--) System.out.println(a + " ");
Output	
1 2 3 4 5 6 7 8 9 10	10 9 8 7 6 5 4 3 2 1

A *for* loop can have initialization and increment/decrement of more than one variable but there can only be one test expression. Multiple statements in a loop needs to be enclosed in a pair of curly braces, on the other hand a single statement belonging to a loop does not require a pair of curly braces (As shown in examples 2 (a) and 2 (b)).

2 (a) <pre>for(int i=1,j=8,sum=0;i<=4;i++,j-=2) { System.out.println(i + " " + j); sum+=j; } System.out.println("sum is: " + sum);</pre>	2 (b) <pre>for(int i=1,j=2,k=5;i<=5;i++,j+=2,k+=5) System.out.println(i + ":" + j + ":" + k);</pre>
Output	
<pre>1 8 2 6 3 4 4 2 sum is: 20</pre>	<pre>1:2:5 2:4:10 3:6:15 4:8:20 5:10:25</pre>

Some more examples of *for* loop are as follows, which are not considered as good programming practices to be followed:

3 (a) loop without any statements <pre>for (int a=1;a<=10;a++); System.out.println(a); //Outside the loop /* The value of a is incremented from 1 to 11 and loop is terminated as the condition turns false*/</pre>	3 (b) loop without initialization and step value <pre>int l=1,sum=0; for(; l<=5;) { sum+=l; l++; } System.out.println(l + "\t" + sum);</pre>
Output	
11	6 15

while Loop:

The *while* loop executes a section of the program till a specified condition is true.

while loop is used when the number of times the loop is to be executed is not known.

The condition is always checked first before the loop is executed.

Syntax:

while(<condition>)

```
{
    statement 1;
    statement 2;
    .
    .
    .
    statement n;
}
```

} Loop Body

Examples:

1. while-loop with a single statement	2. The execution of the following loop depends on the value of Ans , therefore the number of times the loop is executed is not known.
<pre>int a=1; while(a<=4) System.out.println(a++); System.out.println("The final value of a:" + a);</pre>	<pre>char Ans; System.out.println("Want to find the square of a number (Y/N)?"); Ans=input.nextChar(); int N; while(Ans=='y') { System.out.println("Enter a no: "); N=input.nextInt(); System.out.println("Square=" + N*N); System.out.println("Repeat(Y/N):"); Ans=input.nextChar(); }</pre>
Output	
<pre>1 2 3 4 The final value of a: 5</pre>	

do...while Loop:

The **do..while** loop executes a section of the program till the specified condition is true.

do..while loop is used when the number of times the loop is to be executed is not known.

The body of the **do...while** loop is executed at least once, since the condition is placed at the end of the loop. Moreover, the statements written in this loop have to be always enclosed in a pair of curly braces, even if there is only a single statement in the loop.

Syntax:

```
do
{
    statement 1;
    statement 2;
    .
    .
    statement n;
} while(<condition>);
```

} Loop Body

Examples:

<pre>1. int a=1; do { System.out.println(a++); } while (a<=4);</pre>	<pre>2. char Ans; int N; do { System.out.println("Enter a no: "); N=input.nextInt(); System.out.println("Square=" + (N*N)); System.out.println("Repeat(Y/N):"); Ans=input.nextChar(); } while (Ans=='y');</pre>
<pre>Output 1 2 3 4</pre>	

Entry and Exit controlled loops

Entry Controlled Loops: These loops have a test expression/condition in the beginning of the loop, the test expression (condition) is therefore evaluated first, the loop is executed if the condition holds true. The control then moves back to test the condition for the next time and the process continues as long as the test expression evaluates to be true.

If the condition is false right in the beginning the loop will not be executed even once.

Examples: **for** and **while** loops.

Exit Controlled Loops: These loops have a test expression/condition at the end of the loop, thus having a control for exiting from the loop. The loop is executed at least once before the test expression is evaluated. If the condition holds true, the loop is executed again, otherwise, it exits the loop.

Example **do..while** .

In Entry controlled loops **for** and **while**, the body of the loop may not execute even once if the test expression evaluates to be false the first time, whereas in **do..while**, the loop is executed at least once whether the condition holds true the first time or not.

Example:

1. In the following example after the initialization of x with value 1, the control moves to test expression (x>=10), the result evaluates to be <u>false</u> , and hence the body of the loop will not execute even for once.	2. In the following example, first the body of the loop will execute with the value of x taken as 1, and then only the test expression (x>=10) shall be evaluated to be false, so that the control does not go back for the second iteration.
<pre>for (int x = 1; x>=10; x++) System.out.println(x);</pre>	<pre>int x = 1; do { System.out.println(x++); } while (x >=10);</pre>

Nested Loops:

When a **for**, **while** or **do..while** loop is used inside another looping construct, the concept is called a nested loop. This concept is used whenever for each repetition of a process many repetitions of another process are required.

Some sample skeletal structures of Nested Loops

<pre>for (..) { for(..) <statement> }</pre>	<pre>while() { while(..) <statement> }</pre>	<pre>do { while(..) <statement> } while (..);</pre>	<pre>do { for (..) <statement> } while (..);</pre>
---	---	---	--

Example:

```
for (int p = 1; p<=3; p++) //Outer Loop
{
    for (int q = 10; q <=20; q+=10) //Inner Loop
        System.out.print(q + " ");
    System.out.println(); //Statement belongs to outer loop
}
```

Working

p	q
1	10
	20
	30- loop stops
2	10
	20
	30- loop stops
3	10
	20
	30- loop stops
4-	outer loop stops

Output:

```
10 20
10 20
10 20
```

In the above example of nested loop, following steps are followed:

- Step 1: Value of variable p is initialized to 1, condition is checked and the outer loop is executed (i.e. Steps 2 and 3 are repeated) till the value of p becomes 4.
- Step 2: Control of execution moves to the inner loop, where the value of variable q is initialized to 10, condition is checked and the inner loop is executed till the value of q becomes 30, and the execution of inner loop stops.
- Step 3: Control now goes to the next statement [**System.out.println(endl;]**, which belongs to the outer loop.

This concept of nested loops can be used with the other two looping constructs also. i.e. Any type of loop construct can be nested inside any other type.