# UNIT-4 Looping Constructs

| |
|---|
| 4.1 Entry Controlled Loop: while loop, for loop.<br>Exit Controlled Loop: do-while. |
| 4.2 Nesting of Loops, Different patters using nested loop |
| 4.3 Break and Continue Statements |

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.

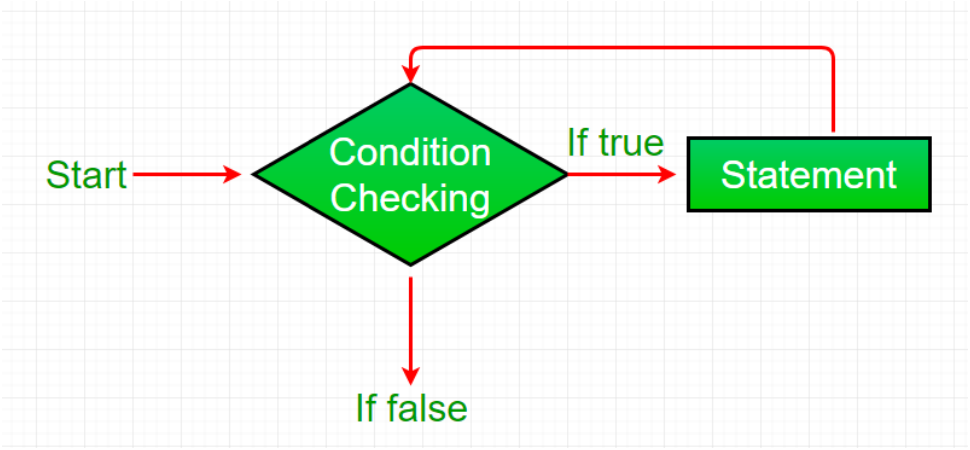
## Entry Controlled Loop vs Exit Controlled Loop

1. In an entry-controlled loop, the test condition is checked first followed by loop body, whereas in an exit-controlled loop, the loop body is executed first followed by the test condition
2. In an entry-controlled loop, if the test condition is false, the loop body will not be executed, whereas in exit-controlled loop, if the test condition is false, the loop body will be executed once.
3. Entry controlled loops are used when checking of test condition is mandatory before executing loop body, whereas exit controlled is used when checking of test condition is mandatory after executing.
4. For loop, Foreach loop and while loops are examples of entry-controlled loops, whereas do-while loop is an example of exit controlled loop.

**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...
}
```




**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.

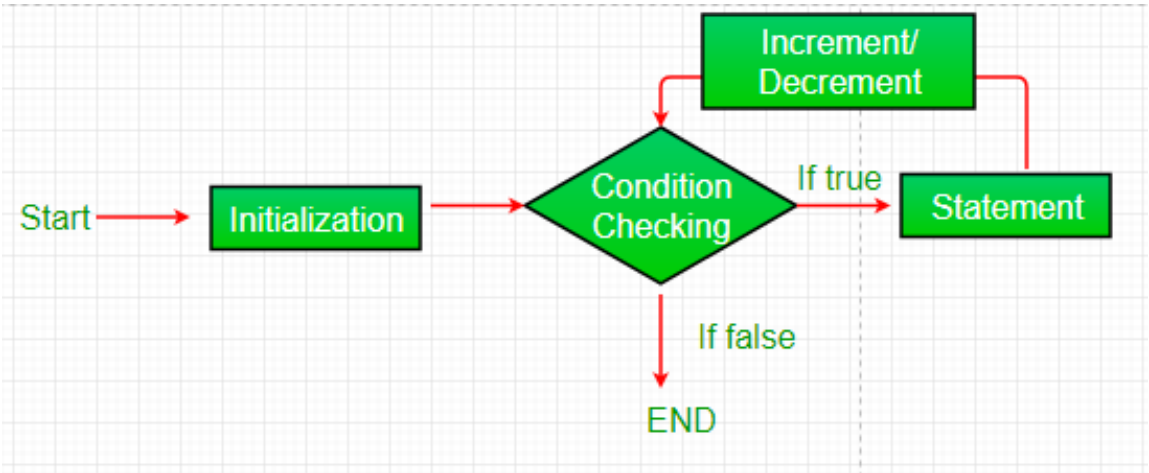
**Syntax:**

for (initialization condition; testing condition; increment/decrement)

{

statement(s)

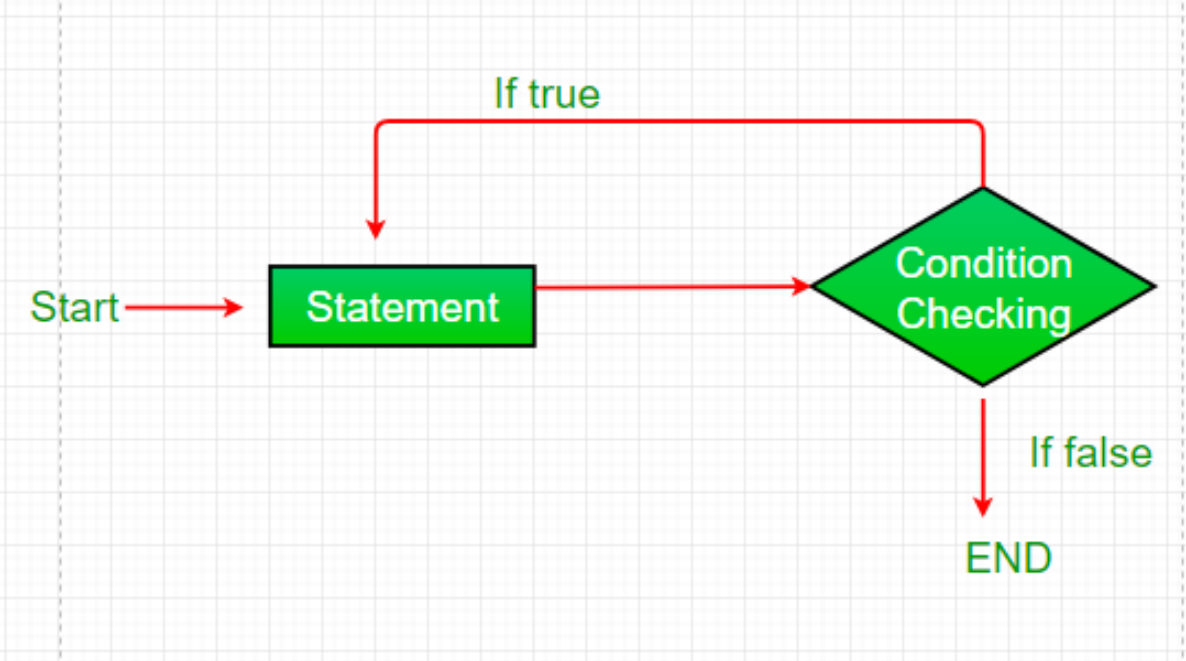
}



**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);
```




**Pitfalls of Loops**

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

```
//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");
        }
    }
}
```

## Nested Loops:-

Nested loop means a loop statement inside another loop statement. That is why nested loops are also called as “loop inside loop “.

**Syntax for Nested For loop:**

```
for ( initialization; condition; increment ) {
   for ( initialization; condition; increment ) {
      // statement of inside loop
   }
   // statement of outer loop
}
```

**Syntax for Nested While loop:**

```
while(condition) {
   while(condition) {
      // statement of inside loop
   }
   // statement of outer loop
}
```

**Syntax for Nested Do-While loop:**

```
do{
   do{
      // statement of inside loop
   }while(condition);
   // statement of outer loop
}while(condition);
```

## Example:-

```
class Main {
```

```
  public static void main(String[] args) {

    int weeks = 3;
    int days = 7;

    // outer loop prints weeks
    for (int i = 1; i <= weeks; ++i) {
      System.out.println("Week: " + i);

      // inner loop prints days
      for (int j = 1; j <= days; ++j) {
        System.out.println("  Day: " + j);
      }
    }
  }
}
```

**Output**

```
Week: 1
  Day: 1
  Day: 2
  Day: 3
    .....  ..  ....
Week: 2
  Day: 1
  Day: 2
  Day: 3
  ....  ..  ....
....  .. ....
```

## Java Break Statement: -

It is used to "jump out" of a switch statement.

The break statement can also be used to jump out of a loop.

This example stops the loop when i is equal to 4:

**Example**

```
for (int i = 0; i < 10; i++) {
  if (i == 4) {
    break;
  }
  System.out.println(i);
}
```

**Output: -**

0

1

2

3

## Java Continue Statement: -

The continue statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

This example skips the value of 4:

Example

```
for (int i = 0; i < 10; i++) {
  if (i == 4) {
    continue;
  }
  System.out.println(i);
```

}

**Output: -**

0

1

2

3

5

6

7

8

9