**Control Statements:**

**1. if:** if statement is the most simple decision-making statement. It is used to decide whether a certain statement or block of statements will be executed or not i.e if a certain condition is true then a block of statements is executed otherwise not.

**Syntax**:

if(condition)

{

// Statements to execute if

// condition is true

}

**2. if-else**: The if statement alone tells us that if a condition is true it will execute a block of statements and if the condition is false it won’t. But what if we want to do something else if the condition is false? Here comes the else statement. We can use the else statement with the if statement to execute a block of code when the condition is false.

**Syntax**:

if (condition)

{

// Executes this block if

// condition is true

}

else

{

// Executes this block if

// condition is false

}

**3. nested-if:** A nested if is an if statement that is the target of another if or else. Nested if statements mean an if statement inside an if statement. Yes, java allows us to nest if statements within if statements. i.e, we can place an if statement inside another if statement.

**Syntax:**

if (condition1)

{

// Executes when condition1 is true

if (condition2)

{

// Executes when condition2 is true

}

}

**4. if-else-if ladder:** Here, a user can decide among multiple options.The if statements are executed from the top down. As soon as one of the conditions controlling the if is true, the statement associated with that ‘if’ is executed, and the rest of the ladder is bypassed. If none of the conditions is true, then the final else statement will be executed. There can be as many as ‘else if’ blocks associated with one ‘if’ block but only one ‘else’ block is allowed with one ‘if’ block.

if (condition)

statement;

else if (condition)

statement;

.

.

else

statement;

**5. switch-case:** The switch statement is a multiway branch statement. It provides an easy way to dispatch execution to different parts of code based on the value of the expression.

**Syntax:**

switch (expression)

{

case value1:

statement1;

break;

case value2:

statement2;

break;

.

.

case valueN:

statementN;

break;

default:

statementDefault;

}

**6. jump:** Java supports three jump statements: **break, continue** and **return**. These three statements transfer control to another part of the program.

* **Break:** In Java, a break is majorly used for:
  + Terminate a sequence in a switch statement (discussed above).
  + To exit a loop.
  + Used as a “civilized” form of goto.
* **Continue:** Sometimes it is useful to force an early iteration of a loop. That is, you might want to continue running the loop but stop processing the remainder of the code in its body for this particular iteration. This is, in effect, a goto just past the body of the loop, to the loop’s end. The continue statement performs such an action.
* **Return:** The return statement is used to explicitly return from a method. That is, it causes program control to transfer back to the caller of the method.

**Loops:**

**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.**

**java provides Three types of Conditional statements this second type is loop statement .**

* **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 :**

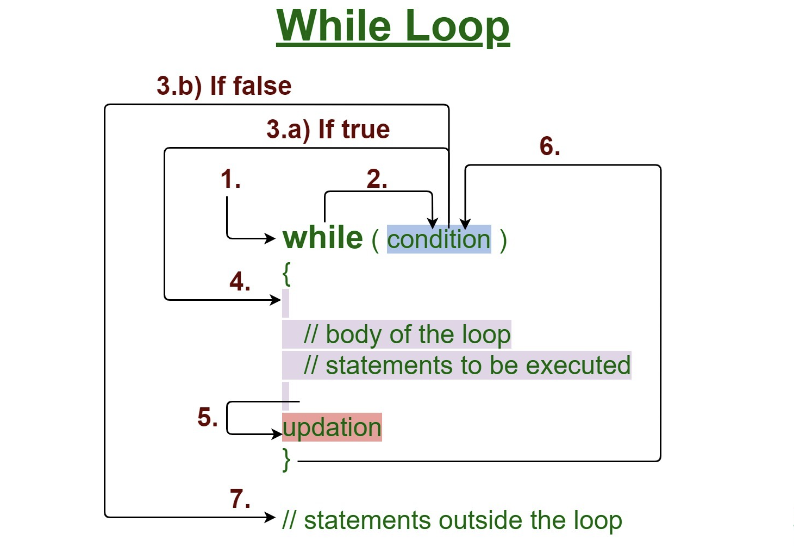
**while (boolean condition)**

**{**

**loop statements...**

**}**

* + **While loop starts with the checking of Boolean condition. If it is evaluated to true, then the loop body statements are executed, otherwise the 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.**

****

* **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.**

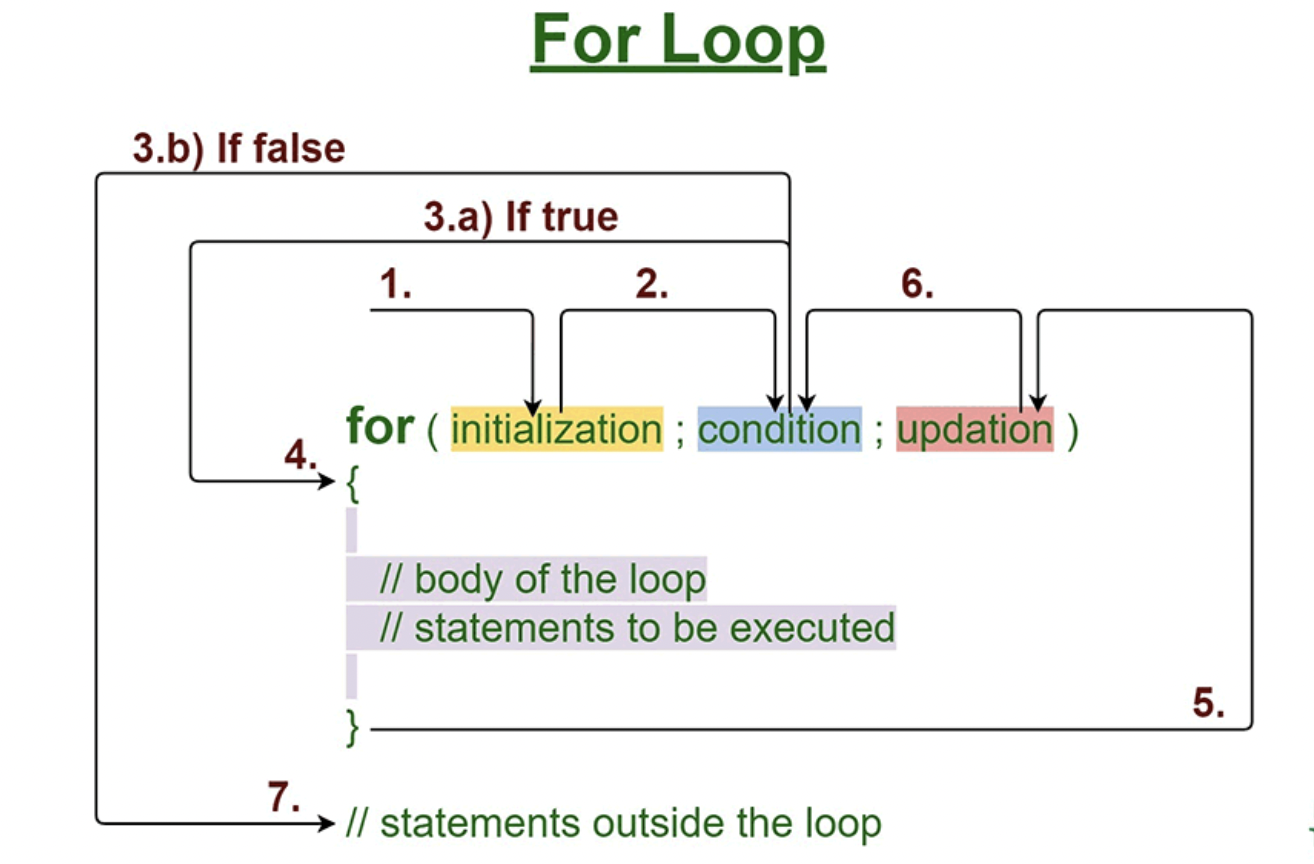
**Syntax:**

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

**{**

**statement(s)**

**}**

****

* + **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.**
  + **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.**
  + **Statement execution: Once the condition is evaluated to true, the statements in the loop body are executed.**
  + **Increment/ Decrement: It is used for updating the variable for the next iteration.**
  + **Loop termination:When the condition becomes false, the loop terminates marking the end of its life cycle.**
* **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:**

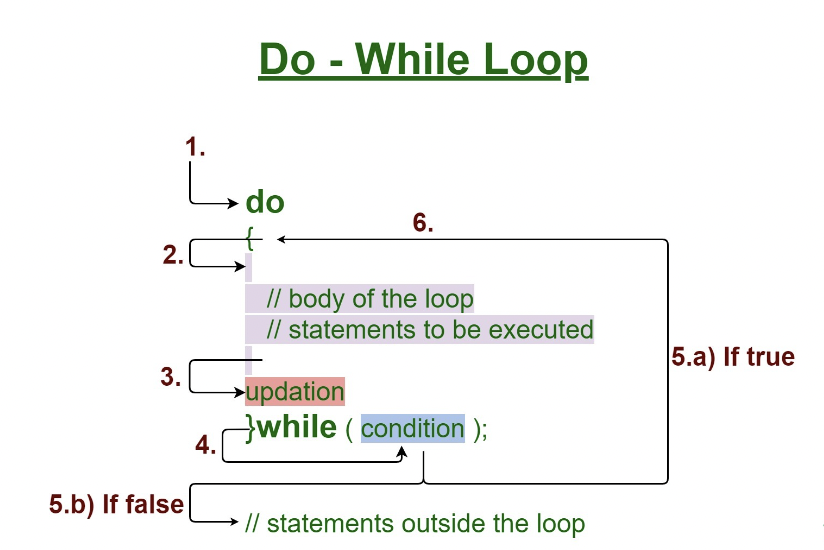
**do**

**{**

**statements..**

**}**

**while (condition);**

****

| * + **do while loop starts with the execution of the statement(s). There is no checking of any condition for the first time.**   + **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 be true, the next iteration of the loop starts.**   + **When the condition becomes false, the loop terminates which marks the end of its life cycle.**   + **It is important to note that the do-while loop will execute its statements at least once before any condition is checked, and therefore is an example of an exit control loop.**   **Pitfalls of Loops**   * **Infinite loop: One of the most common mistakes while implementing any sort of loop is that it may not ever exit, that is the loop runs for infinite time. This happens when the condition fails for some reason.** |
| --- |