Using Looping Constructs

Learning Outcomes:

After successful completion of this lesson, you should be able to:

- Apply loop constructs to indicate repetitive tasks.

Event Controlled Loop Construct

In programming languages, loops are used to execute a set of instructions/functions repeatedly when some conditions become true. There are three types of loops in Java.

- for loop
- while loop
- do-while loop

Advantage with looping statement

- Reduce length of Code
- Take less memory space.
- Burden on the developer is reducing.
- Time consuming process to execute the program is reduced.

Difference between conditional and looping statement

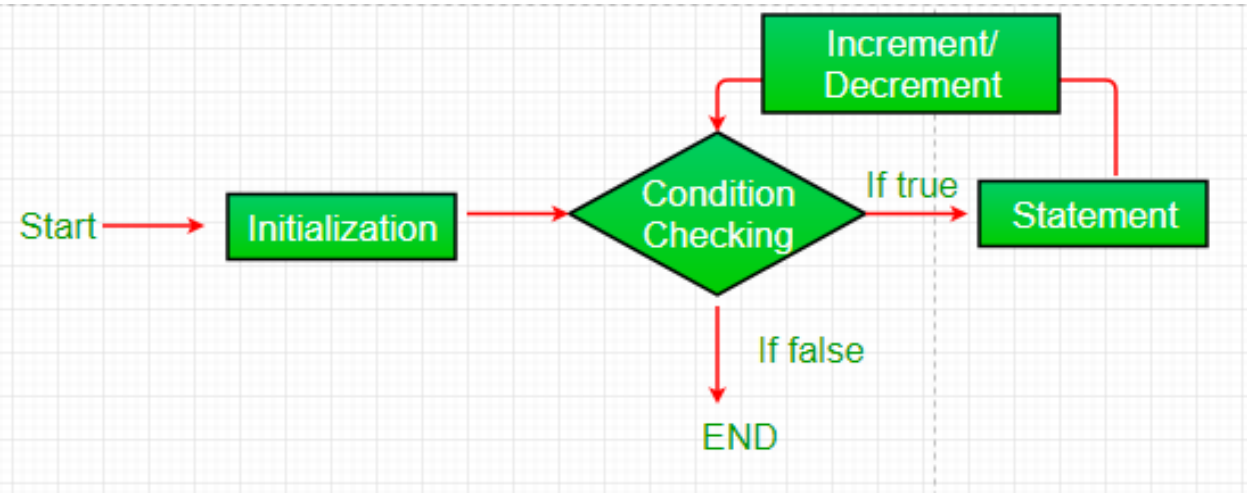
Conditional statement executes only once in the program where as looping statements executes repeatedly several number of time.

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.

Example:

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

Output:

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

Example:

```
for (int i = 0; i < 5; i++) {

  System.out.println(i);

}
```

Example explained

- Statement 1 sets a variable before the loop starts (int i = 0).
- Statement 2 defines the condition for the loop to run (i must be less than 5). If the condition is true, the loop will start over again, if it is false, the loop will end.
- Statement 3 increases a value (i++) each time the code block in the loop has been executed.

Another Example:

This example will only print even values between 0 and 10:

```
for (int i = 0; i <= 10; i = i + 2) {
  System.out.println(i);
}
```

Example:

```
class ForLoopExample {
   public static void main(String args[]){
       for(int i=10; i>1; i--){
           System.out.println("The value of i is: "+i);
       }
   }
}
```

The output of this program is:

The value of i is: 10
The value of i is: 9
The value of i is: 8
The value of i is: 7
The value of i is: 6
The value of i is: 5
The value of i is: 4
The value of i is: 3
The value of i is: 2

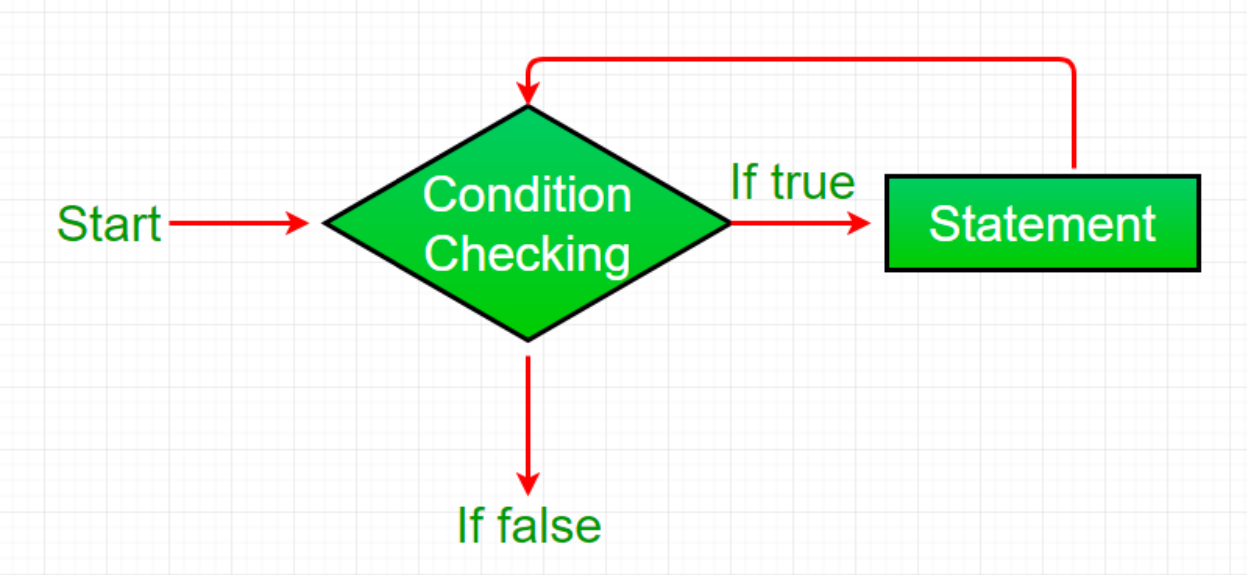
While Loop

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++;
}
}
}
```

Output:

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

Example:

```
int i = 0;
while (i < 5) {
  System.out.println(i);
  i++;
}
```

Note: Do not forget to increase the variable used in the condition, otherwise the loop will never end!

Example:

```
class WhileLoopExample {
    public static void main(String args[]){
        int i=10;
        while(i>1){
            System.out.println(i);
            i--;
        }
    }
}
```

Output:

```
10
9
8
7
6
5
4
3
2
```

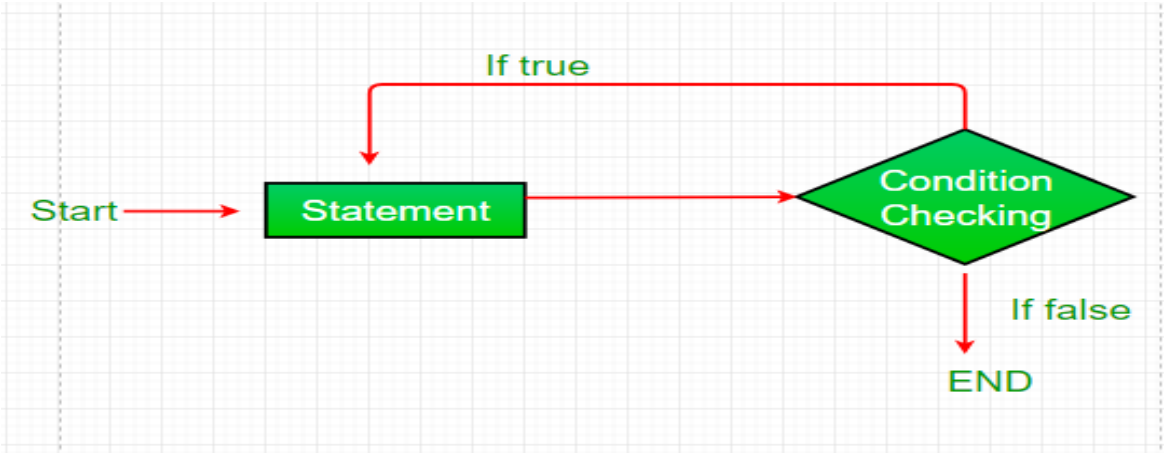
do while

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.

Example:

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

Output:

Value of x: 21

Example:

```
int i = 0;
do {
  System.out.println(i);
  i++;
}
while (i < 5);
```

Do not forget to increase the variable used in the condition, otherwise the loop will never end!

```
class DoWhileLoopExample {
    public static void main(String args[]){
        int i=10;
        do{
            System.out.println(i);
            i--;
        }while(i>1);
    }
}
```

Output:

```
10
9
8
7
6
5
4
3
2
```

Count Controlled Loop Construct

Count-controlled repetition requires

- control variable (or loop counter)
- initial value of the control variable
- increment (or decrement) by which the control variable is modified each iteration through the loop
- condition that tests for the final value of the control variable

A count-controlled repetition will exit after running a certain number of times. The count is kept in a variable called an index or counter. When the index reaches a certain value (the loop bound) the loop will end.

Count-controlled repetition is often called definite repetition because the number of repetitions is known before the loop begins executing. When we do not know in advance the number of times we want to execute a statement, we cannot use count-controlled repetition. In such an instance, we would use sentinel-controlled repetition.

Display the character and its' ASCII value for all lower case characters.

```
for (char x = 'a'; x <= 'z'; x++)
   System.out.println(x + " = " + (int) x);
```