Java Loops

Loop is an important concept of a programming that allows to iterate over the sequence of statements.

Loop is designed to execute particular code block till the specified condition is true or all the elements of a collection (array, list etc) are completely traversed.

The most common use of loop is to perform repetitive tasks. For example, if we want to print table of a number then we need to write print statement 10 times. However, we can do the same with a single print statement by using loop.

Loop is designed to execute its block till the specified condition is true.

Java provides mainly three loops based on the loop structure.

1. for loop

2. while loop

3. do while loop

We will see each loop individually in details in the below.

1. for loop

The for loop is used for executing a part of the program repeatedly. When the number of executions is fixed then it is suggested to use for loop. For loop can be categories into two types.

for loop

for-each loop

- for loop Syntax:

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

EQUALITY EDUCATION

- for loop Parameters:

To create a for loop, we need to set the following parameters.

1) Initialization

It is the initial part, where we set initial value for the loop. It is executed only once at the starting of loop. It is optional, if we don't want to set initial value.

2) Condition

It is used to test a condition each time while executing. The execution continues until the condition is false. It is optional and if we don't specify, loop will be infinite.

3) Statement

It is in loop body and executed every time until the condition is false.

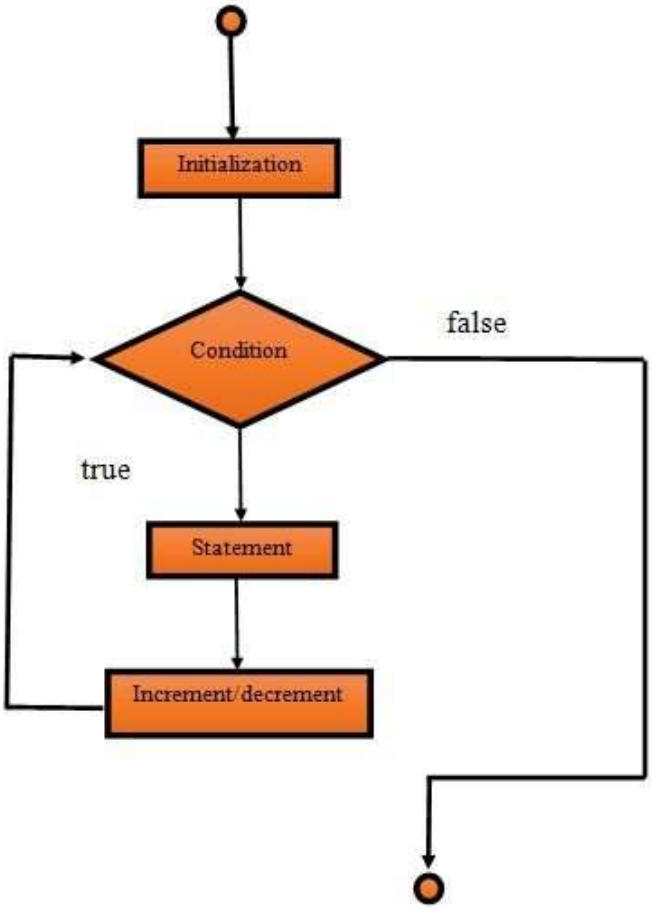
4) Increment/Decrement

It is used for set increment or decrement value for the loop.

- Data Flow Diagram of for loop

This flow diagram shows flow of the loop. Here we can understand flow of the loop.

EQUALITY EDUCATION



- Example

In this example, initial value of loop is set to 1 and incrementing it by 1 till the condition is true and executes 10 times.

```
class ForDemo1
{
	public static void main(String[] args)
	{
		int n, i;
		n=2;
		for(i=1;i<=10;i++)
		{
			System.out.println(n+"*"+i+"="+n*i);
		}
	}
```

EQUALITY EDUCATION

```
}
```

- Example for Nested for loop

Loop can be nested, loop created inside another loop is called nested loop. Sometimes based on the requirement, we have to create nested loop.

Generally, nested loops are used to iterate tabular data.

```
class ForDemo2 {
        public static void main(String[] args) {
                for(inti=1;i<=5;i++) {
                        for(int j=1;j<=i;j++)
                        {
                                System.out.print("* ");
                        }
                System.out.println();
                }
        }
}
```

for-each Loop

In Java, for each loop is used for traversing array or collection elements. In this loop, there is no need for increment or decrement operator.

- Syntax:

```
for(Type var:array)
{
    //code for execution
}
```

- Example:

In this example, we are traversing array elements using the for-each loop. For-each loop terminates automatically when no element is left in the array object.

```
class ForEachDemo1 {
        public static void main(String[] args) {
```

EQUALITY EDUCATION

```
        int a[]={20,21,22,23,24};
        for(int i:a)
        {
            System.out.println(i);
        }
    }
}
```

2. While Loop

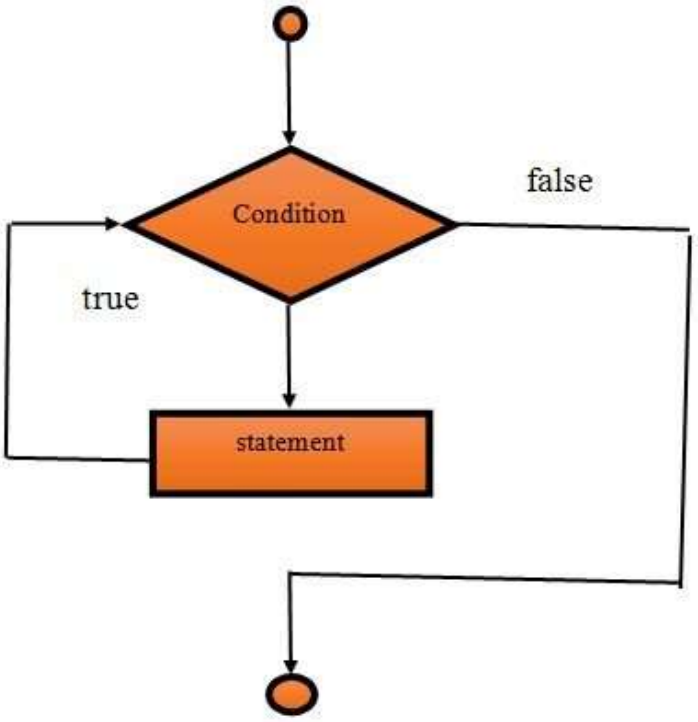
Like for loop, while loop is also used to execute code repeatedly. It is used for iterating a part of the program several times. When the number of iterations is not fixed then while loop is used.

- Syntax:

```
while(condition)
{
    //code for execution
}
```

- Data-flow-diagram of While Block

EQUALITY EDUCATION



- Example:

In this example, we are using while loop to print 1 to 10 values. In first step, we set conditional variable then test the condition and if condition is true execute the loop body and increment the variable by 1.

```
class WhileDemo1
{
	public static void main(String[] args)
	{
		Int i=1;
		while(i<=10)
		{
			System.out.println(i);
			i++;
		}
	}
}
```

EQUALITY EDUCATION

- Example for infinite while loop

A while loop which conditional expression always returns true is called infinite while loop. We can also create infinite loop by passing true literal in the loop.

Be careful, when creating infinite loop because it can issue memory overflow problem.

```
class WhileDemo2
{
	public static void main(String[] args)
	{
		while(true)
		{
			System.out.println("infinitive while loop");
		}
	}
}
```

3. do-while loop

In Java, the do-while loop is used to execute statements again and again. This loop executes at least once because the loop is executed before the condition is checked. It means loop condition evaluates after executing of loop body.

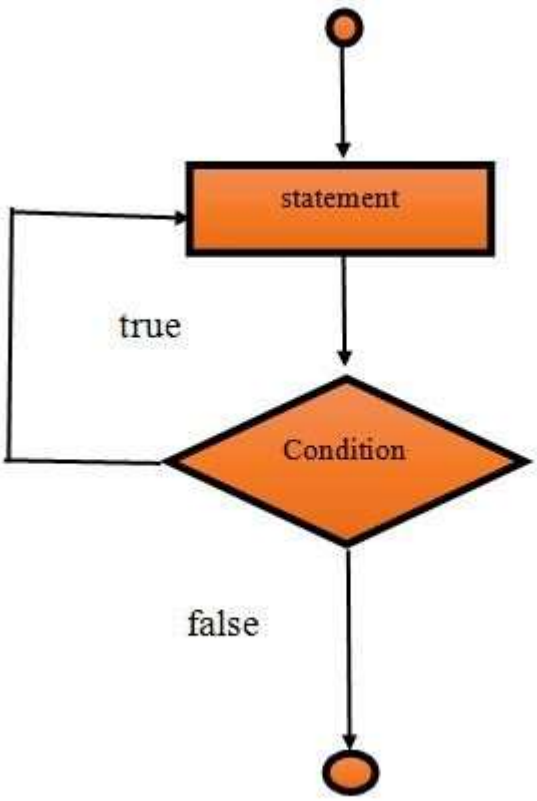
The main difference between while and do-while loop is, in do while loop condition evaluates after executing the loop.

Syntax:

```
do
{
	//code for execution
}
while(condition);
```

EQUALITY EDUCATION

- Data Flow Diagram of do-while Block



- Example:

In this example, we are printing values from 1 to 10 by using the do while loop.

```
class DoWhileDemo1
{
    public static void main(String[] args)
    {
        Int i=1;
        do
        {
            System.out.println(i);
            i++;
```

EQUALITY EDUCATION

```
        }while(i<=10);
    }
}
```

- Example for infinite do-while loop

Like infinite while loop, we can create infinite do while loop as well. To create an infinite do while loop just pass the condition that always remains true.

```
class DoWhileDemo2
{
    public static void main(String[] args)
    {
        do
        {
            System.out.println("infinitive do while loop");
        }while(true);
    }
}
```

EQUALITY EDUCATION