Java Logical Operators with Examples

Difficulty Level : Easy Last Updated : 25 Nov, 2019

Operators constitute the basic building block to any programming language. Java too provides many types of operators which can be used according to the need to perform various calculation and functions be it logical, arithmetic, relational etc. They are classified based on the functionality they provide. Here are a few types:

1. Arithmetic Operators
2. Unary Operators
3. Assignment Operator
4. Relational Operators
5. Logical Operators
6. Ternary Operator
7. Bitwise Operators
8. Shift Operators

This article explains all that one needs to know regarding the Logical Operators.

Logical Operators

These operators are used to perform logical “AND”, “OR” and “NOT” operation, i.e. the function similar to AND gate and OR gate in digital electronics. They are used to combine two or more conditions/constraints or to complement the evaluation of the original condition under particular consideration. One thing to keep in mind is the second condition is not evaluated if the first one is false, i.e. it has a short-circuiting effect. Used extensively to test for several conditions for making a decision. Let’s look at each of the logical operators in a detailed manner:

1. **‘Logical AND’ Operator(&&):** This operator returns true when both the conditions under consideration are satisfied or are true. If even one of the two yields false, the operator results false. For example, cond1 && cond2 returns true when both cond1 and cond2 are true (i.e. non-zero).
   Syntax:

   ```
   condition1 && condition2
   ```

   Example:

```
a = 10, b = 20, c = 20

condition1: a < b
condition2: b == c

if(condition1 && condition2)
d = a+b+c

// Since both the conditions are true
d = 50.
```

```
// Java code to illustrate
// logical AND operator

import java.io.*;

class Logical {
    public static void main(String[] args)
    {
        // initializing variables
        int a = 10, b = 20, c = 20, d = 0;

        // Displaying a, b, c
        System.out.println("Var1 = " + a);
        System.out.println("Var2 = " + b);
        System.out.println("Var3 = " + c);

        // using logical AND to verify
        // two constraints
        if ((a < b) && (b == c)) {
            d = a + b + c;
            System.out.println("The sum is: " + d);
        }
        else
            System.out.println("False conditions");
    }
}
```

Output:

```
Var1 = 10
Var2 = 20
```

```
Var3 = 20
The sum is: 50
```

2. **'Logical OR' Operator(||):** This operator returns true when one of the two conditions under consideration are satisfied or are true. If even one of the two yields true, the operator results true. To make the result false, both the constraints need to return false.
   Syntax:

```
condition1 || condition2
```

   Example:

```
a = 10, b = 20, c = 20

condition1: a < b
condition2: b > c

if(condition1 || condition2)
d = a+b+c

// Since one of the condition is true
d = 50.
```

```
// Java code to illustrate
// logical OR operator

import java.io.*;

class Logical {
    public static void main(String[] args)
    {
        // initializing variables
        int a = 10, b = 1, c = 10, d = 30;

        // Displaying a, b, c
        System.out.println("Var1 = " + a);
        System.out.println("Var2 = " + b);
        System.out.println("Var3 = " + c);
        System.out.println("Var4 = " + d);

        // using logical OR to verify
        // two constraints
        if (a > b || c == d)
            System.out.println("One or both"
                               + " the conditions are true");
```

```
        else
            System.out.println("Both the"
                               + " conditions are false");
    }
}
```

Output:

```
Var1 = 10
Var2 = 1
Var3 = 10
Var4 = 30
One or both the conditions are true
```

3. **'Logical NOT' Operator(!):** Unlike the previous two, this is a unary operator and returns true when the condition under consideration is not satisfied or is a false condition. Basically, if the condition is false, the operation returns true and when the condition is true, the operation returns false.
   Syntax:

```
!(condition)
```

Example:

```
a = 10, b = 20

!(a<b) // returns false
!(a>b) // returns true
```

```
// Java code to illustrate
// logical NOT operator
import java.io.*;

class Logical {
    public static void main(String[] args)
    {
        // initializing variables
        int a = 10, b = 1;

        // Displaying a, b, c
        System.out.println("Var1 = " + a);
```

```
        System.out.println("Var2 = " + b);

        // Using logical NOT operator
        System.out.println("!(a < b) = " + !(a < b));
        System.out.println("!(a > b) = " + !(a > b));
    }
}
```

Output:

```
Var1 = 10
Var2 = 1
!(a < b) = true
!(a > b) = false
```