JAVA CONDITIONAL OR RELATIONAL OPERATORS

Description

If you need to change the execution of the program based on a certain condition you can use “if” statements. The relational operators determine the relationship that one operand has to the other. Specifically, they determine equality condition. Java provides six relational operators, which are listed in below table.

Operator	Description	Example (a=10, b=15)	Result
==	Equality operator	a==b	false
!=	Not Equal to operator	a!=b	true
>	Greater than	a>b	false
<	Less than	a<b	true
>=	Greater than or equal to	a>=b	false
<=	Less than or equal to	a<=b	true

The outcome of these operations is a boolean value. The relational operators are most frequently used in the expressions that control the if statement and the various loop statements. Any type in Java, including integers, floating-point numbers,

characters, and booleans can be compared using the equality test, ==, and the inequality test, !=. Notice that in Java equality is denoted with two equal signs (“==”), not one (“=”).

In Java, the simplest statement you can use to make a decision is the if statement. Its simplest form is shown here:

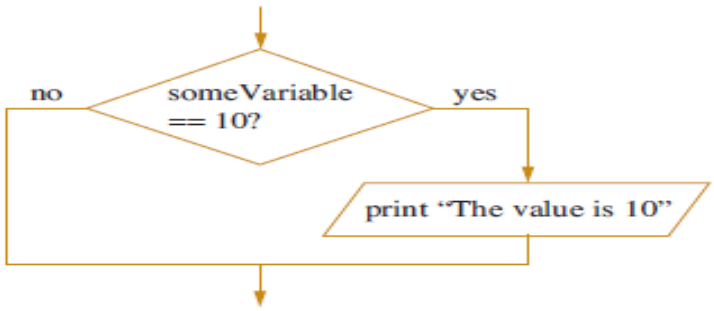
```
if(condition) statement;

  or

if (condition) statement1;

else statement2;
```

Here, condition is a Boolean expression. If condition is true, then the statement is executed. If condition is false, then the statement is bypassed. For example, suppose you have declared an integer variable named someVariable, and you want to print a message when the value of someVariable is 10. Flow chart and java code of the operation looks like below,



```
1.  if (someVariable ==10){
2.
3.  System.out.println("The Value is 10”);
4.
5.  }
```

We can have different flavors of if statements.

Nested if blocks:

A nested if is an if statement that is the target of another if or else. In other terms we can consider one or multiple if statement within one if block to check various condition. For example we have two variables and want to check particular condition for both we can use nested if blocks.

Java Code

```
1.  package logicaloperator;
2.
3.
4.
5.  public class NestedIfDemo {
6.
7.
8.
9.      public static void main(String[] args) {
10.
```

```
11.         int a =10;
12.
13.         int b =5;
14.
15.         if(a==10){
16.
17.             if(b==5){
18.
19.                 System.out.println("Inside Nested Loop");
20.
21.             }
22.
23.         }
24.
25.     }
26.
27. }
```

if –else if ladder:

We might get situation where we need to check value multiple times to find exact matching condition. Below program explain the same thing. Let’s see we have requirement to check if variable value is less than 100, equal to 100 or more than 100. Below code explain the same logic using if-else if ladder.

Short-Circuit Logical Operators

Java provides two interesting Boolean operators not found in many other computer languages. These are secondary versions of the Boolean AND and OR operators, and are known as short-circuit logical operators. Two short-circuit logical operators are as follows,

- && short-circuit AND
- || short-circuit OR

They are used to link little boolean expressions together to form bigger boolean expressions. The && and || operators evaluate only boolean values. For an AND (&&) expression to be true, both operands must be true. For example, The below statement evaluates to true because both operand one (2 < 3) and operand two (3 < 4) evaluate to true.

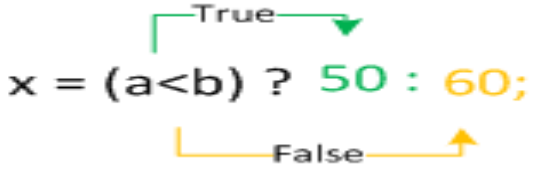
```
if ((2 < 3) && (3 < 4)) { }
```

The short-circuit feature of the && operator is so named because it doesn't waste its time on pointless evaluations. A short-circuit && evaluates the left side of the operation first (operand one), and if it resolves to false, the && operator doesn't bother looking at the right side of the expression (operand two) since the && operator already knows that the complete expression can't possibly be true.

The || operator is similar to the && operator, except that it evaluates to true if EITHER of the operands is true. If the first operand in an OR operation is true, the result will be true, so the short-circuit || doesn't waste time looking at the right side of the equation. If the first operand is false, however, the short-circuit || has to evaluate the second operand to see if the result of the OR operation will be true or false.

Ternary Operator (or ? Operator or Conditional Operator)

The conditional operator is a ternary operator (it has three operands) and is used to evaluate boolean expressions, much like an if statement except instead of executing a block of code if the test is true, a conditional operator will assign a value to a variable. A conditional operator starts with a boolean operation, followed by two possible values for the variable to the left of the assignment (=) operator. The first value (the one to the left of the colon) is assigned if the conditional (boolean) test is true, and the second value is assigned if the conditional test is false. In below example if variable a is less then b then variable x value would be 50 else x =60.



Summary

- Java provides six conditional operators == (equality), > (greater than), < (less than), >=(greater or equal), <= (less or equal), != (not equal)
- The relational operators are most frequently used to control the flow of program.
- Short-Circuit logical operators are && and ||
- Ternary operator is one which is similar to if else block but which is used to assign value based on condition.

Source: http://www.w3resource.com/java-tutorial/java-conditional-operators.php