Boolean Operators

by Sophia

WHAT'S COVERED

In this lesson, you will learn about how to use multiple Boolean operators to handle various conditions in Java. Specifically, this lesson covers:

Table of Contents

1. Single Argument Boolean Operators

The Java boolean type has two possible values: true or false. In addition to the boolean type, Java provides three boolean operators, also called logical operators. These include && (logical "and"), || (logical "or"), and ! (logical "not"). **Boolean operators** provide the ability to build and combine more complex boolean expressions. Boolean operators take boolean inputs and return boolean results.

Since the boolean values can only have one of two values, true or false, the operators are specified and assigned in a **truth table**. A truth table is a table that lists the possible input values and shows the logical results of applying the boolean operators.

Here is the simplest truth table (A is a boolean type and can be either true or false):

A
true
false

Remember that boolean operators allow programmers to build more complex boolean expressions from basic expressions. This is where singular vs. multiple arguments come into play. Of the three boolean operators that have been mentioned, only the **not boolean operator**, !, works with a singular argument.

1a. not Boolean Operator

The only boolean operator that works with one argument is not. In Java, the not operator is represented as an exclamation mark (!). The **! boolean operator** returns the opposite truth value of the argument. This means that if false is passed to the operator, then true is returned. If true is passed to the operator, then false is returned.

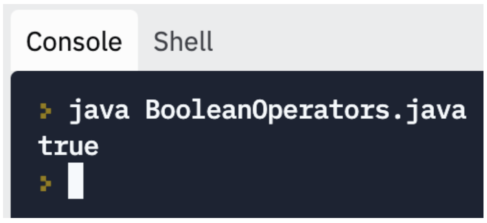
Here is the what the not operator in a truth table looks like:

A	! A
true	false
false	true

Here is a simple example to compare the values when they are equal, which should return true:

```
class BooleanOperators {
  public static void main(String[] args) {
    int number = 1;
    // == will return true or false
    System.out.println(number == 1);
  }
}
```

When running in Replit, the output should be as follows:



```
> java BooleanOperators.java
true
>
```

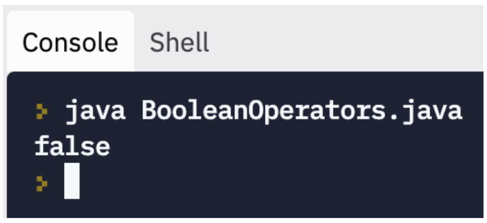
THINK ABOUT IT

As expected, num does equal 1, so it is true.

Here is a comparison of the values when they are not equal. When run, this should return false:

```
class BooleanOperators {
  public static void main(String[] args) {
    int number = 1;
    // == will return true or false
    System.out.println(number == 0);
  }
}
```

The output should look like this:



```
> java BooleanOperators.java
false
>
```

THINK ABOUT IT

As expected, num does not equal 0, so it is false.

Now, let’s try the same example, but this time by adding in the booleannot (!) operator so the code ends up with the != (not equal) operator.

```
class BooleanOperators {
  public static void main(String[] args) {
    int number = 1;
    // == will return true or false
    System.out.println(number != 0);
  }
}
```

TERMS TO KNOW

Boolean Operators
Boolean operators (also known as logical operators) allow us to build and combine more complex boolean expressions from more basic expressions.

Truth Table
A truth table is a small table that lists every possible input value and gives results for the operators.

not Boolean Operator
The not boolean operator (in Java represented by !) works with one argument and returns the opposite truth value of the argument.

2. Multiple Argument Boolean Operators

The next two boolean operator types include and and or. These two operators take multiple arguments.

2a. Boolean && (and) Operator

The **boolean && (and) operator** takes two arguments and returns false unless both inputs are true.

Let’s see what this looks like in a truth table

A	B	A && B
false	false	false
false	true	false
true	false	false

true	true	true

Notice in this table that if A is false, then no matter what the value is in B, the output is still false.

2b. Short-Circuit Evaluation

CONCEPT TO KNOW

Java boolean operators such as && and || use an evaluation method called short-circuit evaluation. Java reads code from left to right, just like we read. Java starts evaluating the left operand and if it detects that there is nothing to be gained by evaluating the rest of a logical expression, it stops its evaluation and does not do the computations in the rest of the logical expression. This is called short-circuit evaluation, when the evaluation of a logical expression stops because the overall value is already known.

For example, with the && boolean operator and as illustrated in the truth table above, if the first operand evaluates as false, Java can stop evaluating the rest of the statement since the overall value has to be false.

Looking back at the table above, the operator short-circuits if the first input is false. That means that if the first input is false, the second input isn't even evaluated.

Double check this through code to ensure that the conditions match the truth table above:

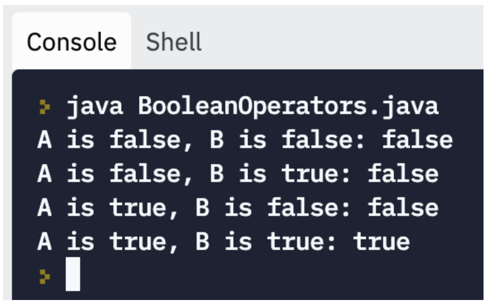
```
class BooleanOperators {
  public static void main(String[] args) {
    int num1 = 1;
    int num2 = 2;

    if(num1 == 0 && num2 == 0) {
      System.out.println("A is false, B is false: true");
    }
    else {
      System.out.println("A is false, B is false: false");
    }

    if(num1 == 0 && num2 == 2) {
      System.out.println("A is false, B is true: true");
    }
    else {
      System.out.println("A is false, B is true: false");
    }

    if(num1 == 1 && num2 == 0) {
      System.out.println("A is true, B is false: true");
    }
    else {
      System.out.println("A is true, B is false: false");
    }
    if(num1 == 1 && num2 == 2) {
      System.out.println("A is true, B is true: true");
    }
    else {
      System.out.println("A is true, b is true: false");
    }
  }
}
```

The output should look like this:



```
> java BooleanOperators.java
A is false, B is false: false
A is false, B is true: false
A is true, B is false: false
A is true, B is true: true
>
```

As you'll see, with the && operator, it only outputs true if and only if both inputs result in true.

2c. or Boolean Operator

The || (**boolean or operator**) outputs true if either or both of the inputs are true. It only outputs false if both inputs are set to false.

Consider the truth table below for the || boolean operator.

A	B	A ǁ B
false	false	false
false	true	true
true	false	true
true	true	true

It's important to note that the English word "or" can have two separate meanings when used informally.

- Exclusive or: there is the "exclusive or" where you can only have one of two conditions. For example, if we say, "You can stay home or you can go to the grocery store," this means that you cannot do both at the same time; you have to choose one of the two.
- Inclusive or: the "inclusive or" is sometimes used with and/or as part of a phrase. For example, we can say, "You can have ice cream and/or a cookie for dessert." This means that you could either eat ice cream, a cookie, or even both. In Java and other programming languages, the ǁ operator is treated as the "inclusive or." This means that if either inputs are true or if both inputs are true, the result is true.

In this case, the || operator also uses short-circuit evaluation. If the first input is true, the result is true and there's no need to evaluate the second argument.
Let's see what this looks like in code that evaluates each possibility:

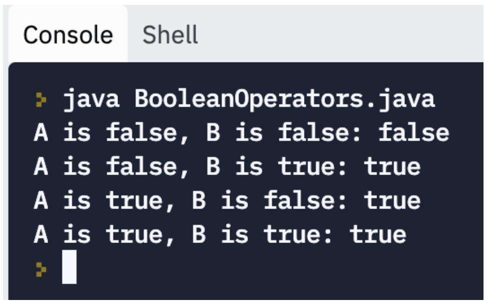
```
class BooleanOperators {
  public static void main(String[] args) {
    int num1 = 1;
    int num2 = 2;

    if(num1 == 0 || num2 == 0) {
      System.out.println("A is false, B is false: true");
    }
    else {
      System.out.println("A is false, B is false: false");
    }

    if(num1 == 0 || num2 == 2) {
      System.out.println("A is false, B is true: true");
    }
    else {
      System.out.println("A is false, B is true: false");
    }

    if(num1 == 1 || num2 == 0) {
      System.out.println("A is true, B is false: true");
    }
    else {
      System.out.println("A is true, B is false: false");
    }

    if(num1 == 1 || num2 == 2) {
      System.out.println("A is true, B is true: true");
    }
    else {
      System.out.println("A is true, b is true: false");
    }
  }
}
```
The output should look look like this:



```
> java BooleanOperators.java
A is false, B is false: false
A is false, B is true: true
A is true, B is false: true
A is true, B is true: true
>
```

TRY IT

Directions: Using Replit, try some of the boolean operators to see if you can change the output based on what you have just learned.

REFLECT

Boolean may be a new term to you. It is derived from the name of George Bool who created the theory behind this concept. The next time you hear a preschooler saying 1 and 1 is 2 you might now think – not always. If you're coding and doing logical operations, 1 and 1 is 1 which is true in this context.

TERMS TO KNOW

&& (and) Boolean Operator
The boolean && operator takes two arguments and returns false unless both inputs are true.

Short-Circuit Evaluation
Short-circuit evaluation is when the evaluation of a logical expression stops because the overall value is already known.

|| **(or) Boolean Operator**
The boolean || operator outputs true if either input is true (or both).

SUMMARY

In this lesson, you learned that Java provides three boolean operators, also called logical operators. They include **boolean && (and),** || **(or), and ! (not)**. You discovered that these operators allow you to build and combine more complex boolean expressions. You explored how a **! boolean operator** works with a **singular argument** to return a true or false value. You also considered **multiple argument** situations, and how the **&& and** || **boolean operators** are used. Finally, you learned that Java conducts **short-circuit evaluation** on and and or operations. This means that Java will stop evaluating logical expressions if the overall value is already known.

Source: This content and supplemental material has been adapted from Java, Java, Java: Object-Oriented Problem Solving. Source cs.trincoll.edu/~ram/jjj/jjj-os-20170625.pdf

It has also been adapted from "Python for Everybody" By Dr. Charles R. Severance. Source py4e.com/html3/

TERMS TO KNOW

&& (and) Boolean Operator
The boolean && operator takes two arguments and returns false unless both inputs are true.

(or) Boolean Operator
The boolean || operator outputs true if either inputs are true (or both).

Boolean Operators
Boolean operators (also known as logical operators) allow us to build and combine more complex boolean expressions from more basic expressions.

Short-Circuit Evaluation
Short-circuit evaluation is when the evaluation of a logical expression stops because the overall value is already known.

Truth Table

A truth table is a small table that lists every possible input value and gives results for the operators.

not Boolean Operator

The not boolean operator (in Java represented by !) works with one argument and returns the opposite truth value of the argument.