Lesson 7 Boolean

Back in Lesson 2 we looked at three fundamental variable types… *int*, *double*, and *String*. Here, we look at another very important type…..***boolean***. This type has only two possible values… ***true*** or ***false***.

Only two values:

Let's look at some statements that could come out either *true* or *false*. Suppose we know that x = 3 and also that y = 97. What could we say about the truth (or falseness) of the following statements?

((x <10) AND (y = 97)) Both parts are *true* so the **whole** thing is *true*.

((x <10) AND (y = -3)) First part is *true*, second part is *false*, **whole** thing *false*

((x <10) OR (y = 97)) If either part is *true* (both are) the **whole** thing is *true*.

((x <10) OR (y = -3)) If either part is *true* (first part is) the **whole** thing *true*.

Correct syntax:

In the above examples there are three things we must change in order to have correct Java syntax:

1. To compare two quantities…such as (y = 97) above we must instead do it this way:

   (y = = 97)….recall that a single "=" is the assignment operator.

   Similarly, read y != 97 as "y is not equal to 97".

2. In Java we don't use the word "and" to indicate an **AND** operation as above. We use "**&&**" instead…….. ((x <10) && (y = = 97))

3. In Java we don't use the word "or" to indicate an **OR** operation as above. We use "||" instead……..((x <10) | | (y = = 97))

Truth tables:

Here are truth tables that show how && and || work for various combinations of *a* and *b*:

a	b	(a && b)
false	false	false
false	true	false
true	false	false
true	true	true

Table 7.1 And

a	b	(a || b)
false	false	false
false	true	true
true	false	true
true	true	true

Table 7.2

Negation (not) operator:
Another operator we need to know about is the **not** operator (!). It is officially called the negation operator. What does it mean if we say **not true** (!true)? … **false**, of course.

1. System.out.println(!true); //false
2. System.out.println(!false); //true
3. System.out.println(!(3 < 5)); //false
4. System.out.println(!(1 = = 0)); //true

Creation of *boolean*s:

Create *boolean* variables as shown in the following two examples:

```
boolean b = true;
boolean z = ( (p < j) && (x != c) );
```

Use the following code for example 1 – 10 below:

```
int x =79, y = 46, z = -3;
double d = 13.89, jj = 40.0;
boolean b = true, c = false;
```

1. System.out.println(true && false); //false
2. System.out.println(true && !false); //true
3. System.out.println(c | | (d > 0)); //true
4. System.out.println(!b || c); //false
5. System.out.println((x >102) && true); //false
6. System.out.println((jj = = 1) | | false); //false
7. System.out.println((jj = = 40) && !false); //true
8. System.out.println(x != 3); //true
9. System.out.println(!(x!=3)); //false
10. System.out.println(!!true); //true

Operator precedence:

Consider a problem like:

```
System.out.println( (true && false) | | ( (true && true) | | false ) );
```

We can tell what parts we should do first because of the grouping by parenthesis. However, what if we had a different problem like this?

```
System.out.println( false && true || true);
```

Which part should we do first? The answers are different for the two different ways it could be done. There is a precedence (order) for the operators we are studying in this lesson (see Appendix H for a complete listing of operator precedence). The order is:

! = = != && ||

Example 1

```
System.out.println( true || false && false);          //true
```

Do the false && false part first to get a result of false.
Now do true || false to get a final result of true.

Example 2

```
System.out.println( true && false || false);          //false
```

Do the true && false part first to get a result of false.
Now do false || false to get a final result of false.

Using a search engine:

You can use your knowledge of Booleans on the Internet. Go to your favorite search engine and type in something like,

"Java script" and "Bill Gates"

and you will find only references that contain **both** these items.
On the other hand, enter something like,

"Java script" or "Bill Gates"

and you will be overwhelmed with the number of responses since you will get references that contain **either** of these items.

You should be aware that the various search engines have their own rules for the syntax of such Boolean searches.

Now that we have learned to write a little code, it's time to turn to another part of our Computer Science education. Computers haven't always been as they are today. Computers of just a few years ago were primitive by today's standards. Would you guess that the computers that your children will use someday would make our computers look primitive? Take a few minutes now to review a short history of computers in Appendix S.