Conditional Statements

BOOLEAN MATH

You can also do math with booleans, but this is not the standard type of math you've seen before it is it's own type of math. Because what would true + true mean? Or false + true? So you need to have some kind of different combinations.

Not

The simplest form of boolean math is a not operation. The symbol is ! (also sometimes called bang). It flips a boolean between false and true.

```
boolean funny = !false; // It's funny because it's true!

funny = !funny; // sorry, that joke was not funny...now it is false
```

You can flip any boolean from false to true or true to false.

And

You can also combine two booleans. One way to do that is with an and. An and looks like this

```
b1 && b2
```

An and is only true if both booleans are true (think if b1 AND b2 are true). So take a look at this truth table. The top row represents the values of b1 the first column represents the values of b2, the intersection cells represent the value of b1 && b2 in that case (like a Punnet square).

&&	T	F
T	T	F
F	F	F

Or

Another way to combine booleans is with an or. An or looks like this

```
b1 || b2
```

That character is a pipe. It is the character on the same key on the backslash (\ ) right above the enter key on most keyboards.

An or is true if at least one of the booleans are true (think if b1 OR b2 is true, then the whole thing is true). This is what the truth table looks like:

||	T	F
T	T	T
F	T	F

Chaining ands and ors

One question that comes up a lot is whether you can do an && or an || with more than 2 booleans. And the answer is yes and no. You can write b1 && b2 && b3 || b4, but it is combining them all as pairs. It is really doing an order of operations. You may need to add parentheses to make sure that things are clear.

Creating booleans

Those are just ways to do work with existing booleans, but you can also create booleans from other types by testing their values against other values.

Equality

First you can do equality tests. You can test any value against any other value. To do this you use a double equals (==)

```
int c = 3;

c == 4 // this would be false

c == 3 // this would be true
```

Notice that you have to use a double equals because if you used a single equals sign you would be setting a value for the variable, so you need to do something different to set up as a test.

You can also test for if something is not equal like this:

```
int c = 3;

c != 4 // this would be true

c != 3 // this would be false
```

Not equals uses the not symbol and a single equals sign.

Strings are a little bit different, for reasons that will explained later. For now, just know that if you test for a string being equal with a == it might look like it works, but it won't work like you expect. You have to do it like this:

```
String s = "s";

s.equals("s") // this would be true

!s.equals("s") // this is how you check for not equals
```

Inequality

You can do some inequality tests with integers and floats. Greater than and less than are pretty straight forward:

```
int c = 3;

c > 4 // this would be false

c < 4 // this would be true
```

You can also do greater than or equals and less than or equals, but those are not the easiest things to type so you have to write them as two separate symbols:

```
int c = 3;

c <= 4 // this would be true

c >= 3 // this would be false
```

CONDITIONAL STATEMENTS

These have all been ways to construct booleans on their own. While you can create booleans as variables and print them out, but most often you want to make decisions based on them. You want to run a block of code only if a particular boolean is true. For this you use a conditional statement, also called an if statement.

That looks like this

```
if ( /* any boolean goes here */) {

    // Some code to run when the boolean is true goes here

}
```

Note that you don't have any semicolons on the lines that end in curly braces. Any boolean can go inside the parentheses, one that is constructed like the above. Almost any code can go in the curly braces, such as another if statement, math, or other things we'll see later on.

So if you want to print a line out if a integer is greater than 3 it would look like this

```
if (c > 3) {

    System.out.println("Greater than 3");

}
```

If c is 2, it will output nothing.

We can also have a series of decisions with if-else if-else statements. If you set up one of these, it will go into the first block of code where the condition is true, but none of the others (even if the conditions are true). An else statement has no condition, but it is a catch all for everything else. So if there is an else

statement and all the other conditions are false, the else statement will run. So let's take a look at this block of code:

```
if (x > 2) {
    System.out.println("x");
} else if (y > 2) {
    System.out.println("y");
} else {
    System.out.println("else");
}
```

So if x is 5 and y is 5, then it will print out x, because x > 2 and it doesn't do any of the other checks. If x is 1 and y is 5, it will print out y. If x is 1 and y is 1, then it will print out else.

TRY IT OUT!

Write a basic calculator. Let the user input what operation they want to do, then input some numbers and output the result.