Set 28: Boolean Expressions

Skill 28.01: Interpret a Boolean expression
Skill 28.02: Apply the AND and NOT operators
Skill 28.03: Apply operator precedence
Skill 28.04: Apply De Morgan's law
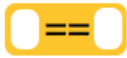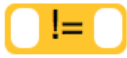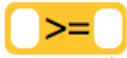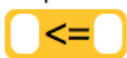**Skill 28.05: Write functions that return Boolean expressions**

Skill 28.01: Interpret a Boolean expression

Skill 28.01 Concepts

- A **Boolean value** is simply a computer science-y term that means **a true/false value**.
- A **Boolean expression** is a statement that *evaluates* to a *Boolean value* (a single true/false).

To determine whether two values are the same or not the same, or whether one value is greater or less than another value requires a *comparison operator*.

Below is a list of comparison operators commonly used in computer science.

== is equal to **!=** is not equal to **>** is greater than **<** is less than **>=** is greater than or equal to **<=** is less than or equal to	To the left are 6 common **comparison operators**. Each compares a value on the left with a value on the right and returns a Boolean value – **true** or **false**. Most of these do what you would expect. Why these symbols: **==, !=, <=,** and **>=**? 1. We use == because the single equal sign = is the assignment operator. We need something different to indicate we want to compare two values instead of assign one to the other. **Common mistake:** writing something like if (age = 18) instead of if (age == 18). We'll make sure we get this down later. 2. We use != , <= , and >= because they only require ASCII symbols. Historically the mathematical symbols ≠, ≤ and ≥ were hard or impossible to produce on some computer systems. The ! is universally read as "not".

Skill 28.01 Exercise 1

Skill 28.02 Concepts

Consider the following example. Suppose that we know that x = 3 and y = 97. Below are statements about x and y and whether they are true or false individually, AND whether the entire statement is true or false.

Statement	True or False
`(( x < 10 ) AND ( y = 97 )`	First part is true, second part is true, entire statement is true
`(( x < 10 ) AND ( y = -3 ))`	First part is true, second part is false, entire statement is false
`(( x < 10 ) AND ( y ≠-3 ))`	First part is true, second part is true, entire statement is true
`(( x < 10 ) OR ( y = 97 ))`	Either part is true, entire statement is true
`(( x < 10 ) OR ( y = -3 ))`	Either part is true, entire statement is true

The above examples illustrate how true and false statements are evaluated, however the syntax is not correct. In order for java to correctly read the syntax the "AND" and "OR" statements must be replaced with the correct symbols as shown below,

Statement	True or False
((x < 10) && (y = = 97)	And is replaced with &&, the = is replaced with "= ="
((x < 10) && (y = = -3)	And is replaced with &&, the = is replaced with "= ="
((x < 10) && (y != -3)	And is replaced with &&, the ≠ is replaced with "!="
((x < 10) || (y = = 97))	Or is replaced with ||, the = is replaced with "= ="
((x < 10) || (y = = -3))	Or is replaced with ||, the = is replaced with "= ="

Skill 28.02 Exercises 1 & 2

Skill 28.03 Concepts

Just like math operations follow an order of precedence, so too do operations like AND (&&) and OR (||). Consider a problem like,

```
console.log( false && true || true );
```

Which part do we do first? As it turns out we would first do && and then ||. Because false and true are not the same thing, false && true evaluates to false. Next we consider false or true, which is true. The order of precedence for the operators we are studying are as follows,

<div align="center">

!   ==   !=   && ||

</div>

To help avoid the confusion of the order of execution of Boolean statements, you can use parentheses like below,

```
console.log( ( true && false ) || ((true && false) || false) );
```

Skill 28.03 Exercise 1

Skill 28.04: Apply De Morgan's Law

Skill 26.04 Concepts

DeMorgan's Theorems are basically two sets of rules or laws developed from the Boolean expressions for AND, OR and NOT using two input variables, A and B. These two rules or theorems allow the input variables to be negated and converted from one form of a Boolean function into an opposite form.

DeMorgan's first theorem states that two (or more) variables NOR´ed together is the same as the two variables inverted (Complement) and AND´ed, while the second theorem states that two (or more) variables NAND´ed together is the same as the two terms inverted (Complement) and OR´ed. *That is replace all the OR operators with AND operators, or all the AND operators with an OR operators.*

The video below explains this more clearly.

Booleans		Numbers	
Original	**Equivalent**	**Original**	**Equivalent**
!(A && B)	!A || !B	!(A > B)	A <= B
!(A || B)	!A && !B	!(A < B)	A >= B
!(A == B)	!A != !B or A != B	!(A >= B)	A < B
		!(A <= B)	A > B
!(A != B)	!A == !B or A == B	!(A == B)	A != B
		!(A != B)	A == B

https://youtu.be/AGyjo2DLxjM

Skill 28.04 Exercises 1 thru 3

Skill 28.05 Concepts

In the previous lesson we wrote functions that returned numeric and text (also called String) values. For example,

```
function rectangleArea(width, height){
    var area = width * height;
    return area;
}

console.log(rectangleArea(5, 7)); //prints 35
```

We can also write functions that return Boolean values

```
function rectangleArea(width, height){
    var area = width * height;

    return (area > 100);
}

var myArea = rectanglearea(10, 5);
console.log(myArea);//prints false
```

Skill 28.05 Exercises 1 & 2