Learning Objectives - Boolean Operators

- **Recognize the difference between = and ==**
- **Be comfortable with evaluating boolean statements**
- **Be comfortable with the AND and OR operators**

Equal To & Not Equal To

Boolean operators are operators that return a boolean value (true or false).

Equal To

Java uses the == operator to determine equality. Beginners often confuse the = and the == operators. Remember, = is the assignment operator.

```java
int a = 5;
int b = 5;
System.out.println(a == b);
```

challenge

What happens if you:

- Change `b` to `1`?
- Change `a` to `boolean a = true;` and `b` to `boolean b = false;`?

Not Equal To

The `!=` operator checks to see if two values are not equal.

```java
int a = 5;
int b = 5;
System.out.println(a != b);
```

challenge

What happens if you:

- Change `b` to `1`?
- Change `a` to `boolean a = true;` and `b` to `1`?
- Change `b` to `boolean b = false;`?

Less Than & Less Than or Equal To

Less Than

The < operator is used to check if one value is less than another value.

```
int a = 5;
int b = 7;
System.out.println(a < b);
```

challenge

What happens if you:

- Change `b` to `1`?
- Change `b` to `5`?
- Change `b` to `false`?

Less Than or Equal To

The <= operator is used to check if one value is less than or equal to another value.

```
int a = 5;
int b = 7;
System.out.println(a <= b;)
```

challenge

What happens if you:

- Change `b` to `1`?
- Change `b` to `5`?
- Change `a` to `false` and `b` to `true`?

Greater Than & Greater Than or Equal To

Greater Than

The `>` operator is used to check if one value is greater than another value.

```
int a = 9;
int b = 17;
System.out.println(a > b);
```

challenge

What happens if you:

- Change `b` to `1`?
- Change `b` to `9`?
- Change `b` to `false`?

Greater Than or Equal To

The `>=` operator is used to check if one value is greater than or equal to another value.

```
int a = 9;
int b = 17;
System.out.println(a >= b);
```

challenge

What happens if you:

- Change `b` to `1`?
- Change `b` to `9`?
- Change `a` to `true` and `b` to `false`?

And

The && Operator

The && operator allows for compound (more than one) boolean expressions. All boolean expressions **must** be true in order for the whole thing to be true. If only one boolean expressions is false, then the whole thing is false.

```java
boolean a = true;
boolean b = true;
boolean c = false;
System.out.println(a && b);
```

▼ **How do I type &&?**

It is towards the top of the keyboard, above the number 7. This means you must hold shift and press the 7 key to type &.

challenge

What happens if you:

- Change the code to `System.out.println(a && c);`?
- Change the code to `System.out.println(c && b);`?

Multiple && Statements

You can chain several && statements together. They are evaluated in a left-to-right manner.

```java
boolean a = true;
boolean b = true;
boolean c = false;
System.out.println(a && b && c);
```

challenge

What happens if you:

- Change the code to `System.out.println(a && b && a && b && a);`?
- Change the code to `System.out.println(a && b && a && b && c);`?

Or

The `||` Operator

The `||` operator allows for compound (more than one) boolean expressions. If only one boolean expressions is true, then the whole thing is true. To be false, **all** boolean expressions must be false.

```
boolean a = true;
boolean b = true;
boolean c = false;
boolean d = false;
System.out.println(a || b);
```

▼ **How do I type `||`?**

It is on the right-hand side, below the backspace/delete key and above the enter/return key. The `|` symbol is the line above the `</code>. This means you must hold shift and press the </code> key to type |.`

challenge

What happens if you:

- Change the code to `System.out.println(a || c);`?
- Change the code to `System.out.println(c || d);`?

Multiple `||` Statements

You can chain several `||` statements together. They are evaluated in a left-to-right manner.

```java
boolean a = true;
boolean b = true;
boolean c = false;
System.out.println(a || b || c);
```

challenge

What happens if you:

- Change the code to

  `System.out.println(a || c || c || c || c);`?

- Change the code to

  `System.out.println(c && c && c && c && c);`?

Not

The `!` Operator

The `!` operator produces the opposite of the boolean expression that it modifies.

```
System.out.println(! true);
```

challenge

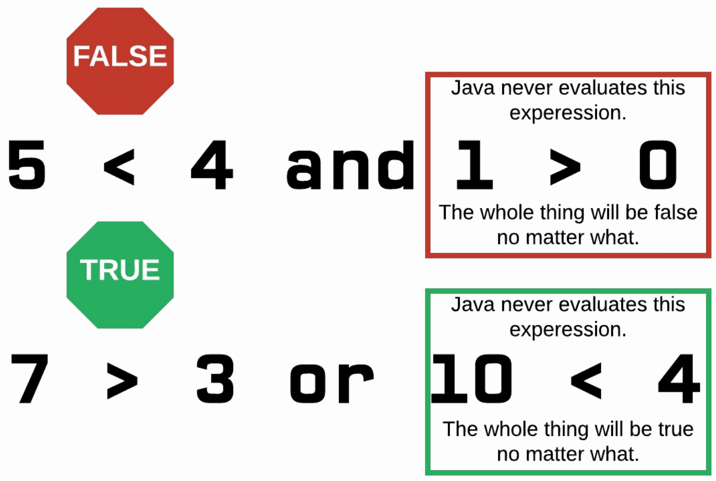
What happens if you:

- Change the code to `System.out.println(! true && false);`?
- Change the code to `System.out.println(! (true && false));`?
- Change the code to `System.out.println(! ! true);`?

Short-Circuiting

Short Circuiting

If Java can determine the result of a boolean expression before evaluating the entire thing, it will stop and return the value.


FALSE
5 < 4 and 1 > 0
Java never evaluates this experession.
The whole thing will be false no matter what.
TRUE
7 > 3 or 10 < 4
Java never evaluates this experession.
The whole thing will be true no matter what.

Short Circuiting

```
System.out.println( false
                             &&
/*Java never reaches this line*/ true);

System.out.println( true
                            ||
/*Java never reaches this line*/ false);
```