# boolean type

### 1. Boolean type

As we have already seen, Java has the super useful if-else statement. It executes one block of statements if the condition in parentheses is true, and a second block of statements if the condition is false.

For convenience when working with expressions that can be either true or false, Java's creator added the special boolean type. Its main feature is that variables of this type can take only two values: true and false.

It is impossible to assign any other values to boolean variables. The compiler won't allow it.

#### And why do we need such a primitive type?

Well, the good thing is that you can use it to store the values of logical expressions. Example:

```
Code
                                                      Explanation
boolean isOK = true;
                                                      The boolean isOK variable contains the value true
boolean hasError = false;
                                                      The boolean hasError variable contains the value false
int age = 70;
                                                      The boolean isSenior variable contains the value true
boolean isSenior = (age > 65);
int record = 612;
                                                      The boolean hasNewRecord variable contains the value true
int value = 615;
boolean hasNewRecord = (value > record);
                                                      The boolean <code>isIce</code> variable contains the value <code>true</code>
int min = 0;
int max = 100;
                                                      The boolean isSteam variable contains the value false
int temperature = -20;
boolean isIce = (temperature < min);</pre>
boolean isSteam = (temperature > max);
```

# 2. Using boolean variables

Boolean variables would be of little use if they could only store the results of expressions. The point here is that you can also use them. Where? Wherever you can write a logical expression.

For example, you can use a boolean variable in condition of an if statement:

In this example, there is little benefit gained from making this replacement, but when programs grow larger, their conditions become more complex. You will be convinced of this in the near future.

## 3. Comparison operators

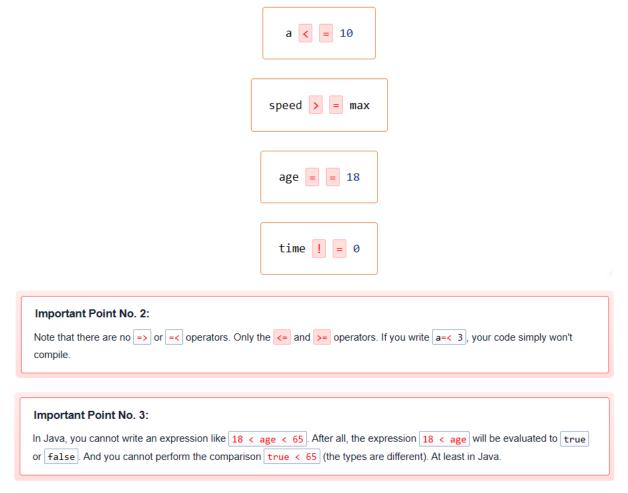
In Java, as in other programming languages, it is often necessary to compare variables with one another. And Java has just the operators you need to make comparisons:

Operator	Explanation	Example
<	Less than	a < 10
>	Greater than	b > a
<=	Less than or equal	a <= 10
>=	Greater than or equal	speed >= max
	Equals	age == 18
!=	Not equals	time

The above operators are used to produce logical expressions. The results can be stored in boolean variables or used as the condition of an if statement.

```
Important Point No. 1:
The operators that consist of two characters cannot be split apart.
```

In other words, code like this won't compile:



What can be done? You will find the answer to this question in the next lesson.