Logical operators

1. Boolean logic

In Java, you can't write the expression `18 < age <65`. That is incorrect syntax and the program won't will not compile.

But you can write it like this:

```
(18 < age) AND (age < 65)
```

Of course, instead of the word `AND`, there would be a *logical operator*. We'll talk about them in more detail now.

There are three logical operators in Java: `AND` (&&), `OR` (||) and `NOT` (!).

The **good news** is that you can use *parentheses* to construct logical expressions of any complexity.

The **bad news** is that Java developers decided to use notation from the *C language* instead of the words `and`, `or` and `not`.

Look at the screen:

Logical operator	Expectation	Reality
`AND` (∧)	`and`	`&&`
`OR` (∨)	`or`	`||`
`NOT` (¬)	`not`	`!`

Here are some examples of using logical operators in Java:

Expression	Interpretation	Explanation
`(0 < a) && (a < 100)`	`(0 < a) and (a < 100)`	`(0 < a) and (a < 100)`
`(!a) && (!b)`	`(not a) and (not b)`	`(NOT a) AND (NOT b)`
`!(!a || !b)`	`not((not a) or (not b))`	`NOT((NOT a) OR (NOT b))`

2. Examples of using comparison operators and boolean variables

Wherever you can write a logical expression, you can write a logical variable.

Example:

Code	Explanation

`int age = 35;`
`if (age >= 18 && age <= 65)`
`  System.out.println("You can work");`
	If the value of age is between `18` and `65`, then the phrase "You can work" is displayed.

`int age = 35;`
`boolean isYoung = (age < 18);`
`if (!isYoung && age <= 65)`
`  System.out.println("You can work");`
	We created an `isYoung` variable and moved the first part of the expression into it. We simply replaced `age >= 18` with `age < 18`.

`int age = 35;`
`boolean isYoung = (age < 18);`
`boolean isOld = (age > 65);`
`if (!isYoung && !isOld)`
`  System.out.println("You can work");`
	We created an isOld variable and moved the second part of the expression into it. Additionally, we replaced `age <= 65` with `age > 65`.

These three examples are equivalent. Only in the second example did we move part of the expression from the `if` statement into a separate boolean variable (`isYoung`). In the third example, we moved the second part of the expression into a second variable (`isOld`).

3. Logical arithmetic

Let's briefly go through logical operations.

The `AND` operator is `&&`, also known as *conjunction*.

Expression	Result
`true && true`	`true`
`true && false`	`false`
`false && true`	`false`
`false && false`	`false`

In other words, the result of an expression is `true` only if both values that make up the expression are `true`. Otherwise, it is always `false`.

The `OR` operator is `||`, also known as *disjunction*.

Expression	Result
`true || true`	`true`
`true || false`	`true`
`false || true`	`true`
`false || false`	`false`

In other words, the result of an expression is always `true` if at least one term in the expression is `true`. If both are `false`, then the result is `false`.

The `NOT` operator is `!`, also known as the *logical inverse*.

Expression	Result
`!true`	`false`
`!false`	`true`

The operator changes `true` to `false` and vice versa.

Useful expressions:

Expression	Result
`m && !m`	`false`
`m || !m`	`true`
`!(a && b)`	`!a || !b`
`!(a || b)`	`!a && !b`