Theory: Boolean and logical operations

8 minutes 5 / 10 problems solved

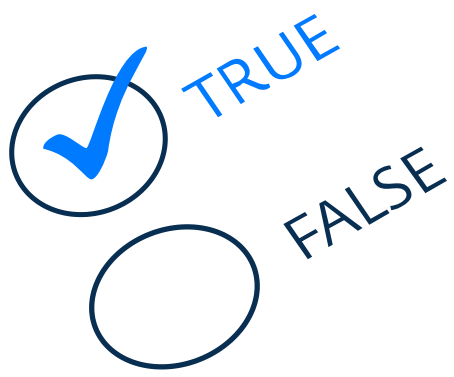
Start practicing

17113 users solved this topic. Latest completion was about 1 hour ago.

§1. Boolean type

The `boolean` is a data type that has only two possible values: `false` and `true`. This is also known as the **logical type**.

This type is a common way in programming languages to represent something that has only two opposite states like *on* or *off*, *yes* or *no*, etc.



If you are writing an application that keeps track of door's openings you'll find it natural to use `boolean` to store current door state.

```
boolean open = true;
boolean closed = false;

System.out.println(open);   // true
System.out.println(closed); // false
```

Important, you cannot assign an integer value to a `boolean` variable. In Java, 0 is not the same as `false`.

§2. Logical operators

Variables of the `boolean` type are often used to build logical expressions using logical operators. Java has four logical operators **NOT**, **AND**, **OR** and **XOR**:

- **NOT** is a unary operator that reverses the boolean value. It is denoted as `!`.

```
boolean f = false; // f is false
boolean t = !f;    // t is true
```

- **AND** is a binary operator that returns `true` if both operands are `true`, otherwise, it is `false`. It is denoted as `&&`.

```
boolean b1 = false && false; // false
boolean b2 = false && true;  // false
boolean b3 = true && false;  // false
boolean b4 = true && true;   // true
```

- **OR** is a binary operator that returns `true` if at least one operand is `true`, otherwise, it returns `false`. It is denoted as `||`.

```
boolean b1 = false || false; // false
boolean b2 = false || true;  // true
boolean b3 = true || false;  // true
boolean b4 = true || true;   // true
```

- **XOR** (**exclusive OR**) is a binary operator that returns `true` if boolean operands have different values, otherwise, it is `false`.

Current topic:

Boolean and logical operations (Stage 2)

Topic depends on:

Types and variables (Stage 1)

Topic is required for:

Relational operators (Stage 2)

Regexps in Java

Table of contents:

↑ Boolean and logical operations

§1. Boolean type

§2. Logical operators

§3. The precedence of logical operators

§4. An example: trekking

§5. Short-circuiting evaluation

Feedback & Comments

```
boolean b1 = false ^ false; // false
boolean b2 = false ^ true;  // true
boolean b3 = true ^ false;  // true
boolean b4 = true ^ true;   // false
```

The XOR operator is used less often than others. Just remember that Java has it. If you really need it, you can use it.

§3. The precedence of logical operators

Below are the logical operations sorted in order of decreasing their priorities in expressions: `!` (NOT), `^` (XOR), `&&` (AND), `||` (OR).

So, the following variable is `true`:

```
boolean b = true && !false; // true, because !false is evaluated first
```

To change the order of execution you can use round brackets `(...)`.

§4. An example: trekking

As an example, let's write a complex boolean expression that determines the possibility of trekking in summer and in other seasons.

```
boolean cold = false;
boolean dry = true;
boolean summer = false; // suppose now is autumn

boolean trekking = dry && (!cold || summer); // true, let's go to trek!
```

Do not get confused in the expression above, otherwise, you will go on the trek in bad weather :) A programmer should understand not only arithmetic but also logical operations.

§5. Short-circuiting evaluation

An interesting thing is that the `&&` and `||` operators don't evaluate the second argument if it isn't necessary. When the first argument of the `&&` operator evaluates to `false`, the overall value must be `false`; and when the first argument of the `||` operator evaluates to `true`, the overall value must be `true`. So:

- `false && ...` -> `false`, since it is not necessary to know what the right-hand side is;
- `true || ...` -> `true`, since it is not necessary to know what the right-hand side is.

This behavior is known as **short-circuit evaluation** (do not confuse it with an electrical short circuit). It reduces the computation time, but can also be used to avoid some errors in programs. We will discuss this in the following topics.

Report a typo

Thanks for your feedback!

Start practicing

Comments (13) Hints (0) Useful links (0) Show discussion