Theory: Conditional statement

The **conditional statement** is a construction that allows a program to perform different computations depending on the value of a Boolean expression. If it is `true`, the program performs one computation; otherwise, it is `false`, the program performs another computation. Here are some examples of Boolean expressions: `a > b`, `i - j == 1` and so on.

The conditional statement has different forms. We will use all of them.

§1. The single if-case

The simplest form of the conditional statement consists of the keyword `if`, a Boolean expression and a body enclosed in curly braces.

```
if (expression) {
    // body: do something
}
```

If the expression is `true`, the statements inside the code block are executed; otherwise, the program skips them.

See the following example.

```
int age = ...; // it has a value
if (age > 100) {
    System.out.println("Very experienced person");
}
```

In this example, if the `age` is greater than 100 the code prints **"Very experienced person"**, otherwise, it does nothing.

Sometimes you will see a situation when the expression in a condition is a single `boolean` type variable. Instead of writing `b == true` or `b == false`, use this variable (or its negation with `!`) as the Boolean expression:

```
boolean b = ...; // it is true or false
if (b) { // or !b
    // do something
}
```

A conditional statement can be used in any place in a program where the statement is expected. It can be even nested inside another conditional statement to perform multistage checks.

§2. The if-else-cases

The if-case above can be extended with the keyword `else` and another body to do alternative actions when the expression is `false`.

```
if (expression) {
    // do something
} else {
    // do something else
}
```

In this case, if the expression is `true`, then the first code block is executed; otherwise, the second code block is executed, but not both together.

In the example below, the program outputs different text depending on the value of `num` (even or odd).

> Note, a number is even if it can be divided exactly by 2; otherwise it's odd.

Current topic:

Conditional statement — Stage 2

Topic depends on:

Relational operators — Stage 2

Topic is required for:

Ternary operator — Stage 2

What is an exception — Stage 7

```
int num = ...; // the num is initialized by some value

if (num % 2 == 0) {
    System.out.println("It's an even number");
} else {
    System.out.println("It's an odd number");
}
```

Since a number can only be even or odd, only one message will be displayed. If `num` is 10, the program outputs `"It's an even number"`. If the value is 11, it outputs `"It's an odd number"`.

§3. The if-else-if-cases

The most general form of the conditional statement consists of several conditions and `else`-branches.

```
if (expression0) {
    // do something
} else if (expression1) {
    // do something else 1
// ...
} else if (expressionN) {
    // do something else N
}
```

The following code outputs recommendations about what computer you need to buy depending on your budget.

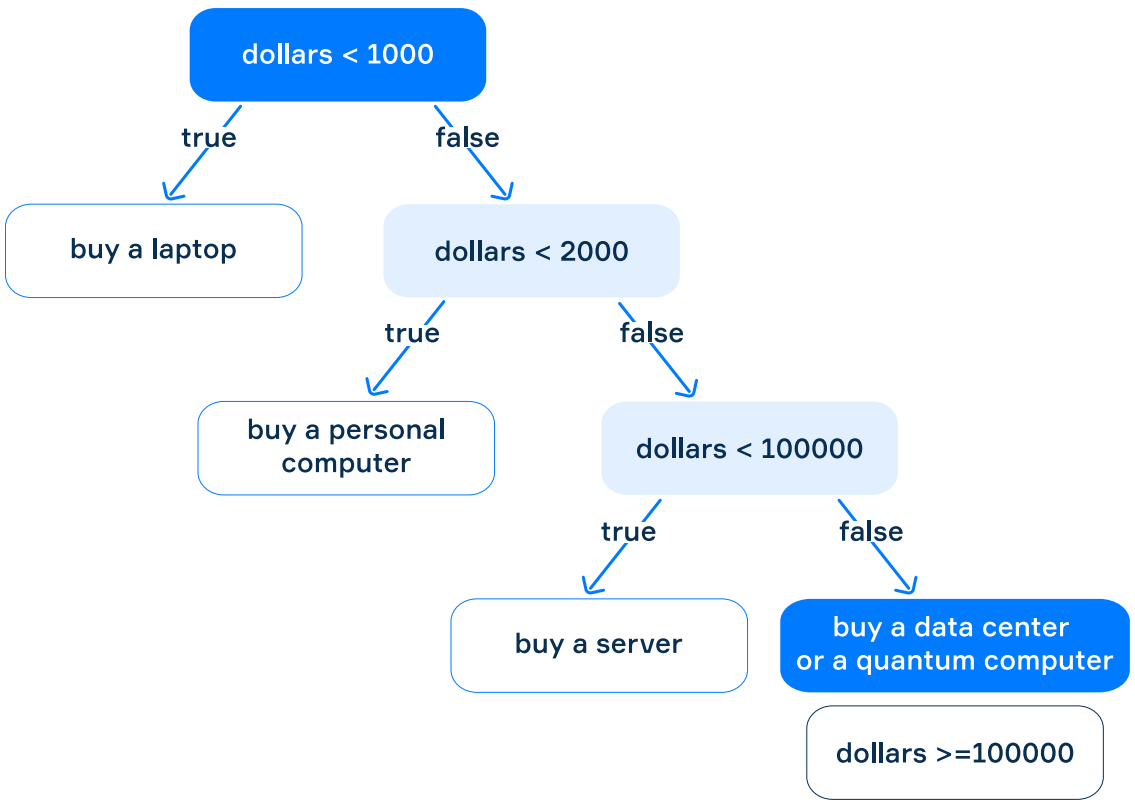
```
long dollars = ...; // your budget

if (dollars < 1000) {
    System.out.println("Buy a laptop");
} else if (dollars < 2000) {
    System.out.println("Buy a personal computer");
} else if (dollars < 100_000) {
    System.out.println("Buy a server");
} else {

    System.out.println("Buy a data center or a quantum computer");

}
```

This conditional statement has four branches: `dollars < 1000`, `dollars < 2000`, `dollars < 100_000` and `dollars >= 100_000`. If the value of dollars is `10_000` it prints `"Buy a server"`.

A conditional statement with multiple branches creates a **decision tree**, whose nodes consist of boolean expressions, and each branch is marked with *true* or *false*. The *true*-branch leads to a block of statements to be executed and a `false`-branch leads to the next condition to be checked. The last false-branch means "*in all other cases*".

> When talking about conditions programmers often use the term "control flow statements". **Control flow** is the order in which various parts of a program are executed. You will probably meet this term in our topics and on other external resources.

The picture below demonstrates such a tree for the example with computers.



Decision tree for buying a laptop

This example completes our examination of conditional statements.

Report a typo

1324 users liked this theory. **7** didn't like it. **What about you?**

Start practicing

Comments (13) Hints (1) Useful links (0) Show discussion