Java → Basic syntax and simple programs → Control flow statements → <u>Branching</u> <u>statements</u>

Theory: Branching statements

© 39 minutes 5 / 16 problems solved

Start practicing

11013 users solved this topic. Latest completion was 17 minutes ago.

Branching statements are used to alter the standard behavior of loops; they can terminate a loop or skip some iterations.

§1. The break statement

The break statement has two uses:

- it terminates the current loop of any type (for, while, do-while);
- it terminates a case in the **switch** statement;

In this topic, we will learn how to use it to terminate loops.

The following example demonstrates a loop that includes one break.

```
int i = 10;
while (true) { // the condition to continue the loop
    if (i == 0) { // the condition to perform break that stops this loop
        break;
}
i--;
}
```

In the code above, the condition to continue the loop is always true, but it will be successfully stopped when the variable i becomes 0 through the use of break inside the conditional statement.

The **break** statement terminates only the loop in which it is currently located. If this loop is performed inside another loop, the outer loop won't be stopped.

The following code prints a ladder of numbers.

```
for (int i = 0; i < 10; i++) {
    for (int j = 0; j < 10; j++) {
        System.out.print(j + " ");
        if (i == j) {
            break;
        }
     }
     System.out.println();
}</pre>
```

The break statement can't stop the outer loop (with variable i) and the code prints:

To stop the outer loop we'd like to declare a Boolean variable stopped and use it as a special Boolean flag.

```
Current topic:

Stage 2

Branching statements

Topic depends on:

The for-loop

The while and do-while loops

Topic is required for:

Stage 2

The while and do-while loops

Calling a method

Iterating over arrays

Stage 2

Processing strings

Stage 2

Processing strings
```

Table of contents:

1 Branching statements

§1. The break statement

§2. The continue statement

Feedback & Comments

https://hyperskill.org/learn/step/3507

```
boolean stopped = false;
for (int i = 0; (i < 10) && !stopped; i++) {
    for (int j = 0; j < 10; j++) {
        System.out.print(j + " ");
        if (i == j) {
            stopped = true;
            break;
        }
    }
}

System.out.println();

System.out.println();

</pre>
```

Now, the program's output is not the same:

```
1 | 0
```

There is another way to stop the outer loop: labeled break operator. However, it's not good practice to use it. Google it if you are really interested.

§2. The continue statement

It causes a loop to skip the current iteration and go to the next one.

This statement can be used inside any kind of loops.

- inside the **for-loop**, the continue causes control to immediately move to the increment/decrement statement;
- inside the **while** or **do-while loop**, control immediately moves to the condition.

In the following example, a sequence of numbers is output. Odd numbers are skipped.

```
int n = 10;
for (int i = 0; i < n; i++) {
    if (i % 2 != 0) {
        continue;
    }
    System.out.print(i + " ");
}</pre>
```

The output:

```
1 | 02468
```

The **continue** statement and the **break** statement only affect the loop in which they are located. The **continue** statement cannot skip the current iteration of the outer loop.

Often, we can rewrite our loop without using the continue statement. Here is an example:

```
int n = 10;
for (int i = 0; i < n; i++) {
    if (i % 2 == 0) {
        System.out.print(i + " ");
    }
}</pre>
```

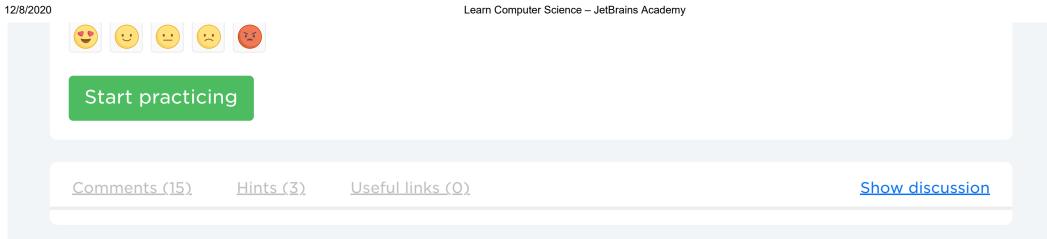
The result is the same as above, but the code became shorter and more readable.

It is important to note that the widespread use of branching statements leads to poorly-structured code because conditions in your loops are not actually what you need to do. So, use them wisely — only when it helps to make code shorter and easier to understand for humans.

Report a typo

984 users liked this theory. 11 didn't like it. What about you?

https://hyperskill.org/learn/step/3507



https://hyperskill.org/learn/step/3507 3/3