

# 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`.

```
1  int i = 10;
2  while (true) { // the condition to continue the loop
3      if (i == 0) { // the condition to perform break that stops this loop
4          break;
5      }
6      i--;
7  }
```

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.

```
1  for (int i = 0; i < 10; i++) {
2      for (int j = 0; j < 10; j++) {
3          System.out.print(j + " ");
4          if (i == j) {
5              break;
6          }
7      }
8      System.out.println();
9  }
```

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

```
1  0
2  0 1
3  0 1 2
4  0 1 2 3
5  0 1 2 3 4
6  0 1 2 3 4 5
7  0 1 2 3 4 5 6
8  0 1 2 3 4 5 6 7
9  0 1 2 3 4 5 6 7 8
10 0 1 2 3 4 5 6 7 8 9
```

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

Current topic:

✓ [Branching statements](#) Stage 2 ...

Topic depends on:

✓ [The for-loop](#) Stage 2 ...

✓ [The while and do-while loops](#) Stage 2 ...

Topic is required for:

✓ [Switch statement](#) Stage 3 ...

✓ [Calling a method](#) Stage 2 ...

[Iterating over arrays](#) ...

[Processing strings](#) Stage 2 ...

Table of contents:

[1 Branching statements](#)

[§1. The break statement](#)

[§2. The continue statement](#)

[Feedback & Comments](#)

```

1  boolean stopped = false;
2  for (int i = 0; (i < 10) && !stopped; i++) {
3      for (int j = 0; j < 10; j++) {
4          System.out.print(j + " ");
5          if (i == j) {
6              stopped = true;
7              break;
8          }
9      }
10     System.out.println();
11 }

```

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.

```

1  int n = 10;
2  for (int i = 0; i < n; i++) {
3      if (i % 2 != 0) {
4          continue;
5      }
6      System.out.print(i + " ");
7  }

```

The output:

```

1  0 2 4 6 8

```

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:

```

1  int n = 10;
2  for (int i = 0; i < n; i++) {
3      if (i % 2 == 0) {
4          System.out.print(i + " ");
5      }
6  }

```

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?



Start practicing

[Comments \(15\)](#)

[Hints \(3\)](#)

[Useful links \(0\)](#)

[Show discussion](#)