Theory: The for-loop

40 minutes 5 / 14 problems solved

Start practicing

Sometimes we need to repeat a block of code a certain number of times. To do this, Java provides the `for`-loop. This loop is often used to iterate over a range of values or through an array. If the number of iterations or the range borders are known, it is recommended to use the `for`-loop. If they are unknown, the `while`-loop may be a preferable solution.

§1. The basic for-loop syntax

The `for`-loop has the following basic syntax:

```
for (initialization; condition; modification) {
    // do something
}
```

Parts of the loop:

- **initialization statement** is executed once before the loop begins; usually, loop variables are initialized here;
- **condition** is a Boolean expression that determines the need for the next iteration; if it's `false`, the loop terminates;
- **modification** is a statement that changes the value of the loop variables; it is invoked after each iteration of the loop; usually, it uses **increment or decrement** to modify the loop's variable.

Inside the loop's body, the program can perform any correct Java statements. It can even contain other loops.
The order of execution for any for-loop is always the same:

1. the initialization statement;
2. if the condition is `false` then terminate the loop;
3. if the condition is `true`, then loop's body is executed;
4. the modification is performed;
5. go to the stage 2 (condition).

Let's write a loop for printing integer numbers from 0 to 9 in the same line.

```
int n = 9;
for (int i = 0; i <= n; i++) {
    System.out.print(i + " "); // here, space is used to separate numbers
}
```

The code displays:

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

The variables declared in the initialization statement are visible only inside the scope that includes all parts of the loop: the condition, the body, and the modification. The integer loop's variables are often named as `i`, `j`, `k` or `index`.

Here's another example. Let's calculate the sum of the integer numbers from 1 to 10 (inclusive) using the for-loop.

```
int startIncl = 1, endExcl = 11;

int sum = 0;
for (int i = startIncl; i < endExcl; i++) {
    sum += i;
}

System.out.println(sum); // it prints "55"
```

Current topic:

✓ The for-loop — Stage 2

Topic depends on:

✓ Increment and decrement — Stage 1

✓ Ternary operator — Stage 2

Topic is required for:

✓ Branching statements — Stage 2

Table of contents:

§2. Skipping parts

The initialization statement, the condition, and the modification parts are optional, the for loop might not have all of them.

It is possible to declare a variable outside the loop:

```
int i = 10;
for (; i > 0; i--) {
    System.out.print(i + " ");
}
```

Moreover, it is also possible to write an infinite loop without these parts at all:

```
for (;;) {
    // do something
}
```

§3. Nested loops

It's possible to nest one for-loop into another for-loop. This approach is used to process multidimensional structures like tables (matrices), data cubes, and so on.

As an example, the following code prints the multiplication table of numbers from 1 to 9 (inclusive).

```
for (int i = 1; i < 10; i++) {
    for (int j = 1; j < 10; j++) {
        System.out.print(i * j + "\t");
    }
    System.out.println();
}
```

It outputs:

```
1   2   3   4   5   6   7   8   9
2   4   6   8   10  12  14  16  18
3   6   9   12  15  18  21  24  27
4   8   12  16  20  24  28  32  36
5   10  15  20  25  30  35  40  45
6   12  18  24  30  36  42  48  54
7   14  21  28  35  42  49  56  63
8   16  24  32  40  48  56  64  72
9   18  27  36  45  54  63  72  81
```

Report a typo

1179 users liked this theory. **26** didn't like it. **What about you?**

Start practicing

Comments (16) Hints (0) Useful links (0) Show discussion