Theory: Switch statement

26 minutes 5 / 13 problems solved

Start practicing

9704 users solved this topic. Latest completion was about 3 hours ago.

§1. When a conditional statement is not so good

Suppose you need to write a program that performs different actions depending on the value of a variable. For example, choosing an action from the menu of a game. To do that you can use a conditional statement with multiple branches as shown below.

```
int action = ...; // a certain value from 1 to 4

if (action == 1) {
    System.out.println("Starting a new game...");
} else if (action == 2) {
    System.out.println("Loading a saved game");
} else if (action == 3) {
    System.out.println("Displaying help...");
} else if (action == 4) {
    System.out.println("Exiting...");
} else {
    System.out.println("Unsuitable action, please, try again");
}
```

Of course, this code handles the task. But if your conditional statement has a lot of branches, it can be hard to understand for people.

§2. Three keywords: switch, case, and default

The switch statement provides a way to choose between multiple cases based on the value of a single variable (not an expression!). The variable can be an integer number, character, string, or enumeration. The last two types will be studied further.

Using the switch statement, the previous code will look like this:

```
switch (action) {
    case 1:
        System.out.println("Starting a new game...");
        break;
    case 2:
        System.out.println("Loading a saved game");
        break;
    case 3:
        System.out.println("Displaying help...");
        break;
    case 4:
        System.out.println("Exiting...");
        break;
    default:
        System.out.println("Unsuitable action, please, try again");
}
```

As you can see, this code is well-structured and easier to read than the equal conditional statement. We have not explained the keywords `switch`, `case` and `break` yet, but you can already guess what they mean.

Current topic:

✓ Switch statement (Stage 3)

Topic depends on:

✓ Branching statements (Stage 2)

Topic is required for:

✓ Functional decomposition (Stage 3)

Enums in Java

Table of contents:

↑ Switch statement

§1. When a conditional statement is not so good

§2. Three keywords: switch, case, and default

§3. The general form of the switch statement

§4. An example with "zero", "one" and "two"

§5. Conclusion

Feedback & Comments

§3. The general form of the switch statement

The most general form of the switch statement is the following

```
switch (variable) {
    case value1:
        // do something here
        break;
    case value2:
        // do something here
        break;

    //... other cases

    case valueN:
        // do something here
        break;
    default:
        // do something by default
        break; // it can be omitted
}
```

The `switch` and `case` keywords are always required here. The keywords `break` and `default` are optional. The keyword `break` stops the execution of the whole switch statement, not just one case.

If a `case` does not have the `break` keyword, the following `case` will be evaluated as well, including the `default` case. The `default` case is also evaluated if there's no other `case` that matches the variable value. The `break` keyword in the `default` branch is optional and can be omitted.

§4. An example with "zero", "one" and "two"

Let's consider another example. The following code outputs the names of integer numbers or a default text. This switch statement has three base cases and a single default case.

```
int val = ...;
switch (val) {
     case 0:
         System.out.println("zero");
         break;
     case 1:
         System.out.println("one");
         break;
     case 2:
         System.out.println("two");
         break;
     default:
 System.out.println("The value is less than zero or greater than two");
}
```

If the `val` is 0, the code prints:

```
zero
```

If the `val` is 1, the code prints:

```
one
```

if the `val` is 10, the code prints:

```
The value is less than zero or greater than two
```

If you forget the keyword `break` in a case, the compiler won't consider it an error. Let's remove it from the second case (case 1) and assign 1 to `val`. The program prints:

```
one
two
```

Omitting `break` keyword is not a good practice. Try to avoid it.

Java 12–14 introduced some new features allowing to use **switch** as an expression. You can read more about switch expressions, but keep in mind that for now our testing platform only supports Java 11.

§5. Conclusion

When you have a limited number of cases to choose from, switch statements can help you avoid unnecessary nested *if-else* constructions. For that, you need `switch` keyword to introduce the value to evaluate, and `case` for each of the possible values. Do not forget to also use the `break` keyword to avoid evaluating extra cases and `default` branch to indicate the default behavior.

Report a typo

788 users liked this theory. 13 didn't like it. **What about you?**

Start practicing

This content was created over 3 years ago and updated 5 days ago. Share your feedback below in comments to help us improve it!

Comments (41) Hints (0) Useful links (2) Show discussion