5.8 break and continue Statements

In addition to selection and iteration statements, Java provides statements break (which we discussed in the context of the switch statement) and continue (presented in this section and online Appendix L) to alter the flow of control. The preceding section showed how break can be used to terminate a switch statement's execution. This section discusses how to use break in iteration statements.

5.8.1 break Statement

The break statement, when executed in a while, for, do…while or switch, causes *immediate* exit from that statement. Execution continues with the first statement after the control statement. Common uses of break are to escape early from a loop or to skip the remainder of a switch (as in Fig. 5.9).

Figure 5.13 demonstrates a break statement exiting a for. When the if statement nested at lines 8–10 in the for statement detects that count is 5, the break statement at line 9 executes. This terminates the for statement, and the program proceeds to line 15 (immediately after the for statement), which displays a message indicating the value of the control variable when the loop terminated. The loop fully executes its body only four times instead of 10.

```
// Fig. 5.13: BreakTest.java
// break statement exiting a for statement.
public class BreakTest {
   public static void main(String[] args) {
      int count; // control variable also used af

      for (count = 1; count <= 10; count++) { //
         if (count == 5) {
            break; // terminates loop if count is
         }

         System.out.printf("%d ", count);
      }

      System.out.printf("%nBroke out of loop at c
   }
}
```

```
1 2 3 4
Broke out of loop at count = 5
```

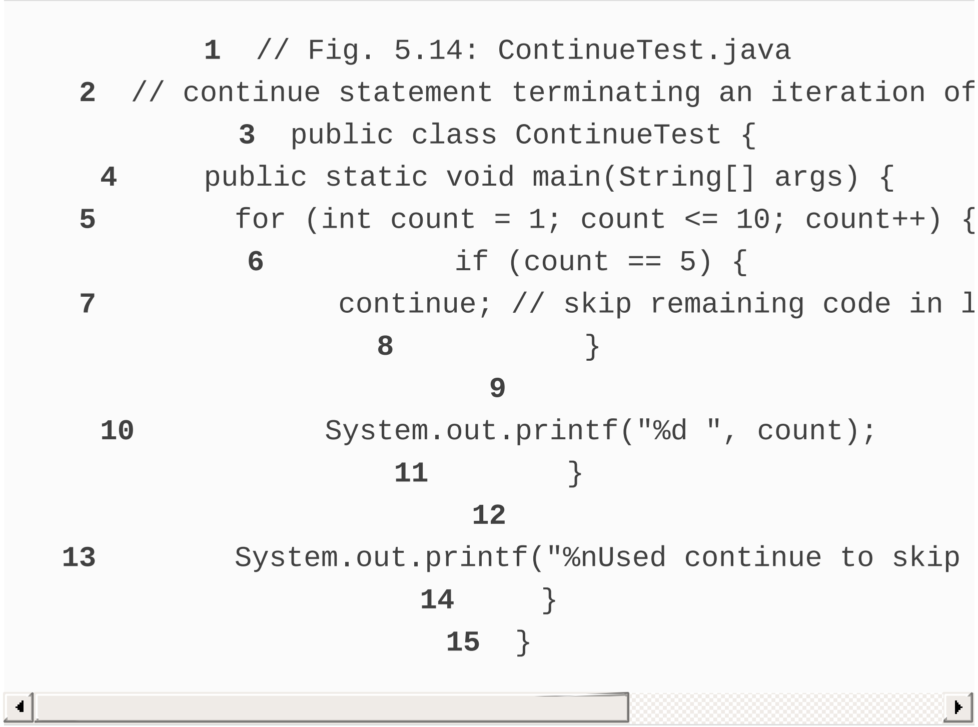
Fig. 5.13

break statement exiting a for statement.

5.8.2 continue Statement

The `continue` statement, when executed in a `while`, `for` or `do…while`, skips the remaining statements in the loop body and proceeds with the *next iteration* of the loop. In `while` and `do…while` statements, the program evaluates the loop-continuation test immediately after the `continue` statement executes. In a `for` statement, the increment expression executes, then the program evaluates the loop-continuation test.

Figure 5.14 uses `continue` (line 7) to skip the statement at line 10 when the nested `if` determines that `count`’s value is `5`. When the `continue` statement executes, program control continues with the increment of the control variable in the `for` statement (line 5).



```
1  // Fig. 5.14: ContinueTest.java
2  // continue statement terminating an iteration of
3  public class ContinueTest {
4     public static void main(String[] args) {
5        for (int count = 1; count <= 10; count++) {
6           if (count == 5) {
7              continue; // skip remaining code in l
8           }
9
10          System.out.printf("%d ", count);
11       }
12
13       System.out.printf("%nUsed continue to skip
14    }
15 }
```

```
1 2 3 4 6 7 8 9 10
Used continue to skip printing 5
```

Fig. 5.14

`continue` statement terminating an iteration of a `for` statement.

In Section 5.3, we stated that `while` could be used in most cases in place of `for`. This is *not* true when the increment expression in the `while` follows a `continue` statement. In this case, the increment does *not* execute before the program evaluates the iteration-continuation condition, so the `while` does not execute in the same manner as the `for`.

Software Engineering Observation 5.1

Some programmers feel that `break` *and* `continue` *violate structured programming. The same effects are achievable with structured-programming techniques, so these programmers do not use* `break` *or* `continue`.

Software Engineering Observation 5.2

There’s a tension between achieving quality software engineering and achieving the best-performing software. Sometimes one of these goals is achieved at the expense of the other. For all but the most performance-intensive situations, apply the following guideline: First, make your code simple and correct; then make it fast and small, but only if necessary.