can be read "If it's not true that 7 == 5," and the statement produces this output:

```
not equal
```

Here's another example using booleans:

```
boolean t = true;
boolean f = false;
System.out.println("! " + (t & !f) + " " + f);
```

It produces this output:

```
! true false
```

In the preceding example, notice that the `&` test succeeded (printing `true`) and that the value of the `boolean` variable `f` did not change, so it printed `false`.

CERTIFICATION SUMMARY

If you've studied this chapter diligently, you should have a firm grasp on Java operators, and you should understand what equality means when tested with the `==` operator. Let's review the highlights of what you've learned in this chapter.

The logical operators (`&&`, `||`, `&`, `|`, and `^`) can be used only to evaluate two `boolean` expressions. The difference between `&&` and `&` is that the `&&` operator won't bother testing the right operand if the left evaluates to `false`, because the result of the `&&` expression can never be `true`. The difference between `||` and `|` is that the `||` operator won't bother testing the right operand if the left evaluates to `true`, because the result is already known to be `true` at that point.

The `==` operator can be used to compare values of primitives, but it can also be used to determine whether two reference variables refer to the same object.

The `instanceof` operator is used to determine whether the object referred to by a reference variable passes the IS-A test for a specified type.

The `+` operator is overloaded to perform `String` concatenation tasks and can also concatenate `String`s and primitives, but be careful—concatenation can be tricky.

The conditional operator (a.k.a. the "ternary operator") has an unusual, three-operand syntax—don't mistake it for a complex assert statement.

The `++` and `--` operators will be used throughout the exam, and you must pay attention to whether they are prefixed or postfixed to the variable being updated.

Be prepared for a lot of exam questions involving the topics from this chapter. Even within questions testing your knowledge of another objective, the code will frequently use operators, assignments, object and primitive passing, and so on.

TWO-MINUTE DRILL

Here are some of the key points from each section in this chapter.

Relational Operators (OCA Objectives 3.1 and 3.3)

- ❑ Relational operators always result in a `boolean` value (`true` or `false`).
- ❑ There are six relational operators: `>`, `>=`, `<`, `<=`, `==`, and `!=`. The last two (`==` and `!=`) are sometimes referred to as *equality operators*.
- ❑ When comparing characters, Java uses the Unicode value of the character as the numerical value.
- ❑ Equality operators
  - ❑ There are two equality operators: `==` and `!=`.
  - ❑ Four types of things can be tested: numbers, characters, booleans, and reference variables.
- ❑ When comparing reference variables, `==` returns `true` only if both references refer to the same object.

instanceof Operator (OCA Objective 3.1)

- ❑ `instanceof` is for reference variables only; it checks whether the object is of a particular type.
- ❑ The `instanceof` operator can be used only to test objects (or `null`) against class types that are in the same class hierarchy.
- ❑ For interfaces, an object passes the `instanceof` test if any of its superclasses implement the interface on the right side of the `instanceof` operator.

Arithmetic Operators (OCA Objectives 3.1 and 3.2)

- ❑ The four primary math operators are add (`+`), subtract (`-`), multiply (`*`), and divide (`/`).
- ❑ The remainder (a.k.a. modulus) operator (`%`) returns the remainder of a division.
- ❑ Expressions are evaluated from left to right, unless you add parentheses, or unless some operators in the expression have higher precedence than others.
- ❑ The `*`, `/`, and `%` operators have higher precedence than `+` and `-`.