3.1 Introduction

problem

If you enter a negative value for `radius` in Listing 2.2, ComputeAreaWithConsoleInput.java, the program prints an invalid result. If the radius is negative, you don't want the program to compute the area. How can you deal with this situation?

Like all high-level programming languages, Java provides selection statements that let you choose actions with two or more alternative courses. You can use the following selection statement to replace lines 12–17 in Listing 2.2:

```
if (radius < 0)
  System.out.println("Incorrect input");
else {
  area = radius * radius * 3.14159;
  System.out.println("Area is " + area);
}
```

Selection statements use conditions. Conditions are Boolean expressions. This chapter first introduces Boolean types, values, comparison operators, and expressions.

3.2 `boolean` Data Type

comparison operators

How do you compare two values, such as whether a radius is greater than `0`, equal to `0`, or less than `0`? Java provides six *comparison operators* (also known as *relational operators*), shown in Table 3.1, which can be used to compare two values (assume radius is `5` in the table).

TABLE 3.1 Comparison Operators

Operator	*Name*	*Example*	*Result*
`<`	less than	`radius < 0`	`false`
`<=`	less than or equal to	`radius <= 0`	`false`
`>`	greater than	`radius > 0`	`true`
`>=`	greater than or equal to	`radius >= 0`	`true`
`==`	equal to	`radius == 0`	`false`
`!=`	not equal to	`radius != 0`	`true`

compare characters

Note

You can also compare characters. Comparing characters is the same as comparing their Unicodes. For example, `'a'` is larger than `'A'` because the Unicode of `'a'` is larger than the Unicode of `'A'`. See Appendix B, "The ASCII Character Sets," to find the order of characters.

== vs. =

Caution

The equality comparison operator is two equal signs (`==`), not a single equal sign (`=`). The latter symbol is for assignment.

The result of the comparison is a Boolean value: `true` or `false`. For example, the following statement displays `true`:

```
double radius = 1;
System.out.println(radius > 0);
```

Boolean variable

A variable that holds a Boolean value is known as a *Boolean variable*. The `boolean` data type is used to declare Boolean variables. A `boolean` variable can hold one of the two values:

3.4 `if` Statements

why `if` statement?

The preceding program displays a message such as "6 + 2 = 7 is false." If you wish the message to be "6 + 2 = 7 is incorrect," you have to use a selection statement to carry out this minor change.

This section introduces selection statements. Java has several types of selection statements: one-way `if` statements, two-way `if` statements, nested `if` statements, `switch` statements, and conditional expressions.

3.4.1 One-Way `if` Statements

A one-way `if` statement executes an action if and only if the condition is `true`. The syntax for a one-way `if` statement is shown below:

`if` statement

```
if (boolean-expression) {
  statement(s);
}
```

The execution flow chart is shown in Figure 3.1(a).

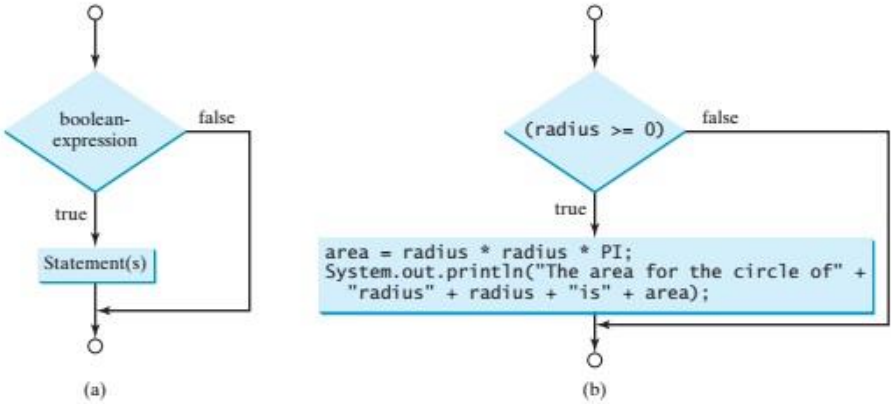

FIGURE 3.1 An `if` statement executes statements if the `boolean-expression` evaluates to `true`.

If the `boolean-expression` evaluates to `true`, the statements in the block are executed. As an example, see the following code:

```
if (radius >= 0) {
  area = radius * radius * PI;
  System.out.println("The area for the circle of radius " +
    radius + " is " + area);
}
```

The flow chart of the preceding statement is shown in Figure 3.1(b). If the value of `radius` is greater than or equal to `0`, then the `area` is computed and the result is displayed; otherwise, the two statements in the block will not be executed.

The `boolean-expression` is enclosed in parentheses. For example, the code in (a) below is wrong. It should be corrected, as shown in (b).

```
if i > 0 {
  System.out.println("i is positive");
}
```

(a) Wrong

```
if (i > 0) {
  System.out.println("i is positive");
}
```

(b) Correct

3.4 `if` Statements

why `if` statement?

The preceding program displays a message such as "6 + 2 = 7 is false." If you wish the message to be "6 + 2 = 7 is incorrect," you have to use a selection statement to carry out this minor change.

This section introduces selection statements. Java has several types of selection statements: one-way `if` statements, two-way `if` statements, nested `if` statements, `switch` statements, and conditional expressions.

3.4.1 One-Way `if` Statements

A one-way `if` statement executes an action if and only if the condition is `true`. The syntax for a one-way `if` statement is shown below:

`if` statement

```
if (boolean-expression) {
  statement(s);
}
```

The execution flow chart is shown in Figure 3.1(a).

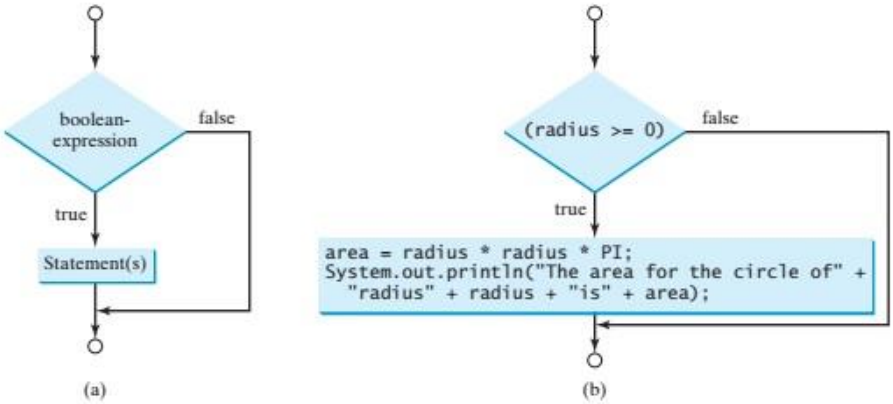

FIGURE 3.1 An `if` statement executes statements if the `boolean-expression` evaluates to `true`.

If the `boolean-expression` evaluates to `true`, the statements in the block are executed. As an example, see the following code:

```
if (radius >= 0) {
  area = radius * radius * PI;
  System.out.println("The area for the circle of radius " +
    radius + " is " + area);
}
```

The flow chart of the preceding statement is shown in Figure 3.1(b). If the value of `radius` is greater than or equal to `0`, then the `area` is computed and the result is displayed; otherwise, the two statements in the block will not be executed.

The `boolean-expression` is enclosed in parentheses. For example, the code in (a) below is wrong. It should be corrected, as shown in (b).

```
if i > 0 {
  System.out.println("i is positive");
}
```

(a) Wrong

```
if (i > 0) {
  System.out.println("i is positive");
}
```

(b) Correct

The block braces can be omitted if they enclose a single statement. For example, the following statements are equivalent.

```
if (i > 0) {
  System.out.println("i is positive");
}
```

(a)

Equivalent

```
if (i > 0)
  System.out.println("i is positive");
```

(b)

Listing 3.2 gives a program that prompts the user to enter an integer. If the number is a multiple of 5, print HiFive. If the number is divisible by 2, print HiEven.

LISTING 3.2 SimpleIfDemo.java

```
1 import java.util.Scanner;
2
3 public class SimpleIfDemo {
4   public static void main(String[] args) {
5     Scanner input = new Scanner(System.in);
6     System.out.println("Enter an integer: ");
7     int number = input.nextInt();                    enter input
8
9     if (number % 5 == 0)                             check 5
10      System.out.println("HiFive");
11
12    if (number % 2 == 0)                             check even
13      System.out.println("HiEven");
14  }
15 }
```

```
Enter an integer: 4 Enter
HiEven
```

```
Enter an integer: 30 Enter
HiFive
HiEven
```

The program prompts the user to enter an integer (line 7) and displays HiFive if it is divisible by 5 (lines 9–10) and HiEven if it is divisible by 2 (lines 12–13).

3.5 Problem: Guessing Birthdays

You can find out the date of the month when your friend was born by asking five questions. Each question asks whether the day is in one of the five sets of numbers.