2.3 Modifying Your First Java Program

In this section, we modify the example in Fig. 2.1 to print text on one line by using multiple statements and to print text on several lines by using a single statement.

Displaying a Single Line of Text with Multiple Statements

`Welcome to Java Programming!` can be displayed several ways. Class `Welcome2`, shown in Fig. 2.3, uses two statements (lines 7–8) to produce the output shown in Fig. 2.1. [*Note:* From this point forward, we highlight with a yellow background the new and key features in each code listing, as we've done for lines 7–8.]

```
1   // Fig. 2.3: Welcome2.java
2   // Printing a line of text with multiple stateme
3
4   public class Welcome2 {
5      // main method begins execution of Java appli
6      public static void main(String[] args) {
7         System.out.print("Welcome to");
```

```
    8          System.out.println("Java Programming!");
               9       } // end method main
              10    } // end class Welcome2
```

```
Welcome to Java Programming!
```

Fig. 2.3

Printing a line of text with multiple statements.

The program is similar to Fig. 2.1, so we discuss only the changes here. Line 2

```
// Printing a line of text with multiple statements.
```

is an end-of-line comment stating the purpose of the program. Line 4 begins the `Welcome2` class declaration. Lines 7–8 in method `main`

```
System.out.print("Welcome to");
System.out.println("Java Programming!");
```

display *one* line of text. The first statement uses `System.out`'s method `print` to display a string. Each `print` or `println` statement resumes displaying characters

from where the last `print` or `println` statement stopped displaying characters. Unlike `println`, after displaying its argument, `print` does *not* position the output cursor at the beginning of the next line—the next character the program displays will appear *immediately after* the last character that `print` displays. So, line 8 positions the first character in its argument (the letter "`J`") immediately after the last character that line 7 displays (the *space character* before the string's closing double-quote character).

Displaying Multiple Lines of Text with a Single Statement

A single statement can display multiple lines by using **newline characters** (`\n`), which indicate to `System.out`'s `print` and `println` methods when to position the output cursor at the beginning of the next line in the command window. Like blank lines, space characters and tab characters, newline characters are white space characters. The program in Fig. 2.4 outputs four lines of text, using newline characters to determine when to begin each new line. Most of the program is identical to those in Figs. 2.1 and 2.3.

```
1   // Fig. 2.4: Welcome3.java
2   // Printing multiple lines of text with a single
3
4   public class Welcome3 {
5      // main method begins execution of Java applic
6      public static void main(String[] args) {
```

```
   7          System.out.println("Welcome\nto\nJava\nProg
              8       } // end method main
              9    } // end class Welcome3
```

```
Welcome
to
Java
Programming!
```

Fig. 2.4

Printing multiple lines of text with a single statement.

Line 7

```
System.out.println("Welcome\nto\nJava\nProgramming!")
```

displays four lines of text in the command window. Normally, the characters in a string are displayed *exactly* as they appear in the double quotes. However, the paired characters `\` and `n` (repeated three times in the statement) do *not* appear on the screen. The **backslash** (\) is an **escape character**, which has special meaning to `System.out`'s `print` and `println` methods. When a backslash appears in a string, Java combines it with the next character to form an **escape sequence**—`\n` represents the newline character. When a newline character appears in a string being output with `System.out`, the

newline character causes the screen's output cursor to move to the beginning of the next line in the command window.

Figure 2.5 lists several escape sequences and describes how they affect the display of characters in the command window. For the complete list of escape sequences, visit

```
http://docs.oracle.com/javase/specs/jls/se8/html/jls-
```

Escape sequence	Description
`\n`	Newline. Position the screen cursor at the beginning of the *next* line.
`\t`	Horizontal tab. Move the screen cursor to the next tab stop.
`\r`	Carriage return. Position the screen cursor at the beginning of the *current* line—do *not* advance to the next line. Any characters output after the carriage return *overwrite* the characters previously output on that line.
`\\`	Backslash. Used to print a backslash character.
`\"`	Double quote. Used to print a double-quote character. For example, `System.out.println("\"in quotes\"");` displays `"in quotes"`.

Fig. 2.5

Some common escape sequences.