2.4 Displaying Text with printf

Method `System.out.printf` (`f` means “formatted”) displays *formatted* data. Figure 2.6 uses this to output on two lines the strings `"Welcome to"` and `"Java Programming!"`.

```
1   // Fig. 2.6: Welcome4.java
2   // Displaying multiple lines with method System.o
3
4   public class Welcome4 {
5      // main method begins execution of Java applic
6      public static void main(String[] args) {
7         System.out.printf("%s%n%s%n", "Welcome to",
8      } // end method main
9   } // end class Welcome4
```

```
Welcome to
Java Programming!
```

Fig. 2.6

Displaying multiple lines with method `System.out.printf`.

Line 7

```
   System.out.printf("%s%n%s%n", "Welcome to", "Java Pro
```

calls method `System.out.printf` to display the program's output. The method call specifies three arguments. When a method requires multiple arguments, they're placed in a **comma-separated list**. Calling a method is also referred to as **invoking** a method.

Good Programming Practice 2.7

Place a space after each comma (`,`) in an argument list to make programs more readable.

Method `printf`'s first argument is a **format string** that may consist of **fixed text** and **format specifiers**. Fixed text is output by `printf` just as it would be by `print` or `println`. Each format specifier is a *placeholder* for a value and specifies the *type of data* to output. Format specifiers also may include optional formatting information.

Format specifiers begin with a percent sign (%) followed by a character that represents the *data type*. For example, the format specifier `%s` is a placeholder for a string. The format string specifies that `printf` should output two strings, each

followed by a newline character. At the first format specifier's position, `printf` substitutes the value of the first argument after the format string. At each subsequent format specifier's position, `printf` substitutes the value of the next argument. So this example substitutes `"Welcome to"` for the first `%s` and `"Java Programming!"` for the second `%s`. The output shows that two lines of text are displayed on two lines.

Notice that instead of using the escape sequence `\n`, we used the `%n` format specifier, which is a line separator that's *portable* across operating systems. You cannot use `%n` in the argument to `System.out.print` or `System.out.println`; however, the line separator output by `System.out.println` *after* it displays its argument *is* portable across operating systems. Online Appendix I presents more details of formatting output with `printf`.