7.13 Variable-Length Argument Lists

With **variable-length argument lists**, you can create methods that receive an unspecified number of arguments. A type followed by an **ellipsis (...)** in a method's parameter list indicates that the method receives a variable number of arguments of that particular type. This use of the ellipsis can occur only *once* in a parameter list, and the ellipsis, together with its type and the parameter name, must be placed at the *end* of the parameter list. While you can use method overloading and array passing to accomplish much of what is accomplished with variable-length argument lists, using an ellipsis in a method's parameter list is more concise.

Figure 7.20 demonstrates method `average` (lines 6–15), which receives a variable-length sequence of `double`s. Java treats the variable-length argument list as an array whose elements are all of the same type. So, the method body can manipulate the parameter `numbers` as an array of `double`s. Lines 10–12 use the enhanced `for` loop to walk through the array and calculate the total of the `double`s in the array. Line 14 accesses `numbers.length` to obtain the size of the `numbers` array for use in the averaging calculation. Lines 27, 29 and 31 in `main` call method `average` with two, three and four arguments, respectively. Method `average` has a variable-length argument list (line 6), so it can average as

many `double` arguments as the caller passes. The output shows that each call to method `average` returns the correct value.

Common Programming Error 7.5

Placing an ellipsis indicating a variable-length argument list in the middle of a parameter list is a syntax error. An ellipsis may be placed only at the end of the parameter list.

```
1   // Fig. 7.20: VarargsTest.java
2   // Using variable-length argument lists.
3
4   public class VarargsTest {
5      // calculate average
6      public static double average(double… numbers)
7         double total = 0.0;
8
9         // calculate total using the enhanced for
10        for (double d : numbers) {
11           total += d;
12        }
13
14        return total / numbers.length;
15     }
16
17     public static void main(String[] args) {
18        double d1 = 10.0;
19        double d2 = 20.0;
20        double d3 = 30.0;
21        double d4 = 40.0;
22
```

```
23          System.out.printf("d1 = %.1f%nd2 = %.1f%nd
24             d1, d2, d3, d4);
25
26          System.out.printf("Average of d1 and d2 is
27             average(d1, d2));
28          System.out.printf("Average of d1, d2 and d
29             average(d1, d2, d3));
30          System.out.printf("Average of d1, d2, d3 a
31             average(d1, d2, d3, d4));
32       }
33    }
```

```
d1 = 10.0
d2 = 20.0
d3 = 30.0
d4 = 40.0

Average of d1 and d2 is 15.0
Average of d1, d2 and d3 is 20.0
Average of d1, d2, d3 and d4 is 25.0
```

Fig. 7.20

Using variable-length argument lists.