7.3 Declaring and Creating Arrays

Array objects occupy space in memory. Like other objects, arrays are created with keyword `new`. To create an array object, you specify the type of the array elements and the number of elements as part of an **array-creation expression** that uses keyword `new`. Such an expression returns a *reference* that can be stored in an array variable. The following declaration and array-creation expression create an array object containing 12 `int` elements and store the array's reference in the array variable named `c`:

```
int[] c = new int[12];
```

This expression can be used to create the array in Fig. 7.1. When an array is created, each of its elements receives a default value—zero for the numeric primitive-type elements, `false` for `boolean` elements and `null` for references. As you'll soon see, you can provide nondefault element values when you create an array.

Creating the array in Fig. 7.1 can also be performed in two steps as follows:

```
int[] c; // declare the array variable
```

```
c = new int[12]; // create the array; assign to array
```

In the declaration, the *square brackets* following the type indicate that `c` is a variable that will refer to an array (i.e., the variable will store an array *reference*). In the assignment statement, the array variable `c` receives the reference to a new array of 12 `int` elements.

Common Programming Error 7.2

In an array declaration, specifying the number of elements in the square brackets of the declaration (e.g., `int[12] c;`*) is a syntax error.*

A program can create several arrays in a single declaration. The following declaration reserves 100 elements for `b` and 27 elements for `x`:

```
String[] b = new String[100], x = new String[27];
```

When the type of the array and the square brackets are combined at the beginning of the declaration, all the identifiers in the declaration are array variables. In this case, variables `b` and `x` refer to `String` arrays. For readability, we prefer to declare only *one* variable per declaration. The preceding

declaration is equivalent to:

```
String[] b = new String[100]; // create array b
String[] x = new String[27]; // create array x
```

Good Programming Practice 7.1

For readability, declare only one variable per declaration. Keep each declaration on a separate line, and include a comment describing the variable being declared.

When only one variable is declared in each declaration, the square brackets can be placed either after the type or after the array variable name, as in:

```
String b[] = new String[100]; // create array b
String x[] = new String[27]; // create array x
```

but placing the square brackets after the type is preferred.

Common Programming Error 7.3

Declaring multiple array variables in a single declaration can lead to subtle errors. Consider the declaration `int[] a, b, c;`. *If* `a`, `b` *and* `c` *should be declared as array variables, then this declaration is correct—placing square brackets directly following the type indicates that* all *the identifiers in the declaration are array variables. However, if only* `a` *is intended to be an array variable, and* `b` *and* `c` *are intended to be individual* `int` *variables, then this declaration is incorrect —the declaration* `int a[], b, c;` *would achieve the desired result.*

A program can declare arrays of any type. Every element of a primitive-type array contains a value of the array's declared element type. Similarly, in an array of a reference type, every element is a reference to an object of the array's declared element type. For example, every element of an `int` array is an `int` value, and every element of a `String` array is a reference to a `String` object.