method is in a program with an array of objects, which leads us to the next topic.

**An Array of Objects**

Often, as with the **Stock** examples, you want to create several objects of the same class. You can create separate object variables, as the examples have done so far, but it might make more sense to create an array of objects. That might sound like a major leap into the unknown, but, in fact, you declare an array of objects the same way you would an array of any of the standard types:

```cpp
Stock mystuff[4]; // creates an array of 4 Stock objects
```

Recall that a program always calls the default class constructor when it creates class objects that aren't explicitly initialized. This declaration requires either that the class explicitly defines no constructors at all, in which case the implicit do-nothing default constructor is used, or, as in this case, that an explicit default constructor is defined. Each element—`mystuff[0]`, `mystuff[1]`, and so on—is a **Stock** object and thus can be used with the **Stock** methods:

```cpp
mystuff[0].update(); // apply update() to 1st element
mystuff[3].show();   // apply show() to 4th element
Stock tops = mystuff[2].topval(mystuff[1]);
                   // compare 3rd and 2nd elements
```

You can use a constructor to initialize the array elements. In that case, you have to call the constructor for each individual element:

```cpp
const int STKS = 4;
Stock stocks[STKS] = {
    Stock("NanoSmart", 12.5, 20),
    Stock("Boffo Objects", 200, 2.0),
    Stock("Monolithic Obelisks", 130, 3.25),
    Stock("Fleep Enterprises", 60, 6.5)
};
```

Here the code uses the standard form for initializing an array: a comma-separated list of values enclosed in braces. In this case, a call to the constructor method represents each