Exercise 4.58 Implement the **`numberInStock`** method. This should locate a product in the collection with a matching ID and return the current quantity of that product as a method result. If no product with a matching ID is found, return zero. This is relatively simple to implement once the **`findProduct`** method has been completed. For instance, **`numberInStock`** can call the **`findProduct`** method to do the searching and then call the **`getQuantity`** method on the result. Take care with products that cannot be found, though.

Exercise 4.59 Implement the **`delivery`** method using a similar approach to that used in **`numberInStock`**. It should find the product with the given ID in the list of products and then call its **`increaseQuantity`** method.

Exercise 4.60 *Challenge exercise* Implement a method in **`StockManager`** to print details of all products with stock levels below a given value (passed as a parameter to the method).

Modify the **`addProduct`** method so that a new product cannot be added to the product list with the same ID as an existing one.

Add to **`StockManager`** a method that finds a product from its name rather than its ID:

```
public Product findProduct(String name)
```

In order to do this, you need to know that two **`String`** objects, **`s1`** and **`s2`**, can be tested for equality with the boolean expression

```
s1.equals(s2)
```

More details about comparing `Strings` can be found in Chapter 6.

4.15 Summary

We have seen that classes such as **`ArrayList`** conveniently allow us to create collections containing an arbitrary number of objects. The Java library contains more collections like this, and we shall look at some of the others in Chapter 6. You will find that being able to use collections confidently is an important skill in writing interesting programs. There is hardly an application we shall see from now on that does not use collections of some form.

When using collections, the need arises to iterate over all elements in a collection to make use of the objects stored in them. For this purpose, we have seen the use of loops and iterators.

Loops are also a fundamental concept in computing that you will be using in every project from now on. Make sure you familiarize yourself sufficiently with writing loops—you will not get very far without them. When deciding on the type of loop to use in a particular situation, it is often useful to consider whether the task involves definite iteration, or indefinite iteration. Is there certainty or uncertainty about the number of iterations that will be required?

As an aside, we have mentioned the Java class library, a large collection of useful classes that we can use to make our own classes more powerful. We shall need to study the library in some more detail to see what else is in it that we should know about. This will be the topic of a future chapter.

Terms introduced in this chapter

collection, iterator, for-each loop, while loop, index, import statement, library, package, anonymous object, definite iteration, indefinite iteration

Concept summary

- **collection** A collection object can store an arbitrary number of other objects.
- **loop** A loop can be used to execute a block of statements repeatedly without having to write them multiple times.
- **iterator** An iterator is an object that provides functionality to iterate over all elements of a collection.
- **null** The Java reserved word **`null`** is used to mean "no object" when an object variable is not currently referring to a particular object. A field that has not explicitly been initialized will contain the value **`null`** by default.

Exercise 4.84 Java provides another type of loop: the *do-while loop*. Find out how this loop works and describe it. Write an example of a do-while loop that prints out the numbers from 1 to 10. To find out about this loop, find a description of the Java language (for example, at

`http://download.oracle.com/javase/tutorial/java/nutsandbolts/`

in the section "Control Flow Statements").

Exercise 4.85 Rewrite the **`listAllFiles`** method in the **`MusicOrganizer`** class from **`music-organizer-v3`** by using a do-while loop rather than a for-each loop. Test your solution carefully. Does it work correctly if the collection is empty?

Exercise 4.86 *Challenge exercise* Rewrite the **`findFirst`** method in the **`MusicOrganizer`** class from *music-organizer-v4* by using a do-while loop rather than a while loop. Test your solution carefully. Try searches that will succeed and others that will fail. Try searches where the file to be found is first in the list, and also where it is last in the list.

Exercise 4.87 Find out about Java's *switch-case statement*. What is its purpose? How is it used? Write an example. (This is also a *control flow statement*, so you will find information in similar locations as for the *do-while loop*.)