7.4 The Enhanced for Loop

Java version 5.0 introduces a very convenient shortcut for a common loop type. Often, you need to iterate through a sequence of elements—such as the elements of an array or array list. The enhanced for loop makes this process particularly easy to program.

Suppose you want to total up all data values in an array data. Here is how you use the enhanced for loop to carry out that task.

```java
double[] data = ...;
double sum = 0;
for (double e : data)
{
    sum = sum + e;
}
```

The loop body is executed for each element in the array data. At the beginning of each loop iteration, the next element is assigned to the variable e. Then the loop body is executed. You should read this loop as “for each e in data”.

You may wonder why Java doesn’t let you write “for each (e in data)”. Unquestionably, this would have been neater, and the Java language designers seriously considered this. However, the “for each” construct was added to Java several years after its initial release. Had new keywords each and in been added to the language, then older programs that happened to use those identifiers as variable or method names (such as System.in) would no longer have compiled correctly.

You don’t have to use the “for each” construct to loop through all elements in an array. You can implement the same loop with a straightforward for loop and an explicit index variable:

```java
double[] data = ...;
double sum = 0;
for (int i = 0; i < data.length; i++)
{
    double e = data[i];
    sum = sum + e;
}
```

Note an important difference between the “for each” loop and the ordinary for loop. In the “for each” loop, the element variable e is assigned values data[0], data[1], and so on. In the ordinary for loop, the index variable i is assigned values 0, 1, and so on.

You can also use the enhanced for loop to visit all elements of an array list. For example, the following loop computes the total value of all accounts:

```java
ArrayList<BankAccount> accounts = ...;
double sum = 0;
for (BankAccount a : accounts)
{
    sum = sum + a.getBalance();
}
```
This loop is equivalent to the following ordinary for loop:

```java
double sum = 0;
for (int i = 0; i < accounts.size(); i++)
{
    BankAccount a = accounts.get(i);
    sum = sum + a.getBalance();
}
```

The “for each” loop has a very specific purpose: traversing the elements of a collection, from the beginning to the end. Sometimes you don’t want to start at the beginning, or you may need to traverse the collection backwards. In those situations, do not hesitate to use an ordinary for loop.

**Syntax 7.3 The “for each” Loop**

```
for (Type variable : collection)
    statement
```

**Example:**
```
for (double e : data)
    sum = sum + e;
```

**Purpose:**
To execute a loop for each element in the collection. In each iteration, the variable is assigned the next element of the collection. Then the statement is executed.

### Self Check

7. Write a “for each” loop that prints all elements in the array data.

8. Why is the “for each” loop not an appropriate shortcut for the following ordinary for loop?
```
for (int i = 0; i < data.length; i++) data[i] = i * i;
```

### 7.5 Simple Array Algorithms

#### 7.5.1 Counting Matches

To count values in an array list, check all elements and count the matches until you reach the end of the array list.

Suppose you want to find how many accounts of a certain type you have. Then you must go through the entire collection and increment a counter each time you find a match. Here we count the number of accounts whose balance is at least as much as a given threshold:

```java
public class Bank {
    {
```