7. When an array is created, its elements are assigned the default value of `0` for the numeric primitive data types, `\u0000` for char types, and `false` for `boolean` types.

8. Java has a shorthand notation, known as the *array initializer*, which combines declaring an array, creating an array, and initializing an array in one statement, using the syntax `elementType[] arrayRefVar = {value0, value1, ..., valuek}`.

9. When you pass an array argument to a method, you are actually passing the reference of the array; that is, the called method can modify the elements in the caller's original array.

10. If an array is sorted, *binary search* is more efficient than *linear search* for finding an element in the array.

11. *Selection sort* finds the smallest number in the list and swaps it with the first element. It then finds the smallest number remaining and swaps it with the first element in the remaining list, and so on, until only a single number remains.

Quiz

Answer the quiz for this chapter online at www.cs.armstrong.edu/liang/intro10e/quiz.html.

MyProgrammingLab™

Programming Exercises

Sections 7.2–7.5

*7.1 (*Assign grades*) Write a program that reads student scores, gets the best score, and then assigns grades based on the following scheme:

Grade is A if score is ≥ best − 10

Grade is B if score is ≥ best − 20;

Grade is C if score is ≥ best − 30;

Grade is D if score is ≥ best − 40;

Grade is F otherwise.

The program prompts the user to enter the total number of students, then prompts the user to enter all of the scores, and concludes by displaying the grades. Here is a sample run:

```
Enter the number of students: 4 Enter
Enter 4 scores: 40 55 70 58 Enter
Student 0 score is 40 and grade is C
Student 1 score is 55 and grade is B
Student 2 score is 70 and grade is A
Student 3 score is 58 and grade is B
```

7.2 (*Reverse the numbers entered*) Write a program that reads ten integers and displays them in the reverse of the order in which they were read.

**7.3 (*Count occurrence of numbers*) Write a program that reads the integers between 1 and 100 and counts the occurrences of each. Assume the input ends with 0. Here is a sample run of the program:

```
Enter the integers between 1 and 100: 2 5 6 5 4 3 23 43 2 0 Enter
2 occurs 2 times
3 occurs 1 time
4 occurs 1 time
5 occurs 2 times
6 occurs 1 time
23 occurs 1 time
43 occurs 1 time
```

Note that if a number occurs more than one time, the plural word "times" is used in the output.

7.4 (*Analyze scores*) Write a program that reads an unspecified number of scores and determines how many scores are above or equal to the average and how many scores are below the average. Enter a negative number to signify the end of the input. Assume that the maximum number of scores is 100.

**7.5 (*Print distinct numbers*) Write a program that reads in ten numbers and displays the number of distinct numbers and the distinct numbers separated by exactly one space (i.e., if a number appears multiple times, it is displayed only once). (*Hint*: Read a number and store it to an array if it is new. If the number is already in the array, ignore it.) After the input, the array contains the distinct numbers. Here is the sample run of the program:

```
Enter ten numbers: 1 2 3 2 1 6 3 4 5 2 Enter
The number of distinct number is 6
The distinct numbers are: 1 2 3 6 4 5
```

*7.6 (*Revise Listing 5.15, PrimeNumber.java*) Listing 5.15 determines whether a number n is prime by checking whether 2, 3, 4, 5, 6, ..., n/2 is a divisor. If a divisor is found, n is not prime. A more efficient approach is to check whether any of the prime numbers less than or equal to $\sqrt{n}$ can divide n evenly. If not, n is prime. Rewrite Listing 5.15 to display the first 50 prime numbers using this approach. You need to use an array to store the prime numbers and later use them to check whether they are possible divisors for n.

*7.7 (*Count single digits*) Write a program that generates 100 random integers between 0 and 9 and displays the count for each number. (*Hint*: Use an array of ten integers, say `counts`, to store the counts for the number of 0s, 1s, ..., 9s.)

Sections 7.6–7.8

7.8 (*Average an array*) Write two overloaded methods that return the average of an array with the following headers:

```
public static int average(int[] array)
public static double average(double[] array)
```

Write a test program that prompts the user to enter ten double values, invokes this method, and displays the average value.

7.9 (*Find the smallest element*) Write a method that finds the smallest element in an array of double values using the following header:

```
public static double min(double[] array)
```

Write a test program that prompts the user to enter ten numbers, invokes this method to return the minimum value, and displays the minimum value. Here is a sample run of the program:

```
Enter ten numbers: 1.9 2.5 3.7 2 1.5 6 3 4 5 2 Enter
The minimum number is: 1.5
```

7.10 (*Find the index of the smallest element*) Write a method that returns the index of the smallest element in an array of integers. If the number of such elements is greater than 1, return the smallest index. Use the following header:

```
public static int indexOfSmallestElement(double[] array)
```

Write a test program that prompts the user to enter ten numbers, invokes this method to return the index of the smallest element, and displays the index.

***7.11** (*Statistics: compute deviation*) Programming Exercise 5.45 computes the standard deviation of numbers. This exercise uses a different but equivalent formula to compute the standard deviation of `n` numbers.

$$mean = \frac{\sum_{i=1}^{n} x_i}{n} = \frac{x_1 + x_2 + \cdots + x_n}{n} \quad deviation = \sqrt{\frac{\sum_{i=1}^{n} (x_i - mean)^2}{n - 1}}$$

To compute the standard deviation with this formula, you have to store the individual numbers using an array, so that they can be used after the mean is obtained.

Your program should contain the following methods:

```
/** Compute the deviation of double values */
public static double deviation(double[] x)

/** Compute the mean of an array of double values */
public static double mean(double[] x)
```

Write a test program that prompts the user to enter ten numbers and displays the mean and standard deviation, as shown in the following sample run:

```
Enter ten numbers: 1.9 2.5 3.7 2 1 6 3 4 5 2 Enter
The mean is 3.11
The standard deviation is 1.55738
```

***7.12** (*Reverse an array*) The `reverse` method in Section 7.7 reverses an array by copying it to a new array. Rewrite the method that reverses the array passed in the argument and returns this array. Write a test program that prompts the user to

enter ten numbers, invokes the method to reverse the numbers, and displays the numbers.

Section 7.9

***7.13** (*Random number chooser*) Write a method that returns a random number between 1 and 54, excluding the numbers passed in the argument. The method header is specified as follows:

```
public static int getRandom(int... numbers)
```

7.14 (*Computing gcd*) Write a method that returns the gcd of an unspecified number of integers. The method header is specified as follows:

```
public static int gcd(int... numbers)
```

Write a test program that prompts the user to enter five numbers, invokes the method to find the gcd of these numbers, and displays the gcd.

Sections 7.10–7.12

7.15 (*Eliminate duplicates*) Write a method that returns a new array by eliminating the duplicate values in the array using the following method header:

```
public static int[] eliminateDuplicates(int[] list)
```

Write a test program that reads in ten integers, invokes the method, and displays the result. Here is the sample run of the program:

```
Enter ten numbers: 1 2 3 2 1 6 3 4 5 2 Enter
The distinct numbers are: 1 2 3 6 4 5
```

7.16 (*Execution time*) Write a program that randomly generates an array of 100,000 integers and a key. Estimate the execution time of invoking the `linearSearch` method in Listing 7.6. Sort the array and estimate the execution time of invoking the `binarySearch` method in Listing 7.7. You can use the following code template to obtain the execution time:

```
long startTime = System.currentTimeMillis();
perform the task;
long endTime = System.currentTimeMillis();
long executionTime = endTime - startTime;
```

****7.17** (*Sort students*) Write a program that prompts the user to enter the number of students, the students' names, and their scores, and prints student names in decreasing order of their scores.

****7.18** (*Bubble sort*) Write a sort method that uses the bubble-sort algorithm. The bubble-sort algorithm makes several passes through the array. On each pass, successive neighboring pairs are compared. If a pair is not in order, its values are swapped; otherwise, the values remain unchanged. The technique is called a *bubble sort* or *sinking sort* because the smaller values gradually "bubble" their way to the top and the larger values "sink" to the bottom. Write a test program that reads in ten double numbers, invokes the method, and displays the sorted numbers.

**7.19 (*Sorted?*) Write the following method that returns true if the list is already sorted in increasing order.

```
public static boolean isSorted(int[] list)
```

Write a test program that prompts the user to enter a list and displays whether the list is sorted or not. Here is a sample run. Note that the first number in the input indicates the number of the elements in the list. This number is not part of the list.

```
Enter list: 8 10 1 5 16 61 9 11 1 Enter
The list is not sorted
```

```
Enter list: 10 1 1 3 4 4 5 7 9 11 21 Enter
The list is already sorted
```

*7.20 (*Revise selection sort*) In Section 7.11, you used selection sort to sort an array. The selection-sort method repeatedly finds the smallest number in the current array and swaps it with the first. Rewrite this program by finding the largest number and swapping it with the last. Write a test program that reads in ten double numbers, invokes the method, and displays the sorted numbers.

***7.21 (*Game: bean machine*) The bean machine, also known as a quincunx or the Galton box, is a device for statistics experiments named after English scientist Sir Francis Galton. It consists of an upright board with evenly spaced nails (or pegs) in a triangular form, as shown in Figure 7.13.

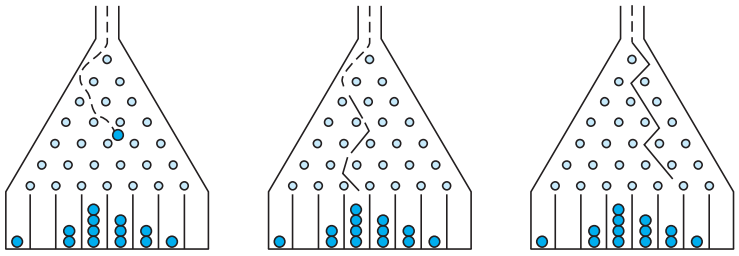

FIGURE 7.13 Each ball takes a random path and falls into a slot.

Balls are dropped from the opening of the board. Every time a ball hits a nail, it has a 50% chance of falling to the left or to the right. The piles of balls are accumulated in the slots at the bottom of the board.

Write a program that simulates the bean machine. Your program should prompt the user to enter the number of the balls and the number of the slots in the machine. Simulate the falling of each ball by printing its path. For example, the path for the ball in Figure 7.13b is LLRRLLR and the path for the ball in Figure 7.13c is

RLRRLRR. Display the final buildup of the balls in the slots in a histogram. Here is a sample run of the program:

```
Enter the number of balls to drop: 5 Enter
Enter the number of slots in the bean machine: 8 Enter

LRLRLRR
RRLLLRR
LLRLLRR
RRLLLLL
LRLRRLR

   0
   0
  000
```

(*Hint*: Create an array named `slots`. Each element in `slots` stores the number of balls in a slot. Each ball falls into a slot via a path. The number of Rs in a path is the position of the slot where the ball falls. For example, for the path LRLRLRR, the ball falls into `slots[4]`, and for the path is RRLLLLL, the ball falls into `slots[2]`.)

***7.22 (*Game: Eight Queens*) The classic Eight Queens puzzle is to place eight queens on a chessboard such that no two queens can attack each other (i.e., no two queens are on the same row, same column, or same diagonal). There are many possible solutions. Write a program that displays one such solution. A sample output is shown below:

```
|Q| | | | | | | |
| | | | |Q| | | |
| | | | | | | |Q|
| | | | | |Q| | |
| | |Q| | | | | |
| | | | | | |Q| |
| |Q| | | | | | |
| | | |Q| | | | |
```

**7.23 (*Game: locker puzzle*) A school has 100 lockers and 100 students. All lockers are closed on the first day of school. As the students enter, the first student, denoted S1, opens every locker. Then the second student, S2, begins with the second locker, denoted L2, and closes every other locker. Student S3 begins with the third locker and changes every third locker (closes it if it was open, and opens it if it was closed). Student S4 begins with locker L4 and changes every fourth locker. Student S5 starts with L5 and changes every fifth locker, and so on, until student S100 changes L100.

After all the students have passed through the building and changed the lockers, which lockers are open? Write a program to find your answer and display all open locker numbers separated by exactly one space.

(*Hint*: Use an array of 100 Boolean elements, each of which indicates whether a locker is open (`true`) or closed (`false`). Initially, all lockers are closed.)

VideoNote
Coupon collector's problem

**7.24 (*Simulation: coupon collector's problem*) Coupon collector is a classic statistics problem with many practical applications. The problem is to pick objects from a set of objects repeatedly and find out how many picks are needed for all the

objects to be picked at least once. A variation of the problem is to pick cards from a shuffled deck of 52 cards repeatedly and find out how many picks are needed before you see one of each suit. Assume a picked card is placed back in the deck before picking another. Write a program to simulate the number of picks needed to get four cards from each suit and display the four cards picked (it is possible a card may be picked twice). Here is a sample run of the program:

```
Queen of Spades
5 of Clubs
Queen of Hearts
4 of Diamonds
Number of picks: 12
```

7.25 (*Algebra: solve quadratic equations*) Write a method for solving a quadratic equation using the following header:

```
public static int solveQuadratic(double[] eqn, double[] roots)
```

The coefficients of a quadratic equation $ax^2 + bx + c = 0$ are passed to the array `eqn` and the real roots are stored in roots. The method returns the number of real roots. See Programming Exercise 3.1 on how to solve a quadratic equation.

Write a program that prompts the user to enter values for a, b, and c and displays the number of real roots and all real roots.

7.26 (*Strictly identical arrays*) The arrays `list1` and `list2` are *strictly identical* if their corresponding elements are equal. Write a method that returns `true` if `list1` and `list2` are strictly identical, using the following header:

```
public static boolean equals(int[] list1, int[] list2)
```

Write a test program that prompts the user to enter two lists of integers and displays whether the two are strictly identical. Here are the sample runs. Note that the first number in the input indicates the number of the elements in the list. This number is not part of the list.

```
Enter list1: 5 2 5 6 1 6 Enter
Enter list2: 5 2 5 6 1 6 Enter
Two lists are strictly identical
```

```
Enter list1: 5 2 5 6 6 1 Enter
Enter list2: 5 2 5 6 1 6 Enter
Two lists are not strictly identical
```

7.27 (*Identical arrays*) The arrays `list1` and `list2` are *identical* if they have the same contents. Write a method that returns `true` if `list1` and `list2` are identical, using the following header:

```
public static boolean equals(int[] list1, int[] list2)
```

Write a test program that prompts the user to enter two lists of integers and displays whether the two are identical. Here are the sample runs. Note that the first number in the input indicates the number of the elements in the list. This number is not part of the list.

```
Enter list1: 5 2 5 6 6 1 Enter
Enter list2: 5 5 2 6 1 6 Enter
Two lists are identical
```

```
Enter list1: 5 5 5 6 6 1 Enter
Enter list2: 5 2 5 6 1 6 Enter
Two lists are not identical
```

***7.28** (*Math: combinations*) Write a program that prompts the user to enter 10 integers and displays all combinations of picking two numbers from the 10.

***7.29** (*Game: pick four cards*) Write a program that picks four cards from a deck of 52 cards and computes their sum. An Ace, King, Queen, and Jack represent 1, 13, 12, and 11, respectively. Your program should display the number of picks that yields the sum of 24.

***7.30** (*Pattern recognition: consecutive four equal numbers*) Write the following method that tests whether the array has four consecutive numbers with the same value.

VideoNote
Consecutive four

```
public static boolean isConsecutiveFour(int[] values)
```

Write a test program that prompts the user to enter a series of integers and displays if the series contains four consecutive numbers with the same value. Your program should first prompt the user to enter the input size—i.e., the number of values in the series. Here are sample runs:

```
Enter the number of values: 8 Enter
Enter the values: 3 4 5 5 5 5 4 5 Enter
The list has consecutive fours
```

```
Enter the number of values: 9 Enter
Enter the values: 3 4 5 5 6 5 5 4 5 Enter
The list has no consecutive fours
```

****7.31** (*Merge two sorted lists*) Write the following method that merges two sorted lists into a new sorted list.

```
public static int[] merge(int[] list1, int[] list2)
```

Implement the method in a way that takes at most `list1.length` + `list2.length` comparisons. Write a test program that prompts the user to enter two sorted lists and displays the merged list. Here is a sample run. Note that the first number in the input indicates the number of the elements in the list. This number is not part of the list.

```
Enter list1: 5 1 5 16 61 111 Enter
Enter list2: 4 2 4 5 6 Enter
The merged list is 1 2 4 5 5 6 16 61 111
```

**7.32 (*Partition of a list*) Write the following method that partitions the list using the first element, called a *pivot*.

```
public static int partition(int[] list)
```

After the partition, the elements in the list are rearranged so that all the elements before the pivot are less than or equal to the pivot and the elements after the pivot are greater than the pivot. The method returns the index where the pivot is located in the new list. For example, suppose the list is {5, 2, 9, 3, 6, 8}. After the partition, the list becomes {3, 2, 5, 9, 6, 8}. Implement the method in a way that takes at most `list.length` comparisons. Write a test program that prompts the user to enter a list and displays the list after the partition. Here is a sample run. Note that the first number in the input indicates the number of the elements in the list. This number is not part of the list.

```
Enter list: 8 10 1 5 16 61 9 11 1 Enter
After the partition, the list is 9 1 5 1 10 61 11 16
```

*7.33 (*Culture: Chinese Zodiac*) Simplify Listing 3.9 using an array of strings to store the animal names.

**7.34 (*Sort characters in a string*) Write a method that returns a sorted string using the following header:

```
public static String sort(String s)
```

For example, `sort("acb")` returns `abc`.

Write a test program that prompts the user to enter a string and displays the sorted string.

***7.35 (*Game: hangman*) Write a hangman game that randomly generates a word and prompts the user to guess one letter at a time, as shown in the sample run. Each letter in the word is displayed as an asterisk. When the user makes a correct guess, the actual letter is then displayed. When the user finishes a word, display

the number of misses and ask the user whether to continue to play with another word. Declare an array to store words, as follows:

```
// Add any words you wish in this array
String[] words = {"write", "that", ...};
```

```
(Guess) Enter a letter in word ******* > p Enter
(Guess) Enter a letter in word p****** > r Enter
(Guess) Enter a letter in word pr**r** > p Enter
     p is already in the word
(Guess) Enter a letter in word pr**r** > o Enter
(Guess) Enter a letter in word pro*r** > g Enter
(Guess) Enter a letter in word progr** > n Enter
     n is not in the word
(Guess) Enter a letter in word progr** > m Enter
(Guess) Enter a letter in word progr*m > a Enter
The word is program. You missed 1 time
Do you want to guess another word? Enter y or n>
```