Chapter 7

Arrays

- 7.1 Array Basics
- 7.2 Arrays in Classes and Methods
- 7.3 Programming with Arrays and Classes
- 7.4 Sorting and Searching
- 7.5 Multidimensional Arrays
- 7.6 Graphics Supplement (Optional)

Objectives

- 1) Find out what an array is and how to use arrays in simple Java programs
- 2) Learn how to use array parameters and how to define methods that returns an array
- 3) learn the proper way to use an array as an instance variable in a class
- 4) introduce yourself to the topic of sorting an array.
- 5) become familiar with multidimensional arrays

7.1 ARRAY BASICS

- An array
  - » a single name for a collection of data values, all of **the [redacted] data type**
  - » [redacted] identifies precisely one of the values
- Array: more than a [redacted] type, less than an [redacted]
  - » their methods are invoked with a special subscript notation
    - – most programmers do not even think of them as methods
  - » they work **like** [redacted] when used as method arguments and return types
  - » they **do not have or use** [redacted]
  - » they are sort of like a Java class that is not fully implemented
- Arrays are a natural fit for loops, especially `for` loops

Creating Arrays

- General syntax for declaring an array:

```
Base_Type[] Array_Name = new Base_Type[Length];
```

- Examples:
  80-element array with base type `char`:
  `char[] symbol = new char[80];`

  100-element array of `double`s:
  `double[] reading = new double[100];`

  80-element array of `Species`:
  `Species[] specimen = new Species[80];`

Accessing Arrays

- 80-element array of `Species`:

```
Species[] specimen = new Species[80];
```

- The numbering starts with 0, not 1 (indexing)
  - » Specimen[0], specimen[1], specimen[2]…

- [redacted] elements
  - » subscripted variables.. Specimen[1]…
- Index (subscript)
  - » the integer expression within the square brackets.

Listing 7.1 An Array of Temperatures

```
/**
 Reads 7 temperatures from the user and shows which are above
 and which are below the average of the 7 temperatures.
*/
import java.util.Scanner;

public class ArrayOfTemperatures
{
   public static void main(String[] args)
   {
      double[] temperature = new double[7];

      // Read temperatures and compute their average:
      Scanner keyboard = new Scanner(System.in);
      System.out.println("Enter 7 temperatures:");
      double sum = 0;
      for (int index = 0; index < 7; index++)
      {
                  temperature[index] = keyboard.nextDouble( );
         sum = sum + temperature[index];
      }
```

```
        double average = sum / 7;
         System.out.println("The average temperature is " + average);

                        // Display each temperature and its relation to the average:
         System.out.println("The temperatures are");
         for (int index = 0; index < 7; index++)
         {
            if (temperature[index] < average)
               System.out.println(temperature[index] +
                                  " below average");
            else if (temperature[index] > average)
               System.out.println(temperature[index] +
                                  " above average");
            else //temperature[index] == average
               System.out.println(temperature[index] +
                                  " the average");
         }
         System.out.println("Have a nice week.");
      }
}
```

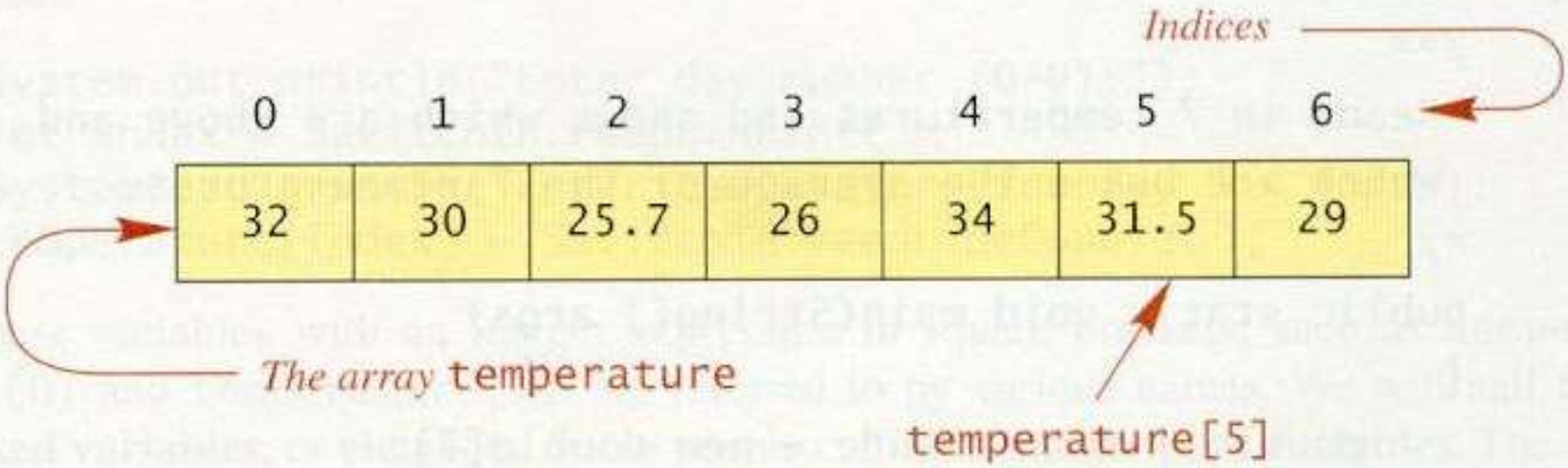
C:₩WINDOWS₩system32₩cmd.exe

```
Enter 7 temperatures:
32
30
25.7
26
34
31.5
29
The average temperature is 29.7428571428571 4
The temperatures are
32.0 above average.
30.0 above average.
25.7 below average.
26.0 below average.
34.0 above average.
31.5 above average.
29.0 below average.
Have a nice week.
계속하려면 아무 키나 누르십시오 . . .
```

Figure 7.1 A Common way to Visualize an Array

DISPLAY 6.1 An Array Used in a Program *(Part 2 of 2)*

A Common Way to Visualize an Array:



Three Ways to Use [ ] (Brackets) with an Array Name

1. To create a type name, e.g. `int[] pressure;` creates a name with the type "`int` array"

2. To create a new array, e.g. `pressure = new int[100];`

3. To name a specific element in the array
   - also called [redacted], e.g.

```
pressure[3] = keyboard.nextInt( );
System.out.println("You entered" + pressure[3]);
```

Some Array Terminology(Fig 7.2)

```
temperature[n + 2]
```

Array name

```
temperature[n + 2]
```

also called a *subscript*
- must be an `int`,
- or an expression that evaluates to an `int`

```
temperature[n + 2]
```

- also called an *element* or *subscripted variable*

```
temperature[n + 2] = 32;
```

Value of the indexed variable
- also called an element of the array

Note that "element" may refer to either a single indexed variable in the array or the *value* of a single indexed variable.

Programming Tip: Use Singular Array Names

- Using [redacted]**rather than plural names** for arrays improves readability
  - » Species[] entries = new Species[20]; // valid but not nice
  - » Species[] entry = new Species[20]; // nicer

- Although the array contains many elements the most common use of the name will be with a subscript, which references a [redacted]value.
  - » ex) System.out.println("The entry is " + entry[2]);
  - » ex) System.out.println("The entry is " + entries[2]);

Array Length– The length instance variable

- Length of an array is specified by the number in brackets when it is created with `new`
  - » it determines **the amount of memory** allocated for the array elements (values)
  - » it determines **the *maximum* number of elements** the array can hold
    - – storage is allocated whether or not the elements are assigned values
- The array length can be read with the instance variable `length`, e.g. the following code displays the number 20 (the *size*, or *length* of the `Species` array, `entry`):

```
Species[] entry = new Species[20];
System.out.println(entry.length);
```

- The `length` attribute is established in the declaration and **cannot be changed** unless the array is redeclared
  - » entry.length = 10 ; // illegal!!!

Listing 7.2 The Length Instance Variable - ArrayOfTemperature2.java

```
/** (7 ➔ temperature.length)
 Reads temperatures from the user and shows which are above
 and which are below the average of all the temperatures.
*/
import java.util.Scanner;

public class ArrayOfTemperatures2
{
   public static void main(String[] args)
   {
      Scanner keyboard = new Scanner(System.in);
         System.out.println("How many temperatures do you have?");
                  int size = keyboard.nextInt( );
                  double[] temperature = new double[size];

      // Read temperatures and compute their average:
      System.out.println("Enter " + temperature.length + " temperatures:");
                  double sum = 0;
```

```
for (int index = 0; index < temperature.length; index++)
      {
                                    temperature[index] = keyboard.nextDouble( );
         sum = sum + temperature[index];
      }

      double average = sum / temperature.length;
      System.out.println("The average temperature is " + average);

                        // Display each temperature and its relation to the average:
      System.out.println("The temperatures are");
      for (int index = 0; index < temperature.length; index++)
      {
         if (temperature[index] < average)
            System.out.println(temperature[index] +
                              " below average");
         else if (temperature[index] > average)
            System.out.println(temperature[index] +
                              " above average");
         else //temperature[index] == average
            System.out.println(temperature[index] +
                              " the average");
      }
      System.out.println("Have a nice week.");
   }
}
```

Subscript Range

- Array subscripts use zero-numbering
  - » the first element has subscript 0
  - » the second element has subscript 1
  - » etc. - the nth element has subscript n-1
  - » the last element has subscript [redacted]

- For example:

```
int[] scores = {97, 86, 92, 71};
```

Subscript:	0	1	2	3
Value:	97	86	92	71

Use for loop to step through an Array

```
for (index = 0; index < temperature.length; index++)
{
   temperature[index] = keyboard.nextDouble( )
   sum = sum + temperature[index];
}
```

Subscript out of Range Error

```
import java.util.Scanner;
public class ArrayOutOfBounds
{
   public static void main(String[] args)
   {
      Scanner keyboard = new Scanner(System.in);
       System.out.println("Enter a list of nonnegative integers.");
       System.out.println("Place a negative integer at the end.");

       int[] a = new int[10];

       int number = keyboard.nextInt();
       int i = 0;
       while (number >=0)
       {
                                a[i] = number;
                                i++;
                                number = keyboard.nextInt();
       }
   }
} //????
```

```
C:₩WINDOWS₩system32₩cmd.exe
Enter a list of nonnegative integers.
Place a negative integer at the end.
1
2
3
4
5
6
7
8
9
10
11

계속하려면 아무 키나 누르십시오 . . .
```

Subscript out of Range Error

- Using a subscript larger than `length-1` causes **a [redacted] (not a compiler) error**
  - » an [redacted] is thrown
    - – you do not need to catch it or declare it in a throws-clause
    - – you need to fix the problem and recompile your code

- Other programming languages, e.g. C and C++, do not even cause a run time error!
  - » one of the most dangerous characteristics of these languages is that they allow out of bounds array indexes.

```
import java.util.Scanner;
public class ArrayOutOfBoundsMody
{
   public static void main(String[] args)
   {
                    Scanner keyboard = new Scanner(System.in);
      System.out.println("Enter a list of nonnegative integers.");
      System.out.println("Place a negative integer at the end.");
      int[] a = new int[10];

      int number = keyboard.nextInt();
      int i = 0;
      while (              && (number >=0))
      {
                           a[i] = number;
                           i++;
                           number = keyboard.nextInt();
      }
      if (number >= 0)
      {
         System.out.println("Could not read in all the numbers.");
         System.out.println("Only read in" + a.length + " numbers.");
      }
   }
}
```

C:₩WINDOWS₩system32₩cmd.exe

```
Enter a list of nonnegative integers.
Place a negative integer at the end.
1
2
3
4
5
6
7
8
9
10
11
Could not read in all the numbers.
Only read in10 numbers.
계속하려면 아무 키나 누르십시오 . . .
```

Initializing an Array's Values in Its Declaration

- Array elements can be initialized in the declaration statement by putting a comma-separated list in braces
- Uninitialized elements will be assigned some default value, e.g. 0 for `int` arrays
- The length of an array is automatically determined when the values are explicitly initialized in the declaration
- For example:

```
double[] reading = {5.1, 3.02, 9.65};
System.out.println(reading.length);
```

  - displays 3, the length of the array `readings`

Initializing Array Elements in a Loop

- Array processing is easily done in a loop
- A `for` loop is commonly used **to initialize array elements**
- For example:

```
int i;//loop counter/array index
int[] a = new int[10];
for(i = 0; i < a.length; i++)
   a[i] = 0;
```

  - » note that the loop counter/array index goes from `0` to `length - 1`
  - » it counts through `length = 10` iterations/elements using the zero-numbering of the array index

Programming Tip:

Do not count on **default initial values** for array elements

- » explicitly initialize elements in the declaration or in a loop

끝