Chapter 11

How to work with arrays

Objectives

Applied

1. Write code that creates a one-dimensional array that stores a list of values or objects.
2. Use for loops and enhanced for loops to work with the values or objects in an array.
3. Use the methods of the Arrays class to fill an array, compare two arrays, sort an array, or search an array for a value.
4. Implement the Comparable interface in any class you create.
5. Create a reference to an array and copy elements from one array to another.
6. Write code that creates a two-dimensional array that stores a table of values or objects. The array can be either rectangular or jagged.
7. Use for loops and enhanced for loops to work with the values or objects in a two-dimensional array.

Objectives (cont.)

8. Given the Java code for an application that uses any of the language elements presented in this chapter, explain what each statement in the application does.

Knowledge

1. In general, explain what an array is and how you work with it.
2. Describe the operation of the enhanced for loop and explain why it's especially useful with arrays.
3. Explain when you need to implement the Comparable interface in a class you create.
4. Explain what happens when you assign a new array to an existing array variable.
5. Describe the difference between a rectangular array and a jagged array, and explain the difference in how you create them.

The syntax for declaring and instantiating an array

Two ways to declare an array

```
type[] arrayName;
type arrayName[];
```

How to instantiate an array

```
arrayName = new type[length];
```

How to declare and instantiate an array in one statement

```
type[] arrayName = new type[length];
```

Examples of array declarations

Code that declares an array of doubles

```
double[] prices;
```

Code that instantiates an array of doubles

```
prices = new double[4];
```

Code that declares and instantiates an array of doubles

```
double[] prices = new double[4];
```

Other examples of arrays

An array of String objects

```
String[] titles = new String[3];
```

An array of Product objects

```
Product[] products = new Product[5];
```

Code that uses a constant to specify the array length

```
final int TITLE_COUNT = 100;
String[] titles = new String[TITLE_COUNT];
```

Code that uses a variable to specify the array length

```
Scanner sc = new Scanner(System.in);
int titleCount = sc.nextInt();
String[] titles = new String[titleCount];
```

The syntax for referring to an element of an array

```
arrayName[index]
```

Code that assigns values to an array of doubles

```
double[] prices = new double[4];
prices[0] = 14.95;
prices[1] = 12.95;
prices[2] = 11.95;
prices[3] = 9.95;
//prices[4] = 8.95;  // this would throw
ArrayIndexOutOfBoundsException
```

Code that assigns values to an array of strings

```
String[] names = new String[3];
names[0] = "Ted Lewis";
names[1] = "Sue Jones";
names[2] = "Ray Thomas";
```

Code that assigns objects to an array of Product objects

```
Product[] products = new Product[2];
products[0] = new Product("java");
products[1] = new Product("jsps");
```

The syntax for creating an array and assigning values in one statement

```
type[] arrayName = {value1, value2, value3, ...};
```

Examples that create an array and assign values in one statement

```
double[] prices = {14.95, 12.95, 11.95, 9.95};

String[] names = {"Ted Lewis", "Sue Jones",
                  "Ray Thomas"};

Product[] products = {new Product("java"),
                      new Product("jsps")};
```

The syntax for getting the length of an array

```
arrayName.length
```

Code that assigns number values to an array

```
int[] values = new int[10];
for (int i = 0; i < values.length; i++) {
   values[i] = i;
}
```

Code that prints an array of prices to the console

```
double[] prices = {14.95, 12.95, 11.95, 9.95};
for (int i = 0; i < prices.length; i++) {
   System.out.println(prices[i]);
}
```

The console output

```
14.95
12.95
11.95
9.95
```

Code that computes the average of the array of prices

```
double sum = 0.0;
for (int i = 0; i < prices.length; i++) {
   sum += prices[i];
}
double average = sum / prices.length;
```

The syntax of the enhanced for loop

```
for (type variableName : arrayName) {
    statements
}
```

Code that prints an array of prices to the console

```
double[] prices = {14.95, 12.95, 11.95, 9.95};
for (double price : prices) {
   System.out.println(price);
}
```

The console output

```
14.95
12.95
11.95
9.95
```

Code that computes the average of the array of prices

```
double sum = 0.0;
for (double price : prices) {
   sum += price;
}
double average = sum / prices.length;
```

The Arrays class

```
java.util.Arrays
```

Some static methods of the class

```
fill(array, value)
sort(array)
binarySearch(array, value)
```

How to fill an array

```
int[] quantities = new int[5];   // all elements set to 0
Arrays.fill(quantities, 1);      // all elements set to 1
```

How to sort an array

```
int[] numbers = {2,6,4,1,8,5,9,3,7,0};
Arrays.sort(numbers);
for (int number : numbers) {
    System.out.print(number + " ");
}
```

The console output

```
0 1 2 3 4 5 6 7 8 9
```

How to search an array

```
String[] productCodes = {"mysql", "jsp", "java"};
Arrays.sort(productCodes);
int index = Arrays.binarySearch(productCodes, "mysql");
// sets index to 2
```

More static methods of the Arrays class

```
copyOf(array, length)
copyOfRange(array, index1, index2)
equals(array1, array2)
```

How to create a reference to an array

```
double[] grades = {92.3, 88.0, 95.2, 90.5};
double[] percentages = grades;
percentages[1] = 70.2;       // changes grades[1] too
System.out.println("grades[1]=" + grades[1]);  // 70.2
```

How to create a shallow copy (Java 6 or later)

```
double[] grades = {92.3, 88.0, 95.2, 90.5};
double[] percentages = Arrays.copyOf(
    grades, grades.length);
percentages[1] = 70.2;       // doesn't change grades[1]
System.out.println("grades[1]=" + grades[1]);  // 88.0
```

How to determine if two variables refer to the same array

```
if (grades == percentages) {
    System.out.println(
        "Both variables refer to the same array.");
} else {
    System.out.println(
        "Each variable refers to a different array.");
    System.out.println(
        "But these arrays may contain the same data.");
}
```

How to determine if two variables contain the same data

```
if (Arrays.equals(grades, percentages)) {
    System.out.println(
        "Both variables contain the same data.");
} else {
    System.out.println(
        "Both variables do not contain the same data.");
}
```

The Comparable interface defined in the Java API

```
public interface Comparable {
    int compareTo(Object obj);
}
```

An Item class that implements the Comparable interface

```
public class Item implements Comparable {
    private int number;
    private String description;

    public Item(int number, String description) {
        this.number = number;
        this.description = description;
    }

    public int getNumber() {
        return number;
    }

    public String getDescription() {
        return description;
    }
```

An Item class that implements the Comparable interface (cont.)

```
        @Override
        public int compareTo(Object o) {
            Item i = (Item) o;
            if (this.getNumber() < i.getNumber()) {
                return -1;
            }
            if (this.getNumber() > i.getNumber()) {
                return 1;
            }
            return 0;
        }
    }
```

Code that sorts an array of Item objects

```
Item[] items = new Item[3];
items[0] = new Item(102, "Duct Tape");
items[1] = new Item(103, "Bailing Wire");
items[2] = new Item(101, "Chewing Gum");
Arrays.sort(items);
for (Item i : items) {
    System.out.println(i.getNumber() + ": " +
    i.getDescription());
}
```

The console output

```
101: Chewing Gum
102: Duct Tape
103: Bailing Wire
```

The console for the Number Cruncher application

```
Numbers: 16 17 17 19 21 30 31 37 39 42 46
Total: 315
Count: 11
Average: 28.6
Median: 30
```

The code for the Number Cruncher application

```
import java.util.Arrays;

public class NumberCruncherApp {

    public static void main(String[] args) {
        // create array of 11 random integers
        int[] numbers = new int[11];
        for (int i = 0; i < numbers.length; i++) {
            numbers[i] = (int) (Math.random() * 51);
                                 // number is >= 0 and <= 50
        }

        // sort the array
        Arrays.sort(numbers);

        // display numbers
        String numbersString = "";
        for (int number : numbers) {
            numbersString += number + " ";
        }
        System.out.println("Numbers: " + numbersString);
```

The Number Cruncher application (cont.)

```
        // calculate total and display
        int total = 0;
        for (int number : numbers) {
            total += number;
        }
        System.out.println("Total: " + total);

        // get count of numbers and display
        int count = numbers.length;
        System.out.println("Count: " + count);

        // calculate average and display
        double average = (double) total / count;
        average = (double) Math.round(average * 10) / 10;
        System.out.println("Average: " + average);

        // if count of numbers is odd
        if (count % 2 != 0) {
            int medianIndex = count / 2;
            int median = numbers[medianIndex];
            System.out.println("Median: " + median);
        }
    }
}
```

How to create a rectangular array

The syntax for creating a rectangular array

```
type[][] arrayName = new type[rowCount][columnCount];
```

A statement that creates a 3x2 array

```
int[][] numbers = new int[3][2];
```

How to assign values to a rectangular array

The syntax for referring to an element of a rectangular array

```
arrayName[rowIndex][columnIndex]
```

The indexes for a 3x2 array

```
[0][0]           [0][1]
[1][0]           [1][1]
[2][0]           [2][1]
```

Code that assigns values to the array

```
numbers[0][0] = 1;
numbers[0][1] = 2;
numbers[1][0] = 3;
numbers[1][1] = 4;
numbers[2][0] = 5;
numbers[2][1] = 6;
```

A statement that creates and initializes the array

```
int[][] numbers = { {1,2}, {3,4}, {5,6} };
```

How to use nested for loops to process a rectangular array

```
int[][] numbers = { {1,2}, {3,4}, {5,6} };
for (int i = 0; i < numbers.length; i++) {
    for (int j = 0; j < numbers[i].length; j++)
        System.out.print(numbers[i][j] + "  ");
    System.out.print("\n");
}
```

The console output

```
1  2
3  4
5  6
```

The syntax for creating a jagged array

```
type[][] arrayName = new type[rowCount][];
```

Code that creates a jagged array of integers

```
int[][] numbers = new int[3][];
numbers[0] = new int[10];
numbers[1] = new int[15];
numbers[2] = new int[20];
```

Code that creates and initializes a jagged array of strings

```
String[][] titles =
   {{"War and Peace", "Wuthering Heights", "1984"},
    {"Casablanca", "Wizard of Oz", "Star Wars", "Birdy"},
    {"Blue Suede Shoes", "Yellow Submarine"}};
```

Code that creates and initializes a jagged array of integers

```
int number = 0;
int[][] pyramid = new int[4][];
for (int i = 0; i < pyramid.length; i++) {
    pyramid[i] = new int[i+1];
    for (int j = 0; j < pyramid[i].length; j++)
        pyramid[i][j] = number++;
}
```

Code that prints the contents of the jagged array of integers

```
for (int i = 0; i < pyramid.length; i++) {
   for (int j = 0; j < pyramid[i].length; j++)
      System.out.print(pyramid[i][j] + " ");
   System.out.print("\n");
}
```

The console output

```
0
1 2
3 4 5
6 7 8 9
```

Code that uses foreach loops to print a jagged array

```
for (int[] row : pyramid) {
    for (int col : row)
        System.out.print(col + " ");
    System.out.print("\n");
}
```