Chapter 7

Arrays
Arrays

• Arrays are objects that help us organize large amounts of information

• Chapter 7 focuses on:
  - array declaration and use
  - bounds checking and capacity
  - arrays that store object references
  - variable length parameter lists
  - multidimensional arrays
  - the ArrayList class
  - polygons and polylines
  - mouse events and keyboard events
Outline

Declaring and Using Arrays
Arrays of Objects
Variable Length Parameter Lists
Two-Dimensional Arrays
The ArrayList Class
Polygons and Polylines
Mouse Events and Key Events
Arrays

- **An array** is an ordered list of values

  The entire array has a single name
  Each value has a numeric *index*

<table>
<thead>
<tr>
<th>scores</th>
</tr>
</thead>
<tbody>
<tr>
<td>0  1  2  3  4  5  6  7  8  9</td>
</tr>
<tr>
<td>79 87 94 82 67 98 87 81 74 91</td>
</tr>
</tbody>
</table>

  An array of size N is indexed from zero to N-1

  This array holds 10 values that are indexed from 0 to 9
Arrays

• A particular value in an array is referenced using the array name followed by the index in brackets.

• For example, the expression

  \[
  \text{scores}[2]
  \]

  refers to the value 94 (the 3rd value in the array).

• That expression represents a place to store a single integer and can be used wherever an integer variable can be used.
Arrays

• For example, an array element can be assigned a value, printed, or used in a calculation:

```java
scores[2] = 89;

scores[first] = scores[first] + 2;

mean = (scores[0] + scores[1])/2;

System.out.println ("Top = " + scores[5]);
```
Arrays

- The values held in an array are called *array elements*.
- An array stores multiple values of the same type – the *element type*.
- The element type can be a primitive type or an object reference.
- Therefore, we can create an array of integers, an array of *String* objects, an array of *Coin* objects, etc.
- In Java, the array itself is an object that must be instantiated.
Arrays

• Another way to depict the scores array:
Declaring Arrays

• The scores array could be declared as follows:

```java
int[] scores = new int[10];
```

• The type of the variable scores is int[] (an array of integers)

• Note that the array type does not specify its size, but each object of that type has a specific size

• The reference variable scores is set to a new array object that can hold 10 integers
Declaring Arrays

• Some other examples of array declarations:

```java
float[] prices = new float[500];

boolean[] flags;
flags = new boolean[20];

char[] codes = new char[1750];
```
Using Arrays

• The iterator version of the `for` loop can be used when processing array elements

```java
    for (int score : scores)
        System.out.println (score);
```

• This is only appropriate when processing all array elements from top (lowest index) to bottom (highest index)

• See `BasicArray.java` (page 372)
Bounds Checking

• Once an array is created, it has a fixed size
• An index used in an array reference must specify a valid element
  • That is, the index value must be in range 0 to N-1
• The Java interpreter throws an ArrayIndexOutOfBoundsException if an array index is out of bounds
• This is called automatic bounds checking
Bounds Checking

- For example, if the array `codes` can hold 100 values, it can be indexed using only the numbers 0 to 99

- If the value of `count` is 100, then the following reference will cause an exception to be thrown:
  
  ```
  System.out.println (codes[count]);
  ```

- It’s common to introduce off-by-one errors when using arrays

  ```
  for (int index=0; index <= 100; index++)
      codes[index] = index*50 + epsilon;
  ```
Bounds Checking

- Each array object has a public constant called `length` that stores the size of the array.
- It is referenced using the array name:
  ```java
  scores.length
  ```
- Note that `length` holds the number of elements, not the largest index.
- See [ReverseOrder.java](#) (page 375)
- See [LetterCount.java](#) (page 376)
Alternate Array Syntax

• The brackets of the array type can be associated with the element type or with the name of the array.

• Therefore the following two declarations are equivalent:

  ```
  float[] prices;
  float prices[];
  ```

• The first format generally is more readable and should be used.
Initializer Lists

• An *initializer list* can be used to instantiate and fill an array in one step

• The values are delimited by braces and separated by commas

• Examples:

```java
int[] units = {147, 323, 89, 933, 540, 269, 97, 114, 298, 476};

char[] letterGrades = {'A', 'B', 'C', 'D', 'F'};
```
Initializer Lists

• Note that when an initializer list is used:
  ▪ the `new` operator is not used
  ▪ no size value is specified

• The size of the array is determined by the number of items in the initializer list

• An initializer list can be used only in the array declaration

• See `Primes.java` (page 381)
Arrays as Parameters

• An entire array can be passed as a parameter to a method

• Like any other object, the reference to the array is passed, making the formal and actual parameters aliases of each other

• Therefore, changing an array element within the method changes the original

• An individual array element can be passed to a method as well, in which case the type of the formal parameter is the same as the element type
Outline

Declaring and Using Arrays

Arrays of Objects

Variable Length Parameter Lists

Two-Dimensional Arrays

The ArrayList Class

Polygons and Polylines

Mouse Events and Key Events
Arrays of Objects

• The elements of an array can be object references

• The following declaration reserves space to store 5 references to String objects

  ```java
  String[] words = new String[5];
  ```

• It does NOT create the String objects themselves

• Initially an array of objects holds null references

• Each object stored in an array must be instantiated separately
Arrays of Objects

• The `words` array when initially declared:

![Diagram showing an array initially declared as empty](image)

• At this point, the following reference would throw a `NullPointerException`:

```java
System.out.println (words[0]);
```
Arrays of Objects

- After some `String` objects are created and stored in the array:

```
words -> "friendship" -> "loyalty" -> "honor"
```
Arrays of Objects

• Keep in mind that String objects can be created using literals

• The following declaration creates an array object called verbs and fills it with four String objects created using string literals

```java
String[] verbs = {"play", "work", "eat", "sleep"};
```
Arrays of Objects

- The following example creates an array of Grade objects, each with a string representation and a numeric lower bound.
  - See GradeRange.java (page 384)
  - See Grade.java (page 385)
- Now let's look at an example that manages a collection of CD objects.
  - See Tunes.java (page 387)
  - See CDCollection.java (page 388)
  - See CD.java (page 391)
Arrays of Objects

- A UML diagram for the Tunes program:

  Tunes
  + main (args : String[]) : void

  CDCollection
  - collection : CD[]
  - count : int
  - totalCost : double
  + addCD (title : String, artist : String, cost : double, tracks : int) : void
  + toString() : String
  - increaseSize() : void

  CD
  - title : String
  - artist : String
  - cost : double
  - tracks : int
  + toString() : String

  1

  *
Command-Line Arguments

- The signature of the `main` method indicates that it takes an array of `String` objects as a parameter.

- These values come from *command-line arguments* that are provided when the interpreter is invoked.

- For example, the following invocation of the interpreter passes three `String` objects into `main`:

  ```
  > java StateEval pennsylvania texas arizona
  ```

- These strings are stored at indexes 0-2 of the array parameter of the `main` method.

- See `NameTag.java` (page 393)
Outline

Declaring and Using Arrays
Arrays of Objects
Variable Length Parameter Lists
Two-Dimensional Arrays
The ArrayList Class
Polygons and Polylines
Mouse Events and Key Events
Variable Length Parameter Lists

• Suppose we wanted to create a method that processed a different amount of data from one invocation to the next

• For example, let's define a method called average that returns the average of a set of integer parameters

```java
// one call to average three values
mean1 = average (42, 69, 37);

// another call to average seven values
mean2 = average (35, 43, 93, 23, 40, 21, 75);
```
Variable Length Parameter Lists

- We could define overloaded versions of the `average` method
  - Downside: we'd need a separate version of the method for each parameter count

- We could define the method to accept an array of integers
  - Downside: we'd have to create the array and store the integers prior to calling the method each time

- Instead, Java provides a convenient way to create variable length parameter lists
Variable Length Parameter Lists

- Using special syntax in the formal parameter list, we can define a method to accept any number of parameters of the same type.

- For each call, the parameters are automatically put into an array for easy processing in the method.

```java
public double average (int ... list)
{
    // whatever
}
```

- Indicates a variable length parameter list.
- `int ... list` syntax indicates a variable length parameter list.
- `element type` refers to the type of elements in the array.
- `array name` refers to the name of the array.

© 2004 Pearson Addison-Wesley. All rights reserved
public double average (int ... list) {
    double result = 0.0;

    if (list.length != 0) {
        int sum = 0;
        for (int num : list)
            sum += num;
        result = (double)num / list.length;
    }

    return result;
}
Variable Length Parameter Lists

• The type of the parameter can be any primitive or object type

```java
public void printGrades (Grade ... grades)
{
    for (Grade letterGrade : grades)
        System.out.println (letterGrade);
}
```
Variable Length Parameter Lists

• A method that accepts a variable number of parameters can also accept other parameters

• The following method accepts an int, a String object, and a variable number of double values into an array called nums

```java
public void test (int count, String name, double ... nums)
{
    // whatever
}
```
Variable Length Parameter Lists

- The varying number of parameters must come last in the formal arguments
- A single method cannot accept two sets of varying parameters
- Constructors can also be set up to accept a variable number of parameters
- See VariableParameters.java (page 396)
- See Family.java (page 397)
Outline

Declaring and Using Arrays
Arrays of Objects
Variable Length Parameter Lists
Two-Dimensional Arrays
The ArrayList Class
Polygons and Polylines
Mouse Events and Key Events
Two-Dimensional Arrays

- A one-dimensional array stores a list of elements
- A two-dimensional array can be thought of as a table of elements, with rows and columns
Two-Dimensional Arrays

• To be precise, in Java a two-dimensional array is an array of arrays

• A two-dimensional array is declared by specifying the size of each dimension separately:

  ```java
  int[][] scores = new int[12][50];
  ```

• A array element is referenced using two index values:

  ```java
  value = scores[3][6]
  ```

• The array stored in one row can be specified using one index
# Two-Dimensional Arrays

<table>
<thead>
<tr>
<th>Expression</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>table</code></td>
<td><code>int[][]</code></td>
<td>2D array of integers, or array of integer arrays</td>
</tr>
<tr>
<td><code>table[5]</code></td>
<td><code>int[]</code></td>
<td>array of integers</td>
</tr>
<tr>
<td><code>table[5][12]</code></td>
<td><code>int</code></td>
<td>integer</td>
</tr>
</tbody>
</table>

- See [TwoDArray.java](#) (page 399)
- See [SodaSurvey.java](#) (page 400)
Multidimensional Arrays

- An array can have many dimensions – if it has more than one dimension, it is called a *multidimensional array*

- Each dimension subdivides the previous one into the specified number of elements

- Each dimension has its own length constant

- Because each dimension is an array of array references, the arrays within one dimension can be of different lengths
  - these are sometimes called *ragged arrays*
Outline

Declaring and Using Arrays
Arrays of Objects
Variable Length Parameter Lists
Two-Dimensional Arrays
The ArrayList Class
Polygons and Polylines
Mouse Events and Key Events
The ArrayList Class

- The `ArrayList` class is part of the `java.util` package.
- Like an array, it can store a list of values and reference each one using a numeric index.
- However, you cannot use the bracket syntax with an `ArrayList` object.
- Furthermore, an `ArrayList` object grows and shrinks as needed, adjusting its capacity as necessary.
The ArrayList Class

• Elements can be inserted or removed with a single method invocation

• When an element is inserted, the other elements "move aside" to make room

• Likewise, when an element is removed, the list "collapses" to close the gap

• The indexes of the elements adjust accordingly
The ArrayList Class

• An ArrayList stores references to the Object class, which allows it to store any kind of object

• See Beatles.java (page 405)

• We can also define an ArrayList object to accept a particular type of object

• The following declaration creates an ArrayList object that only stores Family objects

   ArrayList<Family> reunion = new ArrayList<Family>

• This is an example of generics, which are discussed further in Chapter 12
ArrayList Efficiency

• The ArrayList class is implemented using an underlying array

• The array is manipulated so that indexes remain continuous as elements are added or removed

• If elements are added to and removed from the end of the list, this processing is fairly efficient

• But as elements are inserted and removed from the front or middle of the list, the remaining elements are shifted
Outline

Declaring and Using Arrays
Arrays of Objects
Variable Length Parameter Lists
Two-Dimensional Arrays
The ArrayList Class
Polygons and Polylines
Mouse Events and Key Events
Polygons and Polylines

- Arrays can be helpful in graphics processing
- For example, they can be used to store a list of coordinates
- A polygon is a multisided, closed shape
- A polyline is similar to a polygon except that its endpoints do not meet, and it cannot be filled

- See Rocket.java (page 409)
- See RocketPanel.java (page 410)
The Polygon Class

• The Polygon class can also be used to define and draw a polygon

• It is part of the java.awt package

• Versions of the overloaded drawPolygon and fillPolygon methods take a single Polygon object as a parameter instead of arrays of coordinates

• A Polygon object encapsulates the coordinates of the polygon
Outline

Declaring and Using Arrays
Arrays of Objects
Variable Length Parameter Lists
Two-Dimensional Arrays
The ArrayList Class
Polygons and Polygons
Mouse Events and Key Events
Mouse Events

- Events related to the mouse are separated into *mouse events* and *mouse motion events*

- **Mouse Events:**

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mouse pressed</td>
<td>the mouse button is pressed down</td>
</tr>
<tr>
<td>mouse released</td>
<td>the mouse button is released</td>
</tr>
<tr>
<td>mouse clicked</td>
<td>the mouse button is pressed down and released without moving the mouse in between</td>
</tr>
<tr>
<td>mouse entered</td>
<td>the mouse pointer is moved onto (over) a component</td>
</tr>
<tr>
<td>mouse exited</td>
<td>the mouse pointer is moved off of a component</td>
</tr>
</tbody>
</table>
Mouse Events

• Mouse Motion Events:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>mouse moved</td>
<td>the mouse is moved</td>
</tr>
<tr>
<td>mouse dragged</td>
<td>the mouse is moved while the mouse button is pressed down</td>
</tr>
</tbody>
</table>

• Listeners for mouse events are created using the MouseListener and MouseMotionListener interfaces

• A MouseEvent object is passed to the appropriate method when a mouse event occurs
Mouse Events

- For a given program, we may only care about one or two mouse events
- To satisfy the implementation of a listener interface, empty methods must be provided for unused events
- See Dots.java (page 413)
- See DotsPanel.java (page 414)
Mouse Events

- *Rubberbanding* is the visual effect in which a shape is "stretched" as it is drawn using the mouse.

- The following example continually redraws a line as the mouse is dragged.

- See RubberLines.java (page 417)
- See RubberLinesPanel.java (page 418)
Key Events

• A key event is generated when the user types on the keyboard

<table>
<thead>
<tr>
<th>key pressed</th>
<th>a key on the keyboard is pressed down</th>
</tr>
</thead>
<tbody>
<tr>
<td>key released</td>
<td>a key on the keyboard is released</td>
</tr>
<tr>
<td>key typed</td>
<td>a key on the keyboard is pressed down and released</td>
</tr>
</tbody>
</table>

• Listeners for key events are created by implementing the KeyListener interface

• A KeyEvent object is passed to the appropriate method when a key event occurs
Key Events

- The component that generates a key event is the one that has the current *keyboard focus*.
- Constants in the `KeyEvent` class can be used to determine which key was pressed.
- The following example "moves" an image of an arrow as the user types the keyboard arrow keys.
- See `Direction.java` (page 421).
- See `DirectionPanel.java` (page 422).
Summary

• Chapter 7 has focused on:
  ▪ array declaration and use
  ▪ bounds checking and capacity
  ▪ arrays that store object references
  ▪ variable length parameter lists
  ▪ multidimensional arrays
  ▪ the ArrayList class
  ▪ polygons and polylines
  ▪ mouse events and keyboard events