More Arrays (Savitch, Chapter 7)

TOPICS
• Array Basics
• Arrays in Classes and Methods
• Programming with Arrays
• Searching and Sorting Arrays
• Multi-Dimensional Arrays
• Static Variables and Constants

Creating and Accessing Arrays

• An array is a special kind of object
• Think of as collection of variables of same type
• Creating an array with 7 variables of type double

\[\text{double[]} \text{ temperature = new double[7]};\]

• To access an element use
  – The name of the array
  – An index number enclosed in braces
• Array indices begin at zero

Array Basics: Outline

• Creating and Accessing Arrays
• Array Details
• The Instance Variable length
• More About Array Indices
• Analyzing Arrays

Creating and Accessing Arrays

• Figure 7.1 A common way to visualize an array

\[
\begin{array}{cccccc}
6 & 1 & 2 & 3 & 4 & 5 & 6 \\
\hline
32 & 30 & 25.7 & 26 & 34 & 31.5 & 29 \\
\hline
\end{array}
\]

• Note \textbf{class ArrayOfTemperatures}
Creating and Accessing Arrays

Enter 7 temperatures:
32
30
25.7
26
34
31.5
29
The average temperature is 29.7428
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.

Array Details

• Syntax for declaring an array with `new`

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

• The number of elements in an array is its length
• The type of the array elements is the array's base type

Square Brackets with Arrays

• With a data type when declaring an array:

```java
int[] pressure;
int pressure[];
```

• To enclose an integer expression to declare the length of the array:

```java
pressure = new int[100];
```

• To access the array with an index value:

```java
pressure[3] = keyboard.nextInt();
```
The Instance Variable `length`

- As an object an array has only one public instance variable:
  - Variable `length`
  - Contains number of elements in the array
  - It is final, value cannot be changed
- Note class `ArrayOfTemperatures2`

More About Array Indices

- Index of first array element is 0
- Last valid index is `arrayName.length - 1`
- Array indices must be within bounds to be valid:
  - When program tries to access outside bounds, run time exception occurs
- Get used to using index 0

Initializing Arrays

- Possible to initialize at declaration time
  ```java
  double[] reading = {3.3, 15.8, 9.7};
  ```
- Also may use normal assignment statements
  - One at a time
  - In a loop
    ```java
    int[] count = new int[100];
    for (int i = 0; i < 100; i++)
      count[i] = 0;
    ```
Arrays in Classes and Methods:
Outline

- Indexed Variables as Method Arguments
- Entire Arrays as Arguments to a Method
- Arguments for the Method main
- Array Assignment and Equality
- Methods that Return Arrays

Indexed Variables as Method Arguments

- Indexed variable of an array
  - Example … \( a[i] \)
  - Can be used anywhere variable of array base type can be used
- View class ArgumentDemo using indexed variable as an argument

Entire Arrays as Arguments

- Declaration of array parameter similar to how an array is declared
- Example:

  ```java
  public class SampleClass
  {
    public static void incrementArray(double[] array)
    {
      for (int i = 0; i < array.length-1; i++)
        array[i] = array[i] + 2;
    } // The rest of the class definition goes here.
  }
  ```

Entire Arrays as Arguments

- Array parameter in a method heading does not specify the length
  - An array of any length can be passed to the method
  - Inside the method, elements of the array can be changed
- When you pass the entire array, do not use square brackets in the actual parameter
Arguments for Method main

- Recall heading of method `main`
  ```java
  public static void main (String[] args)
  ```
- This declares an array
  - Formal parameter named `args`
  - Its base type is `String`
- Thus possible to pass to the run of a program multiple strings
  - These can then be used by the program

Array Assignment and Equality

- Arrays are objects
  - Assignment and equality operators behave (misbehave) as specified in previous chapter
- Variable for the array object contains memory address of the object
  - Assignment operator `=` copies this address
  - Equality operator `===` tests whether two arrays are stored in same place in memory

Two kinds of equality

- View `class TestEquals`

Not equal by `===`.
Equal by the `equals` method.

Array Assignment and Equality

- Note results of `==`
- Note definition and use of method `equals`
  - Receives two array parameters
  - Checks length and each individual pair of array elements
- Remember array types are reference types
Methods that Return Arrays

• A Java method may return an array
• View class ReturnArrayDemo
• Note definition of return type as an array
• To return the array value
  – Declare a local array
  – Use that identifier in the return statement

Programming with Arrays and Classes: Outline

• Programming Example: A Specialized List Class
• Partially Filled Arrays

Programming Example

• A specialized List class
  – Objects can be used for keeping lists of items
• Methods include
  – Capability to add items to the list
  – Also delete entire list, start with blank list
  – But no method to modify or delete list item
• Maximum number of items can be specified

Programming Example

• View class ListDemo
• Note declaration of the list object
• Note method calls
Programming Example

Enter items for the list, when prompted.
Enter an item:
Buy milk
More items for the list? yes
Enter an item:
Walk dog
More items for the list? yes
Enter an item:
Buy milk
More items for the list? yes
Enter an item:
Watch TV
More items for the list? yes
Enter an item:
Buy milk

Sample screen output

• Now view array wrapped in a class to represent a list,
  class OneWayNoRepeatsList

• Notable code elements
  – Declaration of private array
  – Method to find nth list item
  – Method to check if item is on the list or not

Partially Filled Arrays

• Array size specified at definition
• Not all elements of the array might receive values
  – This is termed a partially filled array
• Programmer must keep track of how much of array is used

Partially Filled Arrays

• Figure 7.4 A partially filled array

entry[3]: Buy milk
entry[4]: Watch TV
entry[2]: Go to beach
entry[1]: Call home

entry: Length has a value of 5
countOfEntries has a value of 3
Multidimensional-Array Basics

• Consider Figure 7.6, a table of values

<table>
<thead>
<tr>
<th>Year</th>
<th>5.00%</th>
<th>5.25%</th>
<th>6.00%</th>
<th>6.50%</th>
<th>7.00%</th>
<th>7.50%</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$1050</td>
<td>$1055</td>
<td>$1060</td>
<td>$1065</td>
<td>$1070</td>
<td>$1075</td>
</tr>
<tr>
<td>2</td>
<td>$1105</td>
<td>$1111</td>
<td>$1116</td>
<td>$1121</td>
<td>$1126</td>
<td>$1131</td>
</tr>
<tr>
<td>3</td>
<td>$1156</td>
<td>$1174</td>
<td>$1191</td>
<td>$1208</td>
<td>$1225</td>
<td>$1242</td>
</tr>
<tr>
<td>4</td>
<td>$1216</td>
<td>$1239</td>
<td>$1262</td>
<td>$1286</td>
<td>$1311</td>
<td>$1335</td>
</tr>
<tr>
<td>5</td>
<td>$1276</td>
<td>$1307</td>
<td>$1338</td>
<td>$1370</td>
<td>$1400</td>
<td>$1436</td>
</tr>
<tr>
<td>6</td>
<td>$1340</td>
<td>$1377</td>
<td>$1413</td>
<td>$1459</td>
<td>$1501</td>
<td>$1547</td>
</tr>
<tr>
<td>7</td>
<td>$1407</td>
<td>$1453</td>
<td>$1504</td>
<td>$1554</td>
<td>$1606</td>
<td>$1659</td>
</tr>
<tr>
<td>8</td>
<td>$1477</td>
<td>$1535</td>
<td>$1594</td>
<td>$1655</td>
<td>$1718</td>
<td>$1783</td>
</tr>
<tr>
<td>9</td>
<td>$1551</td>
<td>$1619</td>
<td>$1689</td>
<td>$1763</td>
<td>$1838</td>
<td>$1917</td>
</tr>
<tr>
<td>10</td>
<td>$1629</td>
<td>$1706</td>
<td>$1790</td>
<td>$1877</td>
<td>$1967</td>
<td>$2066</td>
</tr>
</tbody>
</table>

• Figure 7.7 Row and column indices for an array named table

```
```

```
```

Multidimensional-Array Basics

• We can access elements of the table with a nested for loop

• Example:

```java
for (int row = 0; row < 10; row++)
    for (int column = 0; column < 6; column++)
        table[row][column] = balance(1000.00, row + 1, (5 * 0.5 * column));
```

• View class InterestTable

```
```

```java
Balances for Various Interest Rates Compounded Annually (Rounded to whole Dollar Amounts)
<table>
<thead>
<tr>
<th>Years</th>
<th>5.00%</th>
<th>5.25%</th>
<th>6.00%</th>
<th>6.50%</th>
<th>7.00%</th>
<th>7.50%</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$1050</td>
<td>$1055</td>
<td>$1060</td>
<td>$1065</td>
<td>$1070</td>
<td>$1075</td>
</tr>
<tr>
<td>2</td>
<td>$1105</td>
<td>$1111</td>
<td>$1116</td>
<td>$1121</td>
<td>$1126</td>
<td>$1131</td>
</tr>
<tr>
<td>3</td>
<td>$1156</td>
<td>$1174</td>
<td>$1191</td>
<td>$1208</td>
<td>$1225</td>
<td>$1242</td>
</tr>
<tr>
<td>4</td>
<td>$1216</td>
<td>$1239</td>
<td>$1262</td>
<td>$1286</td>
<td>$1311</td>
<td>$1335</td>
</tr>
<tr>
<td>5</td>
<td>$1276</td>
<td>$1307</td>
<td>$1338</td>
<td>$1370</td>
<td>$1400</td>
<td>$1436</td>
</tr>
<tr>
<td>6</td>
<td>$1340</td>
<td>$1377</td>
<td>$1413</td>
<td>$1459</td>
<td>$1501</td>
<td>$1547</td>
</tr>
<tr>
<td>7</td>
<td>$1407</td>
<td>$1453</td>
<td>$1504</td>
<td>$1554</td>
<td>$1606</td>
<td>$1659</td>
</tr>
<tr>
<td>8</td>
<td>$1477</td>
<td>$1535</td>
<td>$1594</td>
<td>$1655</td>
<td>$1718</td>
<td>$1783</td>
</tr>
<tr>
<td>9</td>
<td>$1551</td>
<td>$1619</td>
<td>$1689</td>
<td>$1763</td>
<td>$1838</td>
<td>$1917</td>
</tr>
<tr>
<td>10</td>
<td>$1629</td>
<td>$1706</td>
<td>$1790</td>
<td>$1877</td>
<td>$1967</td>
<td>$2066</td>
</tr>
</tbody>
</table>
```
Java's Representation of Multidimensional Arrays

- Multidimensional array represented as several one-dimensional arrays
- Given
  
  ```java
  int[][] table = new int[10][6];
  ```
- Array table is actually 1 dimensional of type `int[]`
  
  - It is an array of arrays
- Important when sequencing through multidimensional array

Summary

- An array is a collection of variables all of the same type
- Arrays are objects, created with operator `new`
- Elements numbered starting with 0, ending with 1 less than length
- Indexed variable can be used as a parameter – treated like variable of base type

Summary

- Entire array can be passed as parameter to a method
- Method return value can be an array
- Partially filled array usually stores values in initial segment, use an `int` to track how many are used

Summary

- Multidimensional arrays are implemented as an array of arrays
- Treat two-dimensional array as a table with rows and columns
public vs. private

- Variables defined as public are visible (and changeable) outside the class
  - Sometimes called global variables
- Methods defined as public are visible (and callable) outside the class
- Private variables are visible/defined only within their scope

Scope

- Class instance variables
  - Can be accessed anywhere in the class
- Method instance variables
  - Can be accessed anywhere in the method
  - Method parameters are local to the method
  - Method parameters are initialized to the value of the passed argument
- Loop instance variables
  - Can be accessed anywhere in the loop

Static Variables

- Static variables are shared by all objects of a class
  - Variables declared static final are considered constants – value cannot be changed
- Variables declared static (without final)
  - Can be changed
  - Only one instance of the variable exists
  - It can be accessed by all instances of the class

Static Variables

- Static variables also called class variables
  - Contrast with instance variables
- Do not confuse class variables with variables of a class type
- Both static variables and instance variables are sometimes called fields or data members
Using Named Constants

• To avoid confusion, always name constants (and variables).
  \[ \text{area} = \pi \times \text{radius} \times \text{radius}; \]
  is clearer than
  \[ \text{area} = 3.14159 \times \text{radius} \times \text{radius}; \]
• Place constants near the beginning of the program.

Named Constants

• Once the value of a constant is set (or changed by an editor), it can be used (or reflected) throughout the program.
  \[ \text{public static final double INTEREST_RATE} = 6.65; \]
• If a literal (such as 6.65) is used instead, every occurrence must be changed, with the risk than another literal with the same value might be changed unintentionally.

Declaring Constants

• Syntax
  \[
  \text{public static final } \text{Variable_Type} = \text{Constant;}
  \]
• Examples
  \[
  \text{public static final double } \pi = 3.14159;
  \text{public static final String MOTTO = "The customer is always right.";}
  \]
• By convention, uppercase letters are used for constants.