Java Chapter 7 Part 2

* Arrays: Two-Dimensional, Searching, Sorting
* CIS 255 • Shelby-Hoover Campus

Multidimensional Arrays (7.9)

* An array with a single size is like a row of values
* A program may contain a multidimensional array by placing the size of each dimension in a separate set of square brackets after the array name
  + A two-dimensional array is like a table
  + A three-dimensional array is like a document containing multiple tables or worksheets
* Each element of the array will have a subscript for each dimension
* A program often uses nested loops to proceed through every element in a multidimensional array

Multidimensional Array Example

* An array with rows representing the divisions of a company and columns representing the sales for each quarter (CorpSales.java, Code Listing 7-18)
  + The array has three rows and four columns, with each row representing a company’s division and each column representing a quarter of the year
  + The first (outer) loop prompts the user for input for each division, with the inner loop obtaining the values for each quarter in the current division
  + The second (inner) loop set adds each value in the array to an accumulator variable

Multidimensional Arrays Used

* A program may initialize a two-dimensional array using an initialization list with a set of values for each row in a separate set of curly brackets, commas separating the sets, and another set of curly brackets that surrounds the entire set of rows
* The length attribute indicates the size of the first dimension (e.g., the number of rows); to obtain the sizes of other dimensions, apply the length attribute to one of the rows (after the square brackets that contain the subscript of the row whose size you’re trying to determine):  
    
  numbers[row].length
* An array parameter for a multidimensional array requires a set of square brackets for each dimension

Ragged Arrays

* An array does not have to be “square”; in a two-dimensional array, each row can have its own length:  
   int[][] ragged = new int[4][];  
   ragged[0] = new int[3];  
   ragged[1] = new int[4];  
   ragged[2] = new int[5];  
   ragged[3] = new int[6];
* This can make array processing more difficult

Sequential Search (7.8)

* If the elements of an array are unsorted, the only way to determine if a value exists in an array is to loop through the elements sequentially, comparing the value being sought to each element
* The example in the textbook (SearchArray.java, Code Listing 7-17) uses a boolean variable as a flag to indicate if and when the value is found
  + The program initializes the flag to false (the program has not found the value yet)
  + The loop continues until the flag becomes true or the loop reaches the end of the array
  + If the element at the current subscript matches the desired value, the program sets the flag to true and remembers its position
  + The method returns the subscript where the value is found or -1 if the value is not found

Sorting (7.11)

* Some programs require the data in an array to be sorted
* There are several algorithms that can be used to take an array of existing values and sort them in either ascending or descending order
  + Bubble sort: keep swapping the pairs that are out of order until no more swaps are needed
  + Insertion sort: move each element backwards in the elements already sorted to its proper location
  + Selection sort: find the smallest value of the elements that have yet to be sorted, and swap it into the first position in the unsorted elements

Selection Sort

* The program SelectionSortDemo.java contains a method (shown in section 7.11) that performs a selection sort
  + The outer loop allows the method to find the smallest value for all but the last element (the largest element will automatically be swapped there)
  + The inner loop examines every element from the current (starting) position to the end of the array, maintaining the index of the minimum value
  + Once the method knows where the minimum value is, it swaps that value to the starting position

Binary Search

* If the array is already sorted, searching for a value can be more efficient
* The binary search algorithm repeatedly cuts the number of elements remaining to be examined in half
  + The first element examined is at the midpoint of the array
  + If the element at that position is not a match, compare the value being sought to that value
    - If the value being sought is smaller, repeat the process for the elements with lower subscripts
    - If the value being sought is larger, repeat the process for the elements with higher subscripts
  + The process ends when either the boolean variable indicates that the value has been found or there are no elements remaining (when first is greater than last)
* BinarySearchDemo.java includes a method (shown at the end of section 7.11) to implement this algorithm

The main Method’s args

* The main method contains a String array parameter named args
* The program uses this array to see what arguments have been passed into the program from the command line (example: CommandLine.java, Code Listing 7-21)
* To send these arguments to the program, you must run the program from the command prompt (e.g., cmd.exe):  
    
  java ClassName arg1 arg2 arg3

Variable-Length Argument Lists

* When a method has an array parameter, the size of the array is not specified; an array of any size could be the argument
* Java provides another mechanism for a method to receive a set of values with an unknown length, a **variable-length argument list**
  + The parameter for a variable-length argument list is called a **vararg parameter** and is declared with an ellipsis (three periods) after the data type of the values being received and before the name
  + Use the enhanced for loop or subscript notation to process these values
* Example: VarargsDemo1.java (the sum method in section 7.12)

Reminders for Chapter 7

* Declare an array by setting its base type, name, and size
* To access an individual array element, place its subscript in square brackets after the name
* The only valid subscripts for an array are integers between 0 and size – 1
* The subscript of an element and the value contained in the element are two separate values
* Use a loop to apply the same processing to each element in the array, but make sure the subscript variable remains less than the size (.length)
* Comparison and assignment should be done on an element-by-element basis, not between the array names themselves
* The elements in an array of objects must be constructed separately
* A multidimensional array must have a size for every dimension, and each element must have a subscript for every dimension