CS1004: Intro to CS in Java, Spring 2005

Lecture #18: Arrays and algorithms

Janak J Parekh
janak@cs.columbia.edu

Agenda

- One last thing you need for HW#4: command-line arguments
- Don’t worry, it’s straightforward, but we should cover arrays as a topic first

Arrays

- An array is an ordered list of values

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

The entire array has a single name
Each value has a numeric index
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:
  \[
  \begin{align*}
  \text{scores}[2] &= 89; \\
  \text{scores}[\text{first}] &= \text{scores}[\text{first}] + 2; \\
  \text{mean} &= (\text{scores}[0] + \text{scores}[1])/2; \\
  \text{System.out.println} &\left(\text{"Top = "} + \text{scores}[5]\right);
  \end{align*}
  \]

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 characters, 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:

```
   79
  87
  94
  82
  67
  98
  87
  81
  74
  91
```

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];
  ```
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
- Beware of off-by-one errors!

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

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:
  ```java
  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.
- Note no new or size declaration; automatic.
- Examples:
  
  ```java
  int[] units = {147, 323, 89, 933, 540, 269, 97, 114, 298, 476};
  char[] letterGrades = {'A', 'B', 'C', 'D', 'F'};
  ```

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.

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:

```
words
```

- 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
```

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`:
  ```bash
  $ java StateEval pennsylvania texas arizona
  ```
- These strings are stored at indexes 0-2 of the array parameter of the `main` method.
Putting it together…

- Use the iterator model from the Scanner class
- The fact that a Scanner is an iterator is particularly helpful when reading input from a file
  - What if we wanted to change our averaging program to read from a file containing the numbers?
  - Need to handle IOException; we do so by “throwing” for now
  - Use command-line arguments to specify the file to read

So, what can we do?

- Book examples
  - Palindrome tester
  - URL dissector (huh?)
  - Number reverser
  - Multiplicative table
  - Stars (used for HW)
- We need to start thinking on how we can formulate these problems
  - Describe the algorithm in greater detail

Representing algorithms

- Code (of course)
- Natural language (steps, etc.)
- Psuedocode
  - English language constructs modeled to look like statements available in most programming languages
  - Steps presented in a structured manner (numbered, indented, etc.)
  - No fixed syntax for most operations is required, but more readable than natural language
  - Emphasis is on process, not notation
  - Can be easily translated into a programming language
How do we come up with algorithms?

- An imprecise science at best: problem-solving
  - Understand the problem
  - Get an idea of how/which algorithm might solve the problem
  - Formulate the algorithm and represent as a program
  - Evaluate the program for accuracy and potential to solve other problems
  - This is not much help, is it?

“Get a foot in the door”

- Try doing the first (few) step(s) by hand
  - Look at what you had to do to accomplish it
  - See if you can reapply this to continue solving the problem
- Reapply another solution
- Stepwise refinement
  - Look at the problem from a very high level
  - Break it down repeatedly into smaller pieces, until we get a set of algorithmic steps

Board examples

1. Palindrome checker (see book for code)
2. Print out the first $n$ Fibonacci numbers
3. Search for a number in a list
4. Reverse a list (array) of numbers
Next time

- Continue working with algorithms