Arrays of Objects

October 9, 2006

ComS 207: Programming I (in Java)
Iowa State University, FALL 2006
Instructor: Alexander Stoytchev

Quick review of last lecture

Arrays

- An *array* is an ordered list of values

The entire array has a single name

Each value has a numeric *index*

```
        0   1   2   3   4   5   6   7   8   9
scores 79  87  94  82  67  98  87  81  74  91
```
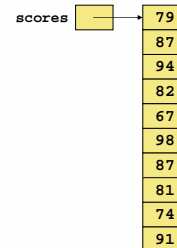
An array of size N is indexed from zero to N-1

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

Arrays

- Another way to depict the `scores` array:

```
scores ─→ 79
          87
          94
          82
          67
          98
          87
          81
          74
          91
```

Declaring Arrays

- The `scores` array could be declared as follows:

```
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:

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

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

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

The iterator version of the `for` loop

```
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)**

What for/in can't do

```
int[] primeNums = {2, 3, 5, 7, 11, 13, 17, 19};

for (int i=0; i< primeNums.length; i++)
  {
      System.out.print("primeNums[" + i + "]= ");
      System.out.println(primeNums[i]);
  }
```

What for/in can't do

```
String word="Test";

for (int i=0; i< word.length(); i++)
{
   if(i>0)
     System.out.print(", ");

   System.out.print( word.charAt(i) );
}
```

Other Stuff from Sections 7.1 & 7.2

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:**

```
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)**

Example: `Primes.java` (page 381)

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**

    problem

    ```
    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:**

    ```
    scores.length
    ```

- **Note that `length` holds the number of elements, not the largest index**

- **See `ReverseOrder.java` (page 375)**

- **See `LetterCount.java` (page 376)**

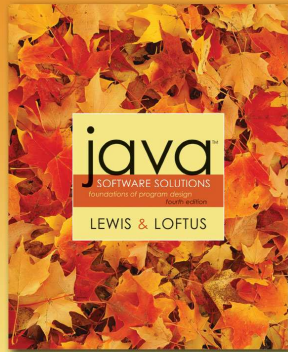**Example: `LetterCount.java` (page 376)**

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**

Chapter 7

Section 7.3

java
SOFTWARE SOLUTIONS
foundations of program design
fourth edition

LEWIS & LOFTUS

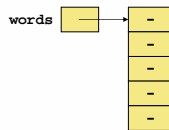---

Arrays of Objects

- **The elements of an array can be object references**
- **The following declaration reserves space to store 5 references to `String` objects**

      ```
      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
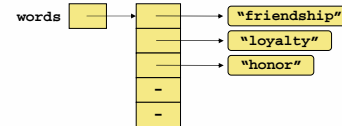
- **The `words` array when initially declared:**

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

      ```
      System.out.println (words[0]);
      ```

Arrays of Objects

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

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**

```
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)
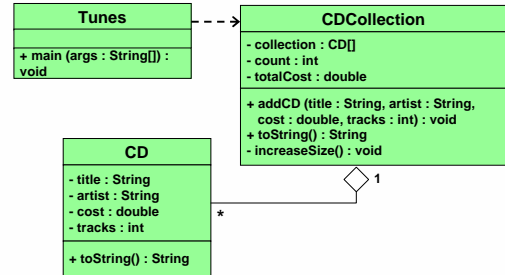- See **CDCollection.java** (page 388)
- See **CD.java** (page 391)

Run Examples

Arrays of Objects

- **A UML diagram for the `Tunes` program:**

```
Tunes                          CDCollection
------------------             ------------------
+ main (args : String[]) :     - collection : CD[]
  void                         - count : int
                               - totalCost : double

                               + addCD (title : String, artist : String,
                                 cost : double, tracks : int) : void
        CD                     + toString() : String
------------------             - increaseSize() : void
- title : String
- artist : String                          1
- cost : double
- tracks : int          *
------------------
+ toString() : String
```

THE END