7.3 – Arrays of Strings (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];
  ```
- Initially an array of objects holds null references
- System.out.println (words[0]);
- At this point, the above line would throw a NullPointerException
- Each object of an array must be instantiated separately
  ```java
  words[1] = "loyalty";
  ```
- You can initialize an array at the time of its creation:
  ```java
  String[] verbs = {"play", "work", "eat", "sleep"};
  ```

7.3 – Arrays of Objects

A UML diagram for the Tunes program
public class CD
{
    private String title, artist;
    private double cost;
    private int tracks;

    public CD (String name, String singer, double price, int numTracks)
    {
        title = name;
        artist = singer;
        cost = price;
        tracks = numTracks;
    }

    public String toString()
    {
        return title + " by " + artist + " costs " + cost + " and has " + tracks + " tracks."
    }
}
private void increaseSize () {
    CD[] temp = new CD[collection.length * 2];
    for (int cd = 0; cd < collection.length; cd++)
        temp[cd] = collection[cd];
    collection = temp;
}
}

7.4 – CommandLine.java

//********************************************************************
// CommandLine.java
Java Foundations
//********************************************************************
public class CommandLine {

    // Prints all of the command line arguments provided by the user.
    public static void main (String[] args) {
        for (String arg : args)
            System.out.println (arg);
    }
}

7.4 - 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
- Very useful to create flexible system-level programs

7.5 – Variable Length Parameter Lists

- Suppose we wanted to create a method that processed a different amount of data from one invocation to the next
  
  // 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);

- We could define overloaded versions of the average method
  - But: 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
- Java provides a way to create variable length parameter lists
7.5 - 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);
}
```

- The method can also accept other parameters, but only one variable length param list, at the end (why?)

- For example:

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

7.6 - Two-Dimensional Arrays

- A two-dimensional array can be thought of as a table of elements, with rows and columns

- In Java a two-dimensional array is an array of arrays

- It is declared by specifying the size of each dimension separately

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

- An 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

```java
test
```
7.6 – Two-Dimensional Arrays

<table>
<thead>
<tr>
<th>Expression</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>int[][]</td>
<td>array of integer arrays (2D array)</td>
</tr>
<tr>
<td>table[3]</td>
<td>int[]</td>
<td>array of integers</td>
</tr>
<tr>
<td>table[3][2]</td>
<td>int</td>
<td>Integer (= 6)</td>
</tr>
<tr>
<td>table[3].length</td>
<td>int</td>
<td>Integer (= 6)</td>
</tr>
<tr>
<td>table.length</td>
<td>int</td>
<td>Integer (= 5)</td>
</tr>
</tbody>
</table>

// TwoDArray.java Java Foundations
// Demonstrates the use of a two-dimensional array.
//********************************************************************
public class TwoDArray
{
    // TwoDArray
    public static void main (String[] args)
    {
        int[][] table = new int[5][10];
        // Load the table with values
        for (int row=0; row < table.length; row++)
            for (int col=0; col < table[row].length; col++)
                table[row][col] = row * 10 + col;

        // Print the table
        for (int row=0; row < table.length; row++)
            for (int col=0; col < table[row].length; col++)
                System.out.print (table[row][col] + " ");
        System.out.println();
    }
}