ACM Jett, Duke Workshop II

Agenda

<table>
<thead>
<tr>
<th>Time</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>8:30 - 9:00</td>
<td>Breakfast/Welcome</td>
</tr>
<tr>
<td>9:00 - 10:30</td>
<td>Arrays, ArrayLists, Exam Questions</td>
</tr>
<tr>
<td>10:30 - 11:45</td>
<td>Programming Examples: IO, Files, Arrays</td>
</tr>
</tbody>
</table>

11:45 - 12:30 Lunch

12:30 - 2:00 Event Programming Overview/Examples
2:00 - 2:30 Discussion of JETT Future
2:30 - 3:30 Model, View, Controller (MVC)

Morning I

We'll use the A Exam Questions (the first two) to discuss different parts of Java, specifically the differences between arrays and the `java.util.ArrayList` class. This will also lead into discussions of interfaces, inheritance, the Java subset, and related issues.

2003 AP exam questions in Java

Morning II

Using the code in `WordCounter.java` as a starting point (and the code in `Dice.java` and `DiceAnalyzer.java` as an example of counting) program solutions to the following problems using JCreator.

1. Modify method `standardCount` so that it calculates how many 1-letter words, 2-letter words, ... 15-letter words there are. Print a table of occurrences, similar to the table printed in `DiceAnalyzer.java` programmer.

   You'll need to create an array as an instance field/data member, initialize it, update the counts in method `standardCount` and then add a new method (and call it) to print the contents of the array. Test the program using `data/poe.txt` and `data/hamlet.txt`.

2. Use an ArrayList to store all the words read from a file, but don't store a word if it's already in the list. You'll find the ArrayList method `indexOf` very helpful, see the API (via JCreator). Don't print the contents of the ArrayList when done, but do print (using a method `reportStats`) the number of words stored. Test the program using `data/poe.txt` and `data/hamlet.txt`.

3. Modify the program so that instead of using an ArrayList, it uses a TreeSet (also from java.util). You'll need to use just the four statements below, but you'll need to put them in the proper location. See how much longer/shorter your program takes to run than when using the ArrayList version.

   ```java
   TreeSet mySet;
   ```
mySet = new TreeSet();
mySet.add(word);
System.out.println("# unique words = " + mySet.size());

Afternoon I

We'll use Browser.java to discuss GUI programs, Swing, and event-driven programming. We won't go into too many details, just a quick overview to get the feel for what goes on, and see how ugly and pretty things can be.

1. Currently if a URL doesn't begin with http:// the browser won't load a page correctly. Fix the program so that if a user-entered URL doesn't begin with the proper prefix, then it is added to the URL before loading.

2. The back/next buttons don't work. Design a solution so that these buttons will work as you expect them to. Implement the solution.

   (hint: One simple idea is to use an ArrayList of URLs that have been visited and a current index into this list. Going back/next decrements or increments, respectively, the index.)

3. Enable/Disable the buttons and menu items appropriately depending on whether there is a next or back page to visit.

Afternoon II

We'll use the code that plays same-game aka clickomania to discuss model-view-controller.

Copy the files to a new folder, then modify only the model so that a game of light-out is played. We'll demo lights out so you can see how it works.

---

Owen L. Astrachan

Last modified: Sat May 17 01:38:57 EDT 2003