Arrays of/in Objects
Partially Filled Arrays
ArrayLists
Do-It-Yourself ArrayLists

Lecture 17

Readings
Next Week: Ch 8.3-8.8 and into Ch 9.1-9.3
(Ch 9.3-9.8 and Ch 11.1-11.3 in old 2nd ed)

(Reminder: Readings are absolutely vital for learning this stuff!)

Labs and Tutorials
Next Week: Lab #8 – A catch-up lab…

Learning Goals
By the end of class today you will be able to…

- Write programs with multi-dimensional arrays, arrays of objects, and arrays inside objects.
- Read and write programs that use partially-filled arrays (in the typical, idiomatic manner).
- Read and write programs that use Java’s ArrayList class.
- Implement your own ArrayList class.

(Ulterior Learning Goal: Get increasingly comfortable with arrays…)

Last Week: Organizing Data
- Is there a better way to store this data?

<table>
<thead>
<tr>
<th>studentNames</th>
<th>labMarks</th>
<th>labTotals</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Alan”</td>
<td>4 4 4 4</td>
<td>40</td>
</tr>
<tr>
<td>“Wolfgang”</td>
<td>4 4 4 4</td>
<td>40</td>
</tr>
<tr>
<td>“Puff Daddy”</td>
<td>2 1 0 0</td>
<td>3</td>
</tr>
</tbody>
</table>

Last Week: Parallel Arrays
- “Parallel arrays” are common, but usually not the best way to do things.

<table>
<thead>
<tr>
<th>studentNames</th>
<th>labMarks</th>
<th>labTotals</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Alan”</td>
<td>4 4 4 4</td>
<td>40</td>
</tr>
<tr>
<td>“Wolfgang”</td>
<td>4 4 4 4</td>
<td>40</td>
</tr>
<tr>
<td>“Puff Daddy”</td>
<td>2 1 0 0</td>
<td>3</td>
</tr>
</tbody>
</table>
**Last Week: Arrays of Objects**
- Usually, it's better to create an array of objects instead.

```
CPSC111Student
  name: "Alan"
  labMarks: [4, 4, 4, 4]
  labTotal: 16
```

**Organizing Data**
- Create a CPSC111Student class!
  - Constructors
  - Methods to get and set name
  - Methods to get and set lab scores.
  - Method to compute total lab score.
  - Etc.

  For simplicity, let's just worry about the student's name and lab scores…
Organizing Data

- Create a CPSC111Student class!
  - Constructors
  - Methods to get and set name
  - Methods to get and set lab scores.
  - Method to compute total lab score.
    - User probably wants a getTotalLabScore() method:
      - What parameters does this take? What’s the return value? Let’s implement this…

getTotalLabScore() Version 1

- We can add up the lab scores each time called.
- This can be wasteful if getTotalLabScore is called often – the same work gets re-done repeatedly.

getTotalLabScore() Version 2

- We can put the result in a variable totalLabScore.
- getTotalLabScore() just returns that value.
- No wasted computation when getTotalLabScore() called.

getTotalLabScore() Version 3

- We can put the result in a variable totalLabScore.
- getTotalLabScore() just returns that value.
- We add a new method computeTotalLabScore() that computes totalLabScore.
- Gives correct results, but only if user always remembers to call computeTotalLabScore().
  - Bad Design: Violates data protection!
getTotalLabScore() Version 4
- We can put the result in a variable totalLabScore.
- getTotalLabScore() just returns that value.
- The object keeps track of whether totalLabScore is up-to-date. When getTotalLabScore() is called, it calls computeTotalLabScore() only if needed.

Teen Talk Barbie™ Reloaded
- A few lectures back, we wrote a program that learns some phrases and prints them back at random (inspired by Mattel’s Teen Talk Barbie™ doll).
  - The goal was to have a fun example to introduce arrays.
  - But the programming style wasn’t very good.
  - Better to create a TalkingDoll class…

TalkingDoll Class
- Constructor that specifies name and maximum vocabulary size.
- Getters/Setters for the name.
- A method to add phrases to the doll.
- A method to get a random phrase back.

Partially Filled Arrays
- Arrays have a fixed size, but often we want to hold a variable number of items.
- A Very Common Solution:
  - Make an array bigger than you need.
  - Have an int variable to keep track of what you are actually using.

Partially Filled Arrays
- Arrays have a fixed size, but often we want to hold a variable number of items.
- A Very Common Solution:
  - Make an array bigger than you need.
  - Have an int variable to keep track of what you are actually using.
Partially Filled Arrays

- Arrays have a fixed size, but often we want to hold a variable number of items.
- A Very Common Solution:
  - Make an array bigger than you need.
  - Have an int variable to keep track of what you are actually using.

```
partiallyFilledArrays.
```

Java’s ArrayList Class

- What if you don’t know (or want to force) a fixed maximum size in advance?
- Java has an ArrayList class for this case:
  - import java.util.ArrayList
  - Declare:
    ```
    ArrayList&lt;type&gt; phrases = new ArrayList&lt;type&gt;();
    ```
  - Methods:
    ```
    int size()
    void add(&lt;type&gt; newValue)
    &lt;type&gt; get(int index)
    void set(int index, &lt;type&gt; newValue)
    ```
  - &lt;type&gt; can be any object type. See Ch 7.2.

TalkingDoll with ArrayLists

- Let’s redo the TalkingDoll class using ArrayLists...

Do-It-Yourself ArrayLists

- ArrayLists are nothing magical!
  - (OK, the generic &lt;type&gt; stuff is kind of magic.)
- It’s just a class. If we fix the type of the elements (e.g., have an ArrayList of String), you know enough to write your own version.
- But how do you allow arrays to grow?
Real-Life Analogy: Moving Homes

- A house (or condo, apartment, etc.) has a fixed size. What happens when your family grows and you need more space?

Answer: You buy a bigger place, and then you pack up and move all your stuff to the new place, and get rid of your old home.

Making Your Own ArrayList

- An array has a fixed size. What happens when your list grows and you need more space?
- Answer: You allocate a bigger array, and then you pack up and move all your stuff to the new array, and get rid of your old array.

Making Your Own ArrayList

- Answer: You allocate a bigger array, and then you pack up and move all your stuff to the new array, and get rid of your old array.

```
<table>
<thead>
<tr>
<th>aCount</th>
<th>a</th>
<th>a.length</th>
<th>newA</th>
<th>newA.length</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>3 1 4 1</td>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Making Your Own ArrayList

- Answer: You allocate a bigger array, and then you pack up and move all your stuff to the new array, and get rid of your old array.

```
<table>
<thead>
<tr>
<th>aCount</th>
<th>a</th>
<th>a.length</th>
<th>newA</th>
<th>newA.length</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>3 1 4 1</td>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Making Your Own ArrayList

- Answer: You allocate a bigger array, and then you pack up and move all your stuff to the new array, and get rid of your old array.

```
<table>
<thead>
<tr>
<th>aCount</th>
<th>a</th>
<th>a.length</th>
<th>newA</th>
<th>newA.length</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>3 1 4 1</td>
<td>8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Making Your Own ArrayList

Answer: You allocate a bigger array, and then you pack up and move all your stuff to the new array, and get rid of your old array.

- aCount
- a
- a.length
- newA
- newA.length

Answer: You allocate a bigger array, and then you pack up and move all your stuff to the new array, and get rid of your old array.

- aCount
- a
- a.length
- newA
- newA.length

Answer: You allocate a bigger array, and then you pack up and move all your stuff to the new array, and get rid of your old array.

- aCount
- a
- a.length
- newA
- newA.length

Answer: You allocate a bigger array, and then you pack up and move all your stuff to the new array, and get rid of your old array.

- aCount
- a
- a.length
- newA
- newA.length

Answer: You allocate a bigger array, and then you pack up and move all your stuff to the new array, and get rid of your old array.

- aCount
- a
- a.length
- newA
- newA.length

Answer: You allocate a bigger array, and then you pack up and move all your stuff to the new array, and get rid of your old array.

- aCount
- a
- a.length
- newA
- newA.length

Answer: You allocate a bigger array, and then you pack up and move all your stuff to the new array, and get rid of your old array.

- aCount
- a
- a.length
- newA
- newA.length

Answer: You allocate a bigger array, and then you pack up and move all your stuff to the new array, and get rid of your old array.

- aCount
- a
- a.length
- newA
- newA.length
**MyStringArrayList**

- Let's create a MyStringArrayList class.
- Methods:
  - int size()
  - void add(String newString)
  - String get(int index)