ArrayList

Talk to your neighbors:
Dogs or cats?

BEFORE WE START

Questions during Class?
Raise hand or send here

sli.do   #cse122
Lecture Outline

- Announcements
- Code Quality Recap
- ArrayList Recap
- ArrayList Examples
Announcements

• P0 is out, due tomorrow @11:59pm
  - make sure you hit ‘mark’ at least once on Ed

• IPL is open!

• Java Review session last Monday
  - recording now available on calendar
  - [Java Tutorial](#)
Lecture Outline

• Announcements
• **Code Quality Recap**
• ArrayList Recap
• ArrayList Examples
Code Quality - Functional Decomposition

Goal: make our code more understandable to humans

• Subgoal: `main` is a concise summary of the program
• Subgoal: all operations in a method are related
• Subgoal: no trivial methods
Code Quality - Functional Decomposition

Example: Solution from Friday’s lecture
Code Quality - Other

122 Code Quality Guide

which includes..

• naming conventions: variableNames, methodNames, ClassNames, CLASS_CONSTANTS
• consistent indentation
• when possible, avoid repeated/redundant code
Commenting

122 Commenting Guide

Header comment

Class comment - what does this class do?

Method comments - every method besides main
• behavior of the method, including
  - parameters
  - returned value (if any)
  - thrown exceptions (if any)
Lecture Outline

• Announcements
• Code Quality Recap
• **ArrayList Recap**
• ArrayList Examples
NEW DATA STRUCTURE DAY!! 🎉
ArrayList - What is it good for?

What is it?
● An **ordered, indexed** collection of elements
● All elements must be of same type*
● Dynamically sized (ie: add and remove elements w/out specifying an initial size)

What is it particularly good at?
● **get** elements by index (like an array)
● **set** elements at index (like an array)
● **add** new elements, as many as you want! (unlike an array)
● **add & remove** elements by index (unlike an array)
ArrayList - dynamic size

```
4   8   16  23  42
list.size(): 5

list.add(2, 15);

4   8   15   16  23  42
list.size(): 6
```
## ArrayList Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>add(type $element$)</td>
<td>Adds $element$ to the end of the ArrayList</td>
</tr>
<tr>
<td>add(int $index$, type $element$)</td>
<td>Adds $element$ to the specified $index$ in the ArrayList</td>
</tr>
<tr>
<td>size()</td>
<td>Returns the number of elements in the ArrayList</td>
</tr>
<tr>
<td>contains(type $element$)</td>
<td>Returns true if $element$ is contained in the ArrayList, false otherwise</td>
</tr>
<tr>
<td>get(int $index$)</td>
<td>Returns the element at $index$ in the ArrayList</td>
</tr>
<tr>
<td>remove(int $index$)</td>
<td>Removes the element at $index$ from the ArrayList and returns the removed element.</td>
</tr>
<tr>
<td>indexOf(type $element$)</td>
<td>Returns the index of $element$ in the ArrayList; returns -1 if the $element$ doesn’t exist in the ArrayList</td>
</tr>
<tr>
<td>set(int $index$, type $element$)</td>
<td>Sets the element at $index$ to the given $element$ and returns the old value</td>
</tr>
</tbody>
</table>
Wrapper Types

```java
int         =>   Integer
double      =>   Double
char        =>   Character
boolean     =>   Boolean

int[] list = new int[1];
list[0] = 3;
// ... is like ...
ArrayList<Integer> list = new ArrayList<Integer>();
list.add(3);
```
Lecture Outline

• Announcements
• Code Quality Recap
• ArrayList Recap
• ArrayList Examples
ArrayList Construction Style

```java
ArrayList<Double> list = new ArrayList<Double>();
```

```java
ArrayList<Double> list = new ArrayList<>();
```

```java
List<Double> list = new ArrayList<>();
```
Practice : Think

How best to construct an ArrayList which stores characters?

A. `List<char> list = new ArrayList<char>();`
B. `ArrayList<Character> list = new ArrayList<Character>();`
C. `List<Character> list = new ArrayList<>();`
D. `ArrayList<> list = new ArrayList<Character>();`
Practice - reading ArrayList code
Practice : Think

What is the best “plain English” description of this method?

public static void method(List<Double> list) {
    for (int i = 0; i < list.size(); i++) {
        System.out.println(" "+ i + ":" + list.get(i));
    }
}

A) Prints stuff
B) Prints out the list from front to back, with elements numbered 0, 1, 2, …
C) Prints out the list from front to back
D) Prints out the list from back to front
E) Prints out the elements of the list using a for loop that starts at 0 and runs until one less than the size of the list and at each point prints out the element at that index.
What is the best “plain English” description of this method?

```java
public static void method(List<Double> list) {
    for (int i = 0; i < list.size(); i++) {
        System.out.println(" "+ i + ":" + list.get(i));
    }
}
```

A) Prints stuff  
B) Prints out the list from front to back, with elements numbered 0, 1, 2, …  
C) Prints out the list from front to back  
D) Prints out the list from back to front  
E) Prints out the elements of the list using a for loop that starts at 0 and runs until one less than the size of the list and at each point prints out the element at that index.
loadFromFile

Write a method called `loadFromFile` that accepts a `Scanner` as a parameter and returns a new `ArrayList` of `Strings` where each element of the `ArrayList` is a line from the `Scanner`, matching the order of the `Scanner`'s contents.

e.g., the first line in the `Scanner` is stored at index 0, the next line is stored at index 1, etc.
moveRight

Write a method called moveRight that accepts an ArrayList of integers list and an int n and moves the element at index n one space to the right in list.

For example, if list contains [8, 4, 13, -7] and our method is called with moveRight(list, 2), after the method call list would contain [8, 4, -7, 13] (notice that the elements at indexes 2 and 3 have swapped places).
Practice: Think

What ArrayList methods (and in what order) could we use to implement the moveRight method?

A) `list.remove(n); list.add(n);`

B) `int element = list.remove(n); list.add(n, element);`

C) `list.add(n); list.remove(n-1);`

D) `int element = list.remove(n); list.add(n+1, element);`
What ArrayList methods (and in what order) could we use to implement the moveRight method?

A) `list.remove(n);`  
   `list.add(n);`

B) `int element = list.remove(n);`  
   `list.add(n, element);`

C) `list.add(n);`  
   `list.remove(n - 1);`

D) `int element = list.remove(n);`  
   `list.add(n + 1, element);`
Edge Cases! (And Testing)

When writing a method, especially one that takes input of some kind (e.g., parameters, user input, a Scanner with input) it’s good to think carefully about what assumptions you can make (or cannot make) about this input.

**Edge case**: A scenario that is uncommon but possible, especially at the “edge” of a parameter’s valid range.

❓ What happens if the user passes a negative number to `moveDown`?

❓ What happens if the user passes a number larger than the length of the list to `moveDown`?

More [testing tips](#) on the course website’s Resources page!
compareToList

Write a method called compareToList that accepts two ArrayLists of integers list1 and list2 as parameters and compares the elements of the two lists, printing out the locations of common elements in each of the ArrayLists.

For example, if list1 contained [5, 6, 7, 8] and list2 contained [7, 5, 9, 0, 2], a call to compareToList(list1, list2) would produce output such as:

- 5 (list1 at 0, list2 at 1)
- 7 (list1 at 2, list2 at 0)
Practice : Think

Spend 1 min on your own thinking about how you would implement this method! (focus on *pseudocode*)

Write a method called `compareToList` that accepts two `ArrayList`s of integers `list1` and `list2` as parameters and compares the elements of the two lists, printing out the locations of common elements in each of the `ArrayList`s.

For example, if `list1` contained `[5, 6, 7, 8]` and `list2` contained `[7, 5, 9, 0, 2]`, a call to `compareToList(list1, list2)` would produce output such as:

- 5 (list1 at 0, list2 at 1)
- 7 (list1 at 2, list2 at 0)
Spend 2 min discussing about how you would implement this method with a neighbor! (focus on *pseudocode*)

Write a method called `compareToList` that accepts two `ArrayList`s of integers `list1` and `list2` as parameters and compares the elements of the two lists, printing out the locations of common elements in each of the `ArrayList`s.

For example, if `list1` contained `[5, 6, 7, 8]` and `list2` contained `[7, 5, 9, 0, 2]`, a call to `compareToList(list1, list2)` would produce output such as:

```
- 5 (list1 at 0, list2 at 3)
- 7 (list1 at 2, list2 at 0)
```
### ArrayList Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>add(type element)</td>
<td>Adds <em>element</em> to the end of the ArrayList</td>
</tr>
<tr>
<td>add(int index, type element)</td>
<td>Adds <em>element</em> to the specified <em>index</em> in the ArrayList</td>
</tr>
<tr>
<td>size()</td>
<td>Returns the number of elements in the ArrayList</td>
</tr>
<tr>
<td>contains(type element)</td>
<td>Returns true if <em>element</em> is contained in the ArrayList, false otherwise</td>
</tr>
<tr>
<td>get(int index)</td>
<td>Returns the element at <em>index</em> in the ArrayList</td>
</tr>
<tr>
<td>remove(int index)</td>
<td>Removes the element at <em>index</em> from the ArrayList and returns the removed element.</td>
</tr>
<tr>
<td>indexOf(type element)</td>
<td>Returns the index of <em>element</em> in the ArrayList; returns -1 if the <em>element</em> doesn’t exist in the ArrayList</td>
</tr>
<tr>
<td>set(int index, type element)</td>
<td>Sets the element at <em>index</em> to the given <em>element</em> and returns the old value</td>
</tr>
</tbody>
</table>
**topN**

Write a method called `topN` that accepts an `ArrayList` of characters `list` and an `int n` and returns a new `ArrayList` of characters that contains the first `n` elements of `list`.

For example, if `list` contained
`['g', 'u', 'm', 'b', 'a', 'l', 'l']`,
a call to `topN(list, 4)` would return an `ArrayList` containing `['g', 'u', 'm', 'b']`