BEFORE WE START

**Talk to your neighbors:**
Coffee or tea? Or something else?

Music: 122 24wi Lecture Tunes 🎶

**Instructor**
Miya Natsuhara and Joe Spaniac

**TAs**
- Aliya
- Alexander
- Ambika
- Andy
- Arkita
- Atharva
- Autumn
- Ayush
- Chaafen
- Chloe
- Claire
- Colin
- Colton
- Connor
- Elizabeth
- Hannah
- Helena
- Jessie
- Katharine
- Kavya
- Ken
- Kyle
- Logan
- Marcus
- Megana
- Mia
- Minh
- Nicolas
- Poojitha
- Rohini
- Ronald
- Rucha
- Sahej
- Shivani
- Smriti
- Steven
- Vinay
- Zane
Lecture Outline

• Announcements

• ArrayList Recap

• ArrayList Examples
Announcements

• Programming Assignment 0 due Thursday, January 18th at 11:59 PM
  - Reminder: 2 very helpful walkthrough videos were posted over the weekend
    (Thanks Joe and Colton!)

• Plan to release C0 grades and feedback tomorrow!
  - General grading turnaround is ~1 week
  - Resubmission Cycle 0 will also be released tomorrow
    - Due Tues Jan 23
    - Eligible assignment(s): C0

• Quiz 0 is tomorrow!
Quiz Tips!💡

• Hit run often!
  - Run, change, run, ...🔁

• Quizzes are graded based on tests that you do or don't pass – make sure your submitted code is able to run!
  - NOTE: You should be testing your own code – we don't provide test results!

• Quizzes are open-note, open-internet
  - BUT no generative AI (e.g., ChatGPT)
  - Have relevant PCMs, section problems, etc. open in other tabs!

• Grading rubrics on slides – think strategically!

• Time budgeting – quizzes are 45 minutes
Lecture Outline

• Announcements

• **ArrayList Recap**

• ArrayList Examples
ArrayList

ArrayLists are very similar to arrays
• Can hold multiple pieces of data (elements)
• Zero-based indexing
• Elements must all have the same type
  - ArrayLists can only hold Objects, so might need to use “wrapper” types: Integer, Double, Boolean, Character, etc.

But ArrayLists have dynamic length (so they can resize!)

4  8  16  23  42

list.add(2, 15);

4  8  15  16  23  42

list.size(): 5

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 <em>element</em> to the <strong>end</strong> of the ArrayList</td>
</tr>
<tr>
<td>add(int index, type element)</td>
<td>Adds <em>element</em> to the specified <strong>index</strong> 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 <strong>true</strong> if <em>element</em> is contained in the ArrayList, <strong>false</strong> otherwise</td>
</tr>
<tr>
<td>get(int index)</td>
<td>Returns the element at <strong>index</strong> in the ArrayList</td>
</tr>
<tr>
<td>remove(int index)</td>
<td>Removes the element at <strong>index</strong> 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 <strong>index</strong> to the given <em>element</em> and returns the old value</td>
</tr>
</tbody>
</table>
ArrayList Methods

• Whenever referring to “the ArrayList”, we are referring to the ArrayList we’re calling the method on!

```java
List<String> list = new ArrayList<String>();
list.add("hello");
list.add(0, "world");
list.indexOf("world"); // what is the output?
```

```java
String[] list = new String[2];
list[0] = "hello";
list[0] = "world";
list[1] = "hello";
//... indexOf?
```
Lecture Outline

• Announcements

• ArrayList Recap

• ArrayList Examples
Practice: Think

In-Class Activities

• **Goal**: Get you actively participating in your learning

• Typical Activity
  - Question is posed
  - **Think** (1 min): Think about the question on your own
  - **Pair** (2 min): Talk with your neighbor to discuss question
    - If you arrive at different conclusions, discuss your logic and figure out why you differ!
    - If you arrived at the same conclusion, discuss why the other answers might be wrong!
  - **Share** (1 min): We discuss the conclusions as a class

• During each of the **Think** and **Pair** stages, you will respond to the question via a sli.do poll
  - Not worth any points, just here to help you learn!
What is the best “plain English” description of this method?

```java
public static void method(ArrayList<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(ArrayList<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].
Practice: Think

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

A) \( \text{list}.\text{remove}(n); \)
   \( \text{list}.\text{add}(n); \)

B) \( \text{int element} = \text{list}.\text{remove}(n); \)
   \( \text{list}.\text{add}(n, \text{element}); \)

C) \( \text{list}.\text{add}(n); \)
   \( \text{list}.\text{remove}(n-1); \)

D) \( \text{int element} = \text{list}.\text{remove}(n); \)
   \( \text{list}.\text{add}(n+1, \text{element}); \)
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);`
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].
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 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 : Pair

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 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)
## ArrayList Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>add(type element)</code></td>
<td>Adds <code>element</code> to the end of the ArrayList</td>
</tr>
<tr>
<td><code>add(int index, type element)</code></td>
<td>Adds <code>element</code> to the specified <code>index</code> in the ArrayList</td>
</tr>
<tr>
<td><code>size()</code></td>
<td>Returns the number of elements in the ArrayList</td>
</tr>
<tr>
<td><code>contains(type element)</code></td>
<td>Returns true if <code>element</code> is contained in the ArrayList, false otherwise</td>
</tr>
<tr>
<td><code>get(int index)</code></td>
<td>Returns the element at <code>index</code> in the ArrayList</td>
</tr>
<tr>
<td><code>remove(int index)</code></td>
<td>Removes the element at <code>index</code> from the ArrayList and returns the removed element.</td>
</tr>
<tr>
<td><code>indexOf(type element)</code></td>
<td>Returns the index of <code>element</code> in the ArrayList; returns -1 if the <code>element</code> doesn’t exist in the ArrayList</td>
</tr>
<tr>
<td><code>set(int index, type element)</code></td>
<td>Sets the element at <code>index</code> to the given <code>element</code> 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 `['m', 'a', 't', 'i', 'l', 'd', 'a']`, a call to `topN(list, 4)` would return an `ArrayList` containing `[‘m’, ‘a’, ‘t’, ‘i’]`