BEFORE WE START

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

Music: Let's Call It Off - Drake

Instructor   **Elba Garza**

TAs

Abigail	Ambika	Arthur	Atharva
Autumn	Ayush	Chaafen	Chloë
Claire	Colin	Elizabeth	Helena
Jacob	Jasmine	Jaylyn	Kavya
Kevin	Kyle	Marcus	Megana
Mia	Poojitha	Rishi	Rohini
Rucha	Saivi	Shananda	Shivani
Shreya	Smriti	Steven	Zane

LEC 02

ArrayList

ArrayList

Questions during Class?
Raise hand or send here

sli.do   #cse122
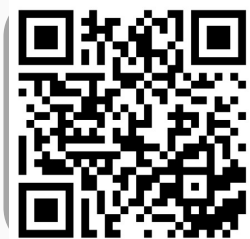
Lecture Outline

- **Announcements**

- ArrayList Recap

- ArrayList Examples

Announcements

- The IPL is open!
  - MGH 334
  - Schedule is on the course website; staffed by our awesome Tas!
  - Open 12:30 to 9:30 PM most days, but check the schedule…

- Programming Assignment 0 due Thursday, October 5th at 11:59 PM
  - Make sure to go to the "Final Submission" slide and submit!
  - Submit as many times as you'd like—we will only grade the latest submission made *before the deadline*

- Just joined CSE 122? That's okay; look at Ed & course website and catch up!
  - Freaking out that P0 is due tomorrow? It's ok! Resubmission cycles allow you to submit it later.

Getting Help

- Discussion Board
  - Feel free to make a public or private post on Ed
  - We encourage you to answer others' questions!
- Introductory Programming Lab (IPL Office Hours)
  - TAs can help you face to face in office hours
  - You can go to the IPL with <u>any</u> course question, not just assignments
- Section
  - Work through related problems, get to know your TA, who's here to support you!
- Your Peers
  - We encourage you to form study groups! Ed is a great place to do that.
- Email
  - We prefer that all content and logistics questions go on the Ed discussion board (even if you make them private. 600+ of you vs. 34 of us!
  - For serious personal circumstances, you can email Elba directly. It never hurts to email, but if it's a common logistics question, I may politely tell you to post on the discussion board.
  - Email inbox is crazy all through the quarter; I may take time to get to your email!

Reminders: Review Java Review

- Java Tutorial reviews all the relevant programming features you should be familiar with (even if you don't know them in Java.)

- Recording of the Java Review Session from Monday (October 2nd) are posted on the course calendar!
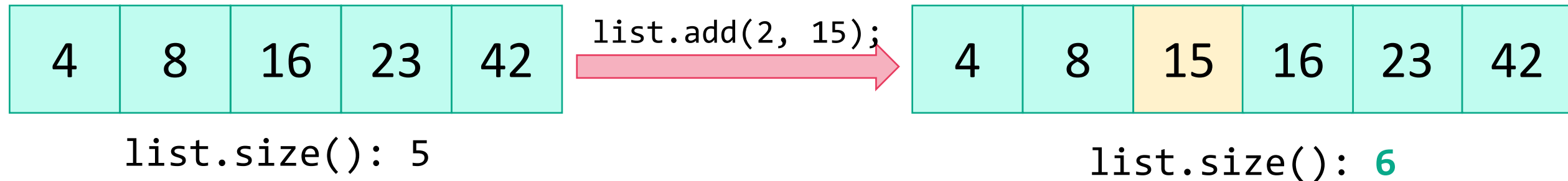
Lecture Outline

- 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 <u>same</u> type

- ArrayLists can <u>only</u> hold Objects, so might need to use "wrapper" types: `Integer, Double, Boolean, Character, etc.`

<u>But</u> 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

Method	Description
add(type *element*)	Adds *element* to the **end** of the ArrayList
add(int *index*, type *element*)	Adds *element* to the specified **index** in the ArrayList
size()	Returns the number of elements in the ArrayList
contains(type *element*)	Returns true if *element* is contained in the ArrayList, false otherwise
get(int *index*)	Returns the element at *index* in the ArrayList
remove(int *index*)	Removes the element at *index* from the ArrayList and returns the removed element.
indexOf(type *element*)	Returns the index of *element* in the ArrayList; returns -1 if the *element* doesn't exist in the ArrayList
set(int *index*, type *element*)	Sets the element at *index* to the given *element* and returns the old value

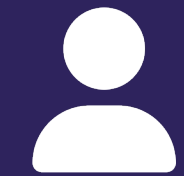# ArrayList Methods

- Whenever referring to "the ArrayList", we are referring to the ArrayList we're calling the method *on*!

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

```
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!

Practice : Think

sli.do   #cse122

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.

Practice : Pair

sli.do   #cse122

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));
    }
}
```

"Plain English" descriptions are what we are generally looking for in your method comments!

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.
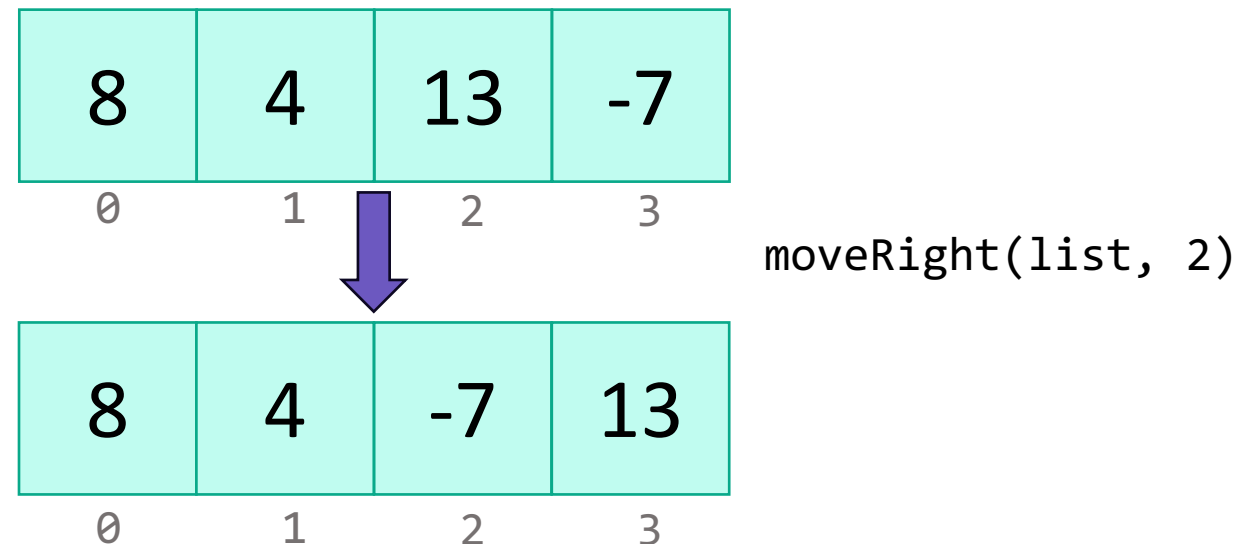
loadFromFile

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 <u>right</u> 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]`.

8	4	13	-7
0	1	2	3

`moveRight(list, 2)`

8	4	-7	13
0	1	2	3

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);
```

Practice : Pair

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 <u>right</u> 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]`.

`moveRight(list, 2)`
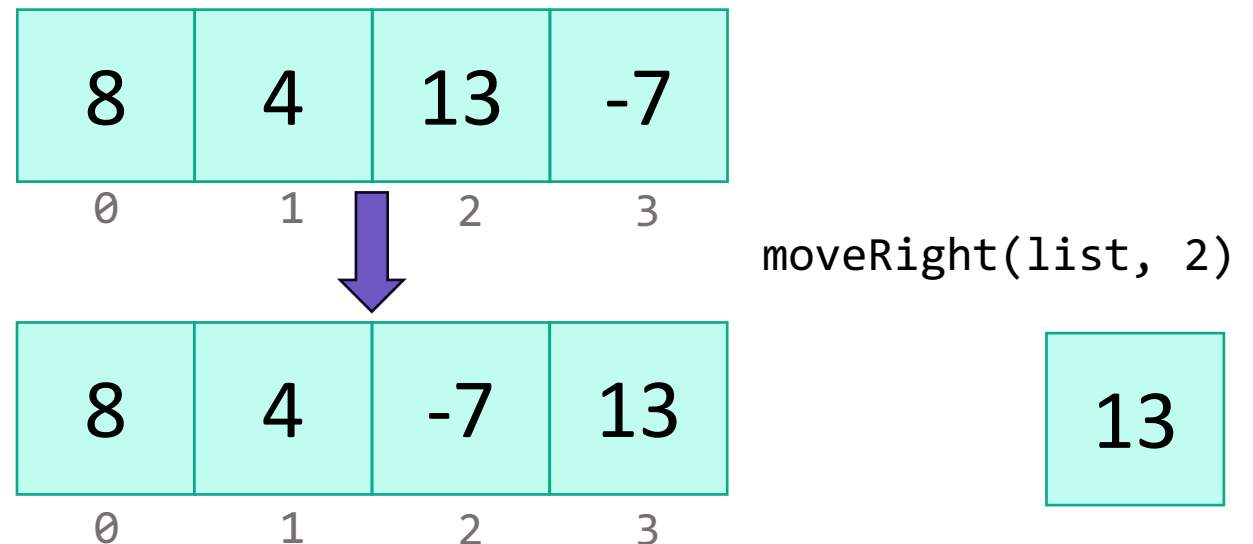
Edge Cases! (And Testing)

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)
```
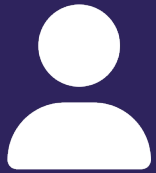
Practice : Think

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

sli.do   #cse122

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)
```

UNIVERSITY *of* WASHINGTON

ArrayList Methods

Method	Description
add(type *element*)	Adds *element* to the **end** of the ArrayList
add(int *index*, type *element*)	Adds *element* to the specified **index** in the ArrayList
size()	Returns the number of elements in the ArrayList
contains(type *element*)	Returns true if *element* is contained in the ArrayList, false otherwise
get(int *index*)	Returns the element at *index* in the ArrayList
remove(int *index*)	Removes the element at *index* from the ArrayList and returns the removed element.
indexOf(type *element*)	Returns the index of *element* in the ArrayList; returns -1 if the *element* doesn't exist in the ArrayList
set(int *index*, type *element*)	Sets the element at *index* to the given *element* and returns the old value

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']`