

LEC 02

# ArrayList

# ArrayList

BEFORE WE START

*Talk to your neighbors:  
Dogs or cats?*

Music: [Miya's 23wi CSE 122 Playlist](#)

Instructor **Miya Natsuhara**

TAs

Ayush  
Connor  
Poojitha  
Andrew A  
Andrew C  
Jasmine  
Darel  
Gabe  
Karen  
Colton

Atharva  
Julia  
Megana  
Joey  
Eesha  
Lilian  
Thomas  
Leon  
Melissa  
Audrey

Ernie  
Di  
Logan  
Shivani  
Michelle  
Steven  
Kevin  
Ken  
Vivek  
Autumn

Ambika  
Elizabeth  
Joe  
Jin  
Ben  
Evelyn  
Kent


Questions during Class?

Raise hand or send here

sli.do #cse122



# Lecture Outline

- **Announcements** 
- ArrayList Recap
- ArrayList Examples

# Announcements

- The IPL is open and in full swing!
  - MGH 334
  - Schedule on the course website, staffed by our awesome TAs!
  - Open 12:30-9:30 most days!
- Programming Assignment 0 due Thursday, Jan 12
  - Make sure to hit “Mark” to submit!
  - Mark as many times as you want – we will grade the latest submission made before the deadline

# Getting Help

- Discussion Board
  - Feel free to make a public or private post on Ed
  - We encourage you to answer other peoples' questions! A great way to learn
- Introductory Programming Lab (Office Hours)
  - TAs can help you face to face in office hours, and look at your code
  - You can go to the IPL with **any** course questions, not just assignments
- Section
  - Work through related problems, get to know your TA who is 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 logistic questions go on the Ed discussion board (even if you make them private). 700+ of you >>> 38 of us!
  - For serious personal circumstances, you can email Miya directly. It never hurts to email me, but if it's a common logistic question, I may politely tell you to post on the discussion board.

mnats@cs.washington.edu

# Reminders: Review Java Syntax

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

Recordings of the Java Review Sessions from Monday (1/9) are posted on the course calendar.

# Lecture Outline

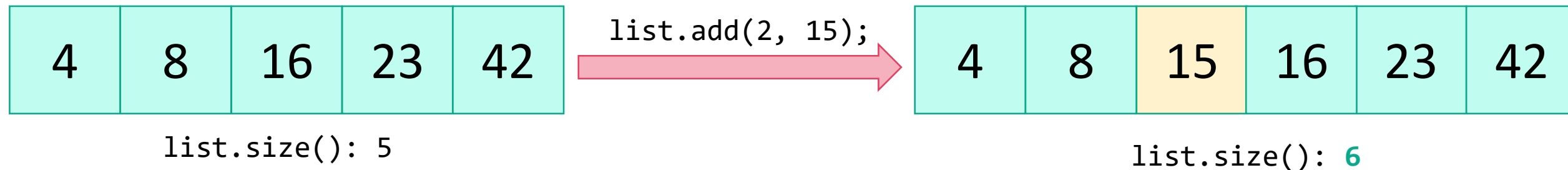
- Announcements
- **ArrayList Recap** 
- ArrayList Examples

# ArrayList

ArrayLists are very similar to arrays

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

**BUT** ArrayLists have dynamic length (so they can resize)



# ArrayList Methods

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



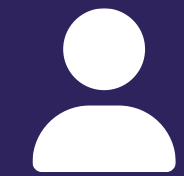
# ArrayList Methods

- 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("hello");
```

# Lecture Outline

- Announcements
- ArrayList Recap
- **ArrayList Examples** ◀



# Practice : Think

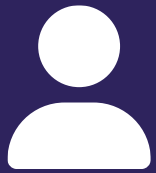


sli.do

#cse122

## 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?

```
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?

```
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

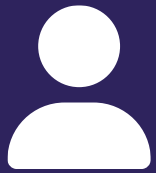
Write a method called `loadFromFile` that accepts a `Scanner` as a parameter and returns a new `ArrayList` of `Strings` where each element of the `ArrayList` is a line from the `Scanner`, matching the order of the `Scanner`'s contents.

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 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



sli.do #cse122

**What ArrayList methods (and in what order) could we use to implement the moveDown 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



sli.do #cse122

## What ArrayList methods (and in what order) could we use to implement the moveDown 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`?

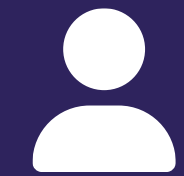
More [testing tips](#) on the course website's Resources page!

# 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

[sli.do](https://sli.do)

#cse122

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

# ArrayList Methods

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