LEC 04

ArrayList

ArrayList

Questions during Class?

Raise hand or send here

sli.do #cse122

BEFORE WE START

Talk to your neighbors:

Coffee or tea? Or something else?

Music: 122 24au Lecture Tunes

Instructors: Miya Natsuhara and Elba Garza

TAs:

Ayush	Heon	Harshitha	Aishah
Andrew	Izak	Marcus	Ben
Logan	Colin	Carson	Ivory
Kyle	Jessica	Jack	Cady
Maggie	Shivani	Connor	Diya
Nicole H	Ken	Cora	Katharine
Caleb	Mia	Hannah	
Nicole W	Ashley	Leo	
Jacob	Chaafen	Anya	

Lecture Outline

- Announcements
- ArrayList Recap
- ArrayList Examples

Announcements

- Programming Assignment 0 due Thursday, Oct 10th at 11:59 PM
- Plan to release C0 grades and feedback tomorrow!
  - General grading turnaround is ~1 week
  - Resubmission Cycle 0 will also be released tomorrow
    - Due Tues Oct 15
    - Eligible assignment(s): C0
- Quiz 0 is next Tuesday (Oct 15)!
  - Check the Ed post for instructions and logistics
  - Practice Quiz to be released later today (and solutions posted over the weekend)

Lecture Outline

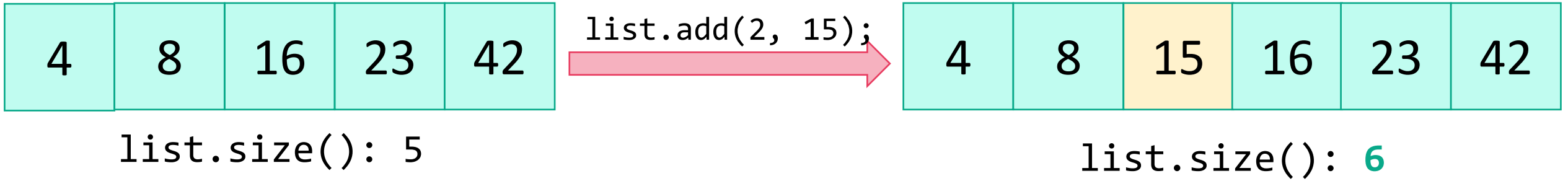
- Announcements
- ArrayList Recap
- ArrayList Examples

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!)



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 Usage

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

```
List<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

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

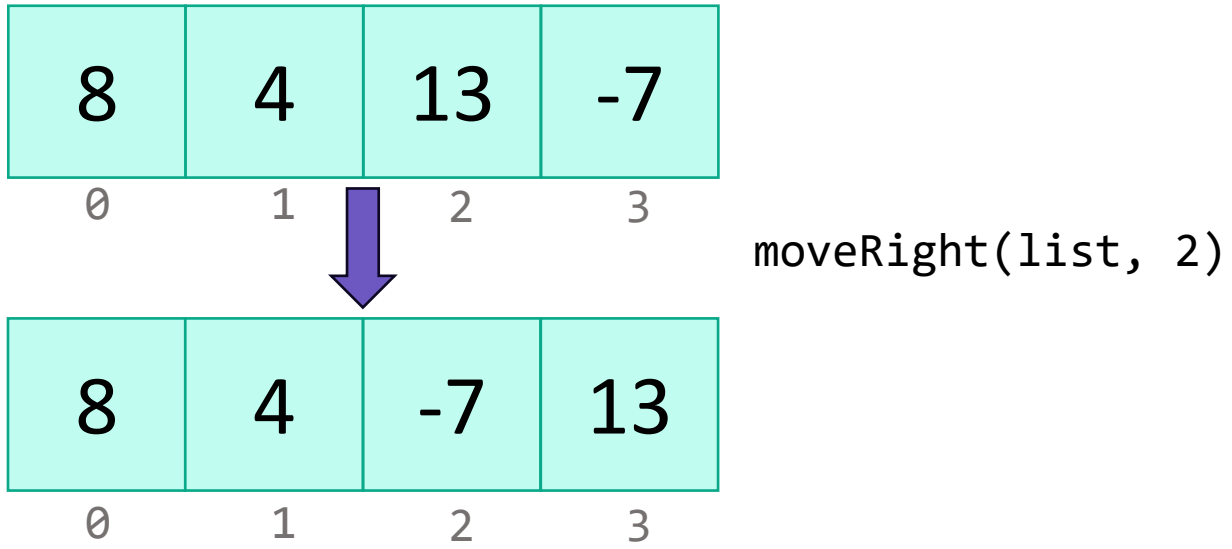
Write a method called `loadFromFile` that accepts a `Scanner` as a parameter and returns a new `ArrayList` of `Strings` where each element of the `ArrayList` is a line from the `Scanner`, matching the order of the `Scanner`'s contents.

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]`.



Practice : Think

sli.do #cse122

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

sli.do #cse122

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);
```

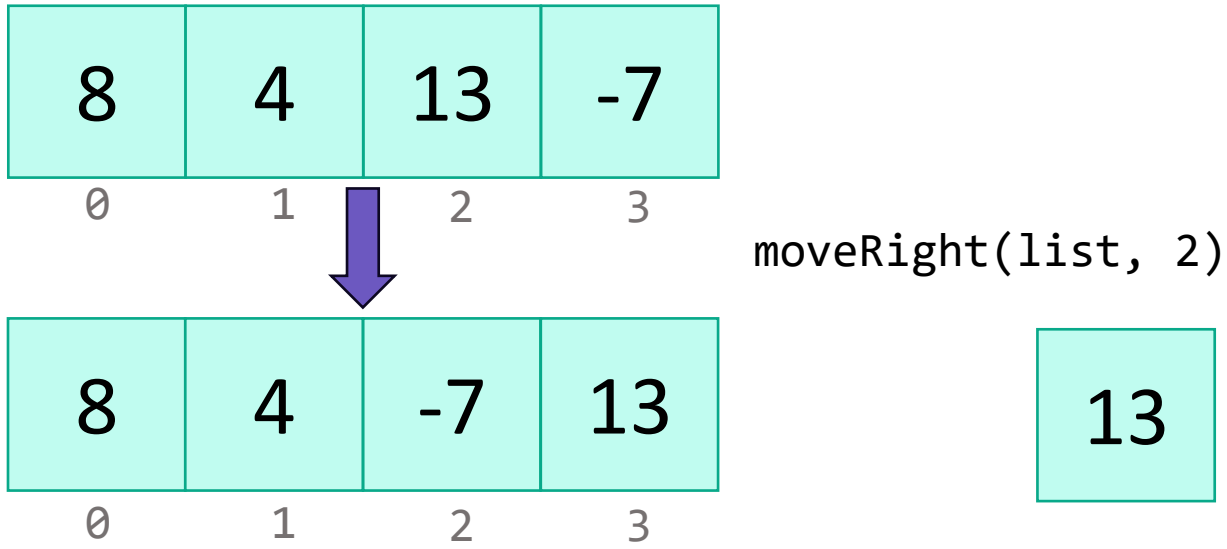
C)
```
list.add(n);
list.remove(n-1);
```

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

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 1)
- 7 (list1 at 2, list2 at 0)
```

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']`