LEC 04

ArrayList Applications

Questions during Class?

Raise hand or send here

sli.do #cse122

BEFORE WE START

Slido vote & chat with neighbors:

What is your favorite part of spring?

Music: 122 26sp Lecture Jams

Instructor: Elba Garza

TAs:

David	Caleb	Cole	Yang
William	Neha	Blake R.	Cady
Dani	Wesley	Carson	Diya
Rohan	Isis	Sushma	
Andrew	Colin	Connor	
Ava	Naomi	Mahima	
Shreyank	Hanna	Nicolae	
Nicole	Blake P.	Ivory	

Lecture Outline (1/6)

- Announcements
- moveRight
- compareToList
- topN
- addAll
- ArrayList Extended Application

Announcements/Reminders

- First quiz in section tomorrow, Apr 16th
  - Practice Quiz 0 released!
- Programming Assignment 0 (P0) due tomorrow!
- Creative Assignment 1 (C1) out Friday!
  - Focused on `ArrayLists`
  - Due next Thursday, Apr 23rd by 11:59pm PT
- Reminder: Section participation in 11+ sections → Extra resub!
  - No need to do anything; TAs will track on Gradescope

Lecture Outline (2/6)

- Announcements
- **moveRight** 
- compareToList
- topN
- addAll
- ArrayList Extended Application

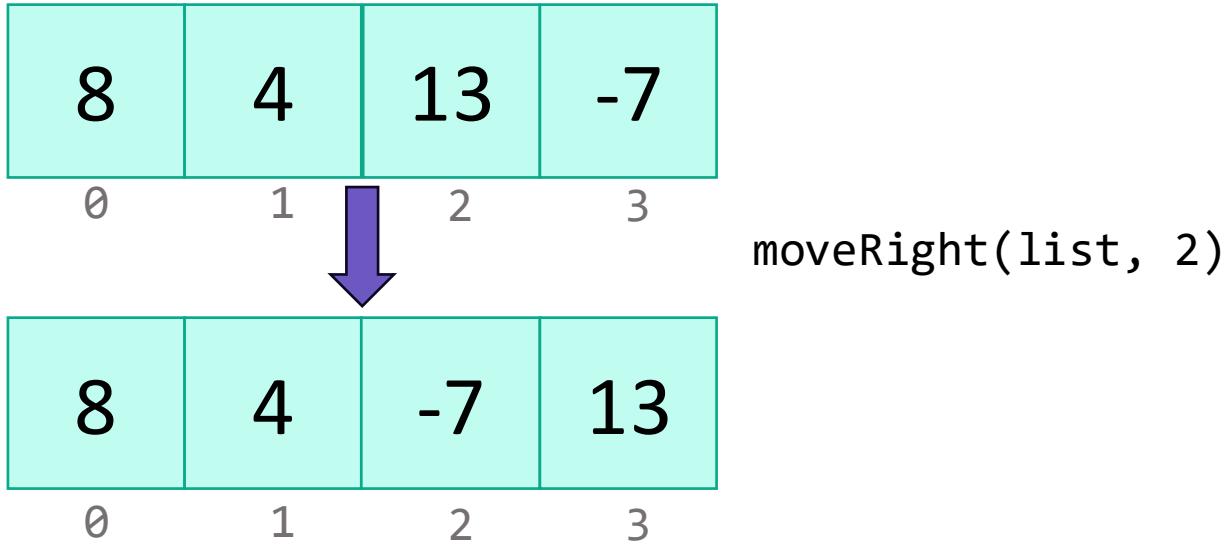
ArrayList Methods

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

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);
```

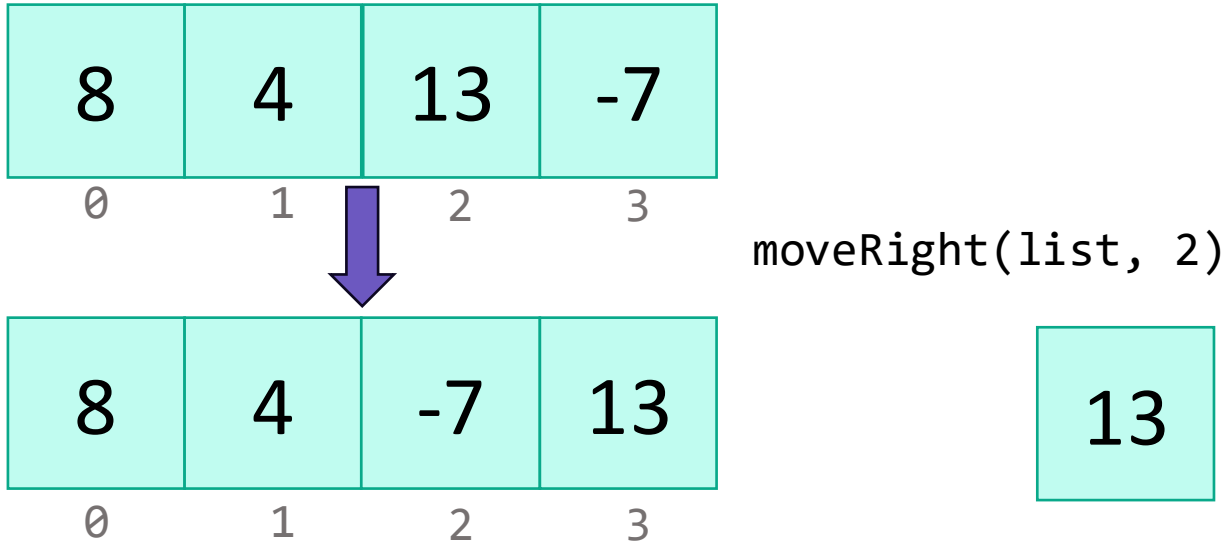
C)
```
list.add(n);
list.remove(n-1);
```

D)
```
int element = list.remove(n);
list.add(n+1, element);
```

moveRight

Write a method called `moveRight` that accepts a `List` 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 `moveRight`?

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

More testing tips on the course website's Resources page!

Lecture Outline (3/6)

- Announcements
- moveRight
- compareToList
- topN
- addAll
- ArrayList Extended Application

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)
```

Lecture Outline (4/6)

- Announcements
- moveRight
- compareToList
- **topN**
- addAll
- ArrayList Extended Application

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']`

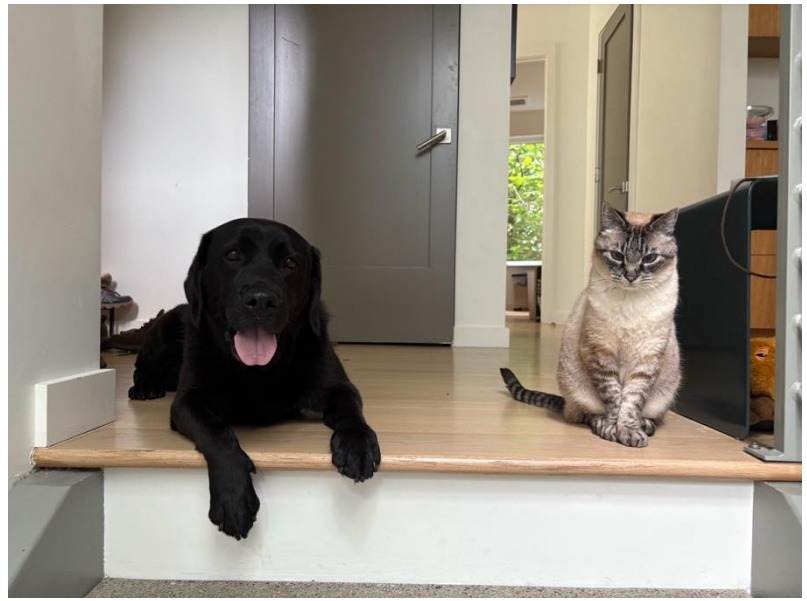
Lecture Outline (5/6)

- Announcements
- moveRight
- compareToList
- topN
- addAll
- ArrayList Extended Application

addAll

Write a method called `addAll` that accepts two `ArrayLists` of Characters, `list1` and `list2` , and an integer `location` as parameters and inserts **all** of the elements from `list2` into `list1` at the specified `location`.





Lecture Outline (6/6)

- Announcements
- moveRight
- compareToList
- topN
- addAll
- **ArrayList Extended Application**

Bakery Favorites

We will write a program called `BakeryFavorites.java` that manages a list of favorite bakeries for a user (using an `ArrayList`) and allows the user to perform various different operations on their stored list of favorite bakeries.

Key skills used:

- User Interaction (UI) loop
- Iterative development strategies
- Functional decomposition
- Practice with `ArrayList` methods!

Bakery Favorites: Operations

- Load a list of favorites in from a file provided by the user.
- Compare the stored list of favorites to another list of favorites provided by the user in another file.
- Report the top *n* favorites according to the list, where the user can specify *n*.
- Move a specific favorite down in the list.
- Add a list of favorites in a user-provided file to the stored list of favorites at a specified location.
- Save the current list of favorites to a file provided by the user.

Bakery Favorites: Development Strategy

- Set up the main scaffold code
- Menu loop
- Develop each operation, one at a time

You'll see a similar development strategy in Creative Project 1's specification — we recommend you follow it!

Bakery Favorites: Operations

- Load a list of favorites in from a file provided by the user.
- Compare the stored list of favorites to another list of favorites provided by the user in another file.
- Report the top *n* favorites, where the user can specify *n*.
- Move [illegible] in the list.
- Add a [illegible]tes in a user-provided file to the stored list of favorites at a specified location.
- Save the current list of favorites to a file provided by the user.

ALREADY DONE!