LEC 05

ArrayList Applications

Questions during Class?

Raise hand or send here

sli.do #cse122

BEFORE WE START

Talk to your neighbors:

Any weekend plans?

Music: 122 25sp Lecture Tunes 🌸

Instructor: Brett Wortzman and Adrian Salguero

TAs:

Andrew
Anya
Brittan
Carson
Christopher
Colin
Dalton
Daniel
Diya
Elizabeth
Ivory
Jack
Jacob
Ken
Kyle
Leo
Logan
Mahima
Medha
Minh
Nicole
Samuel
Shivani
Sreshta
Steven
Yang

Lecture Outline

- Announcements ◀
- Warm Up
- ArrayList Extended Application

Announcements

- C0 grades and R0 out yesterday
  - Now that you have your first set of grades, review the Course Grades section of the syllabus to understand how they factor into your grade at the end of the quarter!
  - Grade Checker spreadsheet also linked from the syllabus to help track your grades throughout the quarter.
  - See Resubmission page and Ed post for R0 logistics
- Creative Assignment 1 (C1) out later today!
  - Focused on `ArrayLists`
  - Due next Thursday, Apr 24th by 11:59 PM
- First quiz in section on Tuesday Apr 22nd
  - Practice Quiz 0 and Reference Sheet will be released later tonight!

Lecture Outline

- Announcements
- **Warm Up** ◀
- ArrayList Extended Application

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!

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

addAll

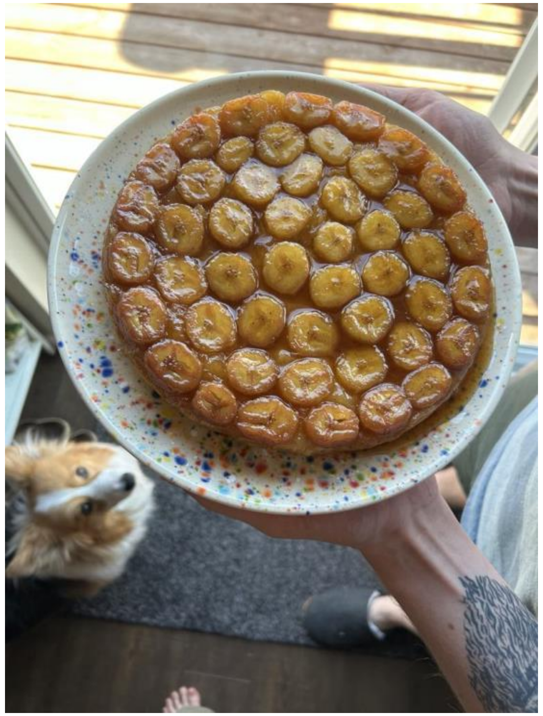
Write a method called `addAll` that accepts two `ArrayLists` of Characters, `list1` and `list2` , and an integer `location` as parameters and inserts **all** of the elements from `list2` into `list1` at the specified `location`.

Lecture Outline

- Announcements
- Warm Up
- **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 pr the user.
- Compare the stored list of f of favorites provided by the us
- Report the to list, where the user can specify *n*.
- Move a specifi ite 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.

ALREADY DONE!

See In-Class 4