CE ArrayList | LinkedList

LATE 10/10 Points

| 9/19/2022

Attempt 1

REVIEW FEEDBACK
10/30/2022

Attempt 1 Score: **10/10**

Add Comment

Unlimited Attempts Allowed

Details

Fundamentals

CE: ArrayList | LinkedList

Learning Objectives

- Determine the big O for ArrayList and LinkedList operations.
- Implement methods for a linked data structure.

Overview

This CE consists of two parts.
In Part 1 you will determine the big O for ArrayList and LinkedList operations.
In Part 2 you will implement some methods for a linked data structure.

Previous (https://slcc.instructure.com/courses/817632/modules/items/18752948)

Try Again

Next (https://slcc.instructure.com/courses/817632/modules/items/18752950)

Part 1

- The table below lists various operations of an **ArrayList**.
  Determine the big O of the first three operations.

Operation	Cost as O( )
read (anywhere in the array)	
add/remove (at the logical end of the array)	
add/remove (in the interior of the array)	
resize	
find by position	
find by target (element)	

When you are done, compare your work with a ***solution***

- We continue working with an **ArrayList**.
  Determine the big O of the next three operations.
  "Find the element on index 41" is an example of 'find by position'.
  "Find the element "Seattle" is an example of 'find by target'.

Operation	Cost as O( )
read (anywhere in the array)	
add/remove (at the logical end of the array)	
add/remove (in the interior of the array)	
resize	
find by position	
find by target (element)	

When you are done, compare your work with the ***solution***

- The table below lists various operations of a **LinkedList**.

Previous (https://slcc.instructure.com/courses/817632/modules/items/18752948)

Try Again

Next (https://slcc.instructure.com/courses/817632/modules/items/18752950)

Operation	Slingly Linked list	Doubly Linked list
read (anywhere in the linked list)		
add/remove (at the head)		
add/remove (at the tail)		
add/remove (in the interior of the structure)		
resize		
find by position		
find by element		

When you determine the big-O of reading, adding, or removing, include the effort it takes to access the specified element that should be read, added, or removed.
Linked lists can have a single connection to the next element (singly-linked lists) or they can connect both to the previous and the next elements (doubly-linked list).
Consider both types of linked lists separately as you determine the big O of the first two operations.
When you are done, compare your work with the ***solution***

- Determine the big O of the third and fourth operations. Again, consider both types of linked lists separately.

Operation	Slingly Linked list	Doubly Linked list
read (anywhere in the linked list)		
add/remove (at the head)		
add/remove (at the tail)		
add/remove (in the interior of the structure)		
resize		
find by position		
find by element		

When you are done, compare your work with the ***solution***

- Determine the big O of the last three operations.

Previous (https://slcc.instructure.com/courses/817632/modules/items/18752948)

Try Again

Next (https://slcc.instructure.com/courses/817632/modules/items/18752950)

Operation	Slingly Linked list	Doubly Linked list
read (anywhere in the linked list)		
add/remove (at the head)		
add/remove (at the tail)		
add/remove (in the interior of the structure)		
resize		
find by position		
find by element		

When you are done, compare your work with the ***solution***

Part 2

In this second part of the class exercise, you will implement some methods for a linked data structure.

- Create a package called **ceLinked**
  Add a class **WordList** and use the provided code as a starter project.
  WordList.java (https://slcc.instructure.com/courses/817632/files/135713595/download?wrap=1) (https://slcc.instructure.com/courses/817632/files/135713595/download?download_frd=1)

- Run the program.
  At this point, the output should look like this:
  ```
  size: 0
  list: ant bat cow dog
  ```

- Take a few minutes to familiarize yourself with the code.
  Notice that there are four areas that still need work. They are marked with comments that tell you what to do and in which order.
  - **TODO 1**
    The first area that needs to be completed is in the main method. Instructions are provided in the form of inline comments.
    This challenge is a good opportunity to review the use of a **ternary operator (https://www.baeldung.com/java-ternary-operator)** .

Previous (https://slcc.instructure.com/courses/817632/modules/items/18752948) Try Again Next (https://slcc.instructure.com/courses/817632/modules/items/18752950)

Write **test code**, that matches the output shown in the 'Expected Output' section.
In order to test prepend, I called it twice. I prepended 'ape' right after TODO 1 when the list was still empty, and I prepended 'auk' right after the list was printed the first time.

Important: When I add or remove elements from the original list, the output needs to change accordingly.

Expected Output

```
size: 0
TODO 1: The list is empty.

TODO 2: prepend 'ape'
list: ape ant bat cow dog

TODO 2: prepend 'auk'
list: auk ape ant bat cow dog
size: 6

TODO 3: indexOf
Index of dog: 5
Index of auk: 0
Index of yak: -1

TODO 4: contains
cow is included in the list.
yak is not included in the list.

list: auk ape ant bat cow dog
size: 6
```

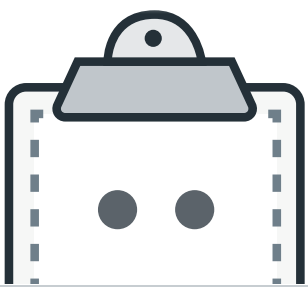
Previous (https://slcc.instructure.com/courses/817632/modules/items/18752948)

Try Again

Next (https://slcc.instructure.com/courses/817632/modules/items/18752950)

(https://slcc.instructure.com/courses/817632/pages/guidelines-for-ce-recordings) .
The video should be **25-50 seconds** long.
Post the video.

(https://slcc.instructure.com/courses/817632/modules/items/18752939)
(https://slcc.instructure.com/courses/817632/modules/items/18752940)
(https://slcc.instructure.com/courses/817632/modules/items/18752941)
(https://slcc.instructure.com/courses/817632/modules/items/18752942)
(https://slcc.instructure.com/courses/817632/modules/items/18752943)
(https://slcc.instructure.com/api/v1/courses/817632/module_item_redirect/18752945)
(https://slcc.instructure.com/api/v1/courses/817632/module_item_redirect/18752946)
(https://slcc.instructure.com/courses/817632/modules/items/18752948)
(https://slcc.instructure.com/courses/817632/modules/items/18752949)
(https://slcc.instructure.com/courses/817632/modules/items/18752950)
(https://slcc.instructure.com/courses/817632/modules/items/18752951)
(https://slcc.instructure.com/courses/817632/modules/items/18752953)



Previous
(https://slcc.instructure.com/courses/817632/modules/items/18752948)
Try Again
Next
(https://slcc.instructure.com/courses/817632/modules/items/18752950)

Preview Unavailable

2022-10-30 19-49-57.mp4

Download

(https://slcc.instructure.com/files/139031282/download?download_frd=1&verifier=i7dFqAyiduMq1ugwnFJ4FYhtZDNSeVJaj7Niu1O3)

Previous (https://slcc.instructure.com/courses/817632/modules/items/18752948)

Try Again

Next (https://slcc.instructure.com/courses/817632/modules/items/18752950)