CS2852 Lab 1

Name: Cherise Malisa

Date: 10/03/2021

Instructor: Dr. Sohoni

# Part 1

## What is the advantage of using the List interface and avoiding direct references to the ArrayList and LinkedList classes?

Your answer here

The methods used are then not limited to a specific type of list but rather the one passed in and can be used for both linked lists and arraylists.

## Why do you think we created the ItemRequest class?

Your answer here

To manage the requests since they come as itemIds and userIds, and enable them to be added to a list easily.

## Why do you think we created the LibraryInventory class? If we wanted to make the simulation more realistic, how could we change this class?

Your answer here

# Part 2

## Create a table of your benchmark results.

Your answer here

|  |  |  |
| --- | --- | --- |
|  | ArrayList (ms) | LinkedList (ms) |
| Add | 5.99 | 3.98 |
| Fulfillable requests | 10.63 | 41.33 |
| Unfulfillable requests | 65.69 | 119.84 |

## The requestItem () method adds the object to the end or back of the list. Did you notice a significant difference in run time between the two types of lists?

Your answer here

Not really, the linked list took less time to add items to the end of the list. Whereas, the array list took a longer time. However, the difference is not a massive one.

## For the benchmark of the nextFulfillableRequest(true) method in which all items are fulfillable, the method will always remove and return the first item from the front of the list. Did you notice a significant difference in performance between the two types of lists?

Your answer here

The linked took a significantly longer amount of time to remove and return the item.

## For the benchmark of the nextFulfillableRequest(false) method in which no items are fulfillable, the method will search through all items in the list, find no matches, and return null. Did you notice a significant difference in performance between the two types of lists? (Note that depending on how you implemented the method, your benchmark results may vary.)

Your answer here

Yes, the array list searched through significantly faster than the linked list which took a much longer time.

## Thinking back to your startup, do you think it will be more common for item requests to be fulfillable or not? Based on that, which type of list would you choose and why?

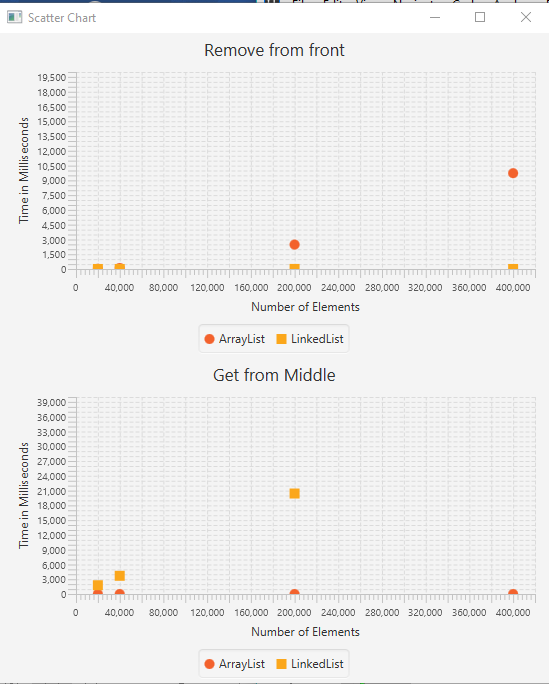
Your answer here

It is more common for the requests to be fulfillable, and hence I would choose array lists to for item requests.

# Part 3

## Paste screenshots of the two graphs created by your GUI program.

Your answer here



## Is the performance of removeFromFront() similar for the two lists?

Your answer here

To a certain extent. The first two values where really close the point where the first points are overlapping. However, as the range number of elements got larger the arraylist the took a longer time to complete the task.

## Is the performance of getMiddleBenchmark() similar for the two lists?

Your answer here

Yes, it is but just like for remove from front method only for the first to values. And when more elements are added to the list the arraylist takes longer to complete the task than the linked list. And its last value is not even within range of the chart and show it doesn’t appear.

## How do the differences in performance of the removeFrontBenchmark() and getMiddleBenchmark() methods for the two list types explain some of the performance differences of the WaitingList.nextRequest() method?

Your answer here

The array list takes more time to do these tasks (remove from front and remove from middle) because there is a lot of shifting of elements that has to be done in order to maintain the index of numbers in the array list, whilst with the linked list no shifting is required and hence it takes less time for these tasks to be fulfilled and this makes it more effective to carry out. On the other hand, when it comes to the performance on waiting list the array becomes more efficient. As it took less time for both nextFulfillableRequest(true) and nextFulfillableRequest(false) . where as the linked list took a significantly longer time to complete the tasks.