**GTU Department of Computer Engineering**

**CSE 222/505 – Spring 2023**

**Homework 3 Report**

**Barış Batuhan Bolat**

**210104004029**

**Running Times**

* Time complexities of all implementations is in the end of this chapter with table representation.

1. **Basic Array**

* “inFollowing” method takes O(1) time and two array resize operation takes O(2n) but for big-O-operation constant number cannot be considered so in total “followAccount” methods time complexity is **O(n)**.

**metin içeren bir resim

Açıklama otomatik olarak oluşturuldu**

* All get method’s time complexities are O(1) and two array resize operation takes O(2n) time but for big-O-notation constant number cannot be considired so “sendMessage “ method’s time complexity is **O(n).**

**metin içeren bir resim

Açıklama otomatik olarak oluşturuldu**

* metin içeren bir resim

  Açıklama otomatik olarak oluşturulduAll get method’s time complexities are O(1) and array resize operation is O(n) so “sharePost” method’s time complexity is **O(n).**
* All get method’s time complexities are O(1) and array resize operation is O(n) so “likePost” method’s time complexity is **O(n).**

**metin içeren bir resim

Açıklama otomatik olarak oluşturuldu**

* All get method’s time complexities are O(1) and array resize operation is O(n) so “commentPost” method’s time complexity is **O(n).**

**metin içeren bir resim

Açıklama otomatik olarak oluşturuldu**

* All get method’s time complexities are O(1) and array resize operation is O(n) so “block” method’s time complexity is **O(n).**

**metin içeren bir resim

Açıklama otomatik olarak oluşturuldu**

1. **ArrayList – LinkedList – LDLinkedList**

LDLinkedList problem solution approach is end of this chapter.

* **ArrayList :** “add” method’s time complexity is O(n) and “inFollowing” method’s time complexity is O(n) so “followAccount“ method’s time complexity is **O(n).**
* **LinkedList :** “add” method’s time complexity is O(1) and “inFollowing” method’s time complexity is O(n) so “followAccount“ method’s time complexity is **O(n).**
* **LDLinkedList :** “add” method’s time complexity is O(1) and “inFollowing” method’s time complexity is O(n) so “followAccount“ method’s time complexity is **O(n).**

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

* **ArrayList :** “add” and “remove” methods time complexities are O(n) and “inFollowing ” method’s time complexity is O(n) so “unFollowAccount“ method’s time complexity is **O(n).**
* **LinkedList :** “add” method’s time complexity is O(1), “remove” method’s time complexity is O(n) and “inFollowing ” method’s time complexity is O(n) so “unFollowAccount“ method’s time complexity is **O(n).**
* metin içeren bir resim

  Açıklama otomatik olarak oluşturuldu**LDLinkedList :** “add” method’s time complexity is O(1) , “inFollowing” method’s time complexity is O(n) and “remove” method’s time complexity is normally O(n) after two deletion it becomes O(2n) but for big -O-notation it still be O(n) so “unFollowAccount“ method’s time complexity is **O(n).**
* **ArrayList :** “add” method’s time complexity is O(n) ,”inFollowing” and “isBlocked” mehod’s time complexities are O(n) so “sendMessage“ method’s time complexity is **O(n).**
* **LinkedList :** “add” method’s time complexity is O(1) ,”inFollowing” and “isBlocked” mehod’s time complexities are O(n) so “sendMessage“ method’s time complexity is **O(n).**
* **LDLinkedList :** “add” method’s time complexity is O(1) ,”inFollowing” and “isBlocked” mehod’s time complexities are O(n) so “sendMessage“ method’s time complexity is **O(n).**

**metin içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**-------------------------------------------------------------------------------**

* **ArrayList :** “add” method’s time complexity is O(n) so “sharePost“ method’s time complexity is **O(n).**
* **LinkedList :** “add” method’s time complexity is O(1) so “sharePost“ method’s time complexity is **O(1).**
* **LDLinkedList :** “add” method’s time complexity is O(1) so “sharePost“ method’s time complexity is **O(1).**

**metin içeren bir resim

Açıklama otomatik olarak oluşturuldu**

* **ArrayList :** “add” method’s time complexity is O(n) and “isLiking” method’s time complexity is O(n) so “likePost“ method’s time complexity is **O(n).**
* **LinkedList :** “add” method’s time complexity is O(1) and “isLiking” method’s time complexity is O(n) so “likePost “method’s time complexity is **O(n).**
* **LDLinkedList :** “add” method’s time complexity is O(1) and “isLiking” method’s time complexity is O(n) so “likePost “method’s time complexity is **O(n)**

**metin içeren bir resim

Açıklama otomatik olarak oluşturuldu**

* **ArrayList :** “add” method’s time complexity is O(n) , for loop with “get” method’s time complexity is O(n) and “remove” method’s time complexity is O(n) so “unlikePost“ time complexity is **O(n).**
* **LinkedList :** “add” method’s time complexity is O(n) , for loop with “get” method’s time complexity is O(n2) and “remove” method’s time complexity is O(n) so “unlikePost“ time complexity is **O(n2).**
* **LDLinkedList :** “add” method’s time complexity is O(n) , for loop with “get” method’s time complexity is O(n2) and “remove” method’s time complexity is normally O(n) after two deletion it becomes O(2n) but for big-O-notation it still be O(n) so “unlikePost “ time complexity is **O(n2).**
* **ArrayList :** “add” method’s time complexity is O(n) so “commentPost“ method’s time complexity is **O(n).**
* **LinkedList :** “add” method’s time complexity is O(1) so “commentPost“ method’s time complexity is **O(1).**
* **LDLinkedList :** “add” method’s time complexity is O(1) so “commentPost“ method’s time complexity is **O(1).**

**metin içeren bir resim

Açıklama otomatik olarak oluşturuldu**

* **ArrayList :** “add” method’s time complexity is O(n) , for loop with “get” method’s time complexity is O(n2) and “remove” method’s time complexity is O(n) so “unlikeComment“ time complexity is **O(n2).**
* **LinkedList :** “add” method’s time complexity is O(n) , for loop with “get” method’s time complexity is O(n2) and “remove” method’s time complexity is O(n) so “unlikeComment“ time complexity is **O(n2).**
* **LDLinkedList :** “add” method’s time complexity is O(n) , for loop with “get” method’s time complexity is O(n2) and “remove” method’s time complexity is normally O(n) after two deletion it becomes O(2n) but for big-O-notation it still be O(n) so “unlikeComment“ time complexity is **O(n2).**
* **ArrayList :** “add” method’s time complexity is O(n) and “unfollowAccount” method’s time complexity is O(n) as I said above so “block“ method’s time complexity is **O(n).**
* **LinkedList :** “add” method’s time complexity is O(1) and “unfollowAccount” method’s time complexity is O(n) as I said above so “block“ method’s time complexity is **O(n).**
* **metin içeren bir resim

  Açıklama otomatik olarak oluşturulduLDLinkedList :** “add” method’s time complexity is O(1) and “unfollowAccount” method’s time complexity is O(n) as I said above so “block“ method’s time complexity is **O(n).**
* **ArrayList :** “add” method’s time complexity is O(1) and “remove” method’s time complexity is O(n) so “unblock“ time complexity is **O(n).**
* **LinkedList :** “add” method’s time complexity is O(n) and “remove” method’s time complexity is O(n) so “unblock“ time complexity is **O(n).**
* **LDLinkedList :** “add” method’s time complexity is O(n) and “remove” method’s time complexity is normally O(n) after two deletion it becomes O(2n) but for big -O-notation it still be O(n) so “unBlock“ method’s time complexity is **O(n).**

****

**Experimental Running Time Table**

* I measured this values by using “System.nanoTime()” method

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Implementation Type** | **Scenario 1** | **Scenario 2** | **Scenario 3** | **Scenario 4** |
| **Basic Array Structure (HW1)** | 0.044 | 0.0042 | 5.12x10-4 | Not available |
| **Array List Structure (a)** | 0.040 | 0.0035 | 5.42x10-4 | 0.0038 |
| **Linked List Structure (b)** | 0.042 | 0.0037 | 8.20x10-4 | 0.0026 |
| **LD Linked List Structure (c)** | 0.040 | 0.0027 | 5.30x10-4 | 0.0031 |

**Time Complexity Analysis Table**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Implementation Type** | **Follow** | **Unfollow** | **Block** | **Unblock** | **Like** | **Unlike** | **Comment** | **Uncomment** |
| **Basic Array Structure** | O(n) | - | O(n) | - | O(n) | - | O(n) | - |
| **Array List Structure** | O(n) | O(n) | O(n) | O(n) | O(n) | O(n2) | O(n) | O(n2) |
| **Linked List Structure** | O(n) | O(n) | O(n) | O(n) | O(n) | O(n2) | O(1) | O(n2) |
| **LDLinkedList Structure** | O(n) | O(n) | O(n) | O(n) | O(n) | O(n2) | O(1) | O(n2) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Implementation Type** | **……** | **Send**  **Message** | **Share Post** |
| **Basic Array Structure** | …… | O(n) | O(n) |
| **Array List Structure** | …… | O(n) | O(n) |
| **Linked List Structure** | …… | O(n) | O(1) |
| **LDLinkedList Structure** | …… | O(n) | O(1) |

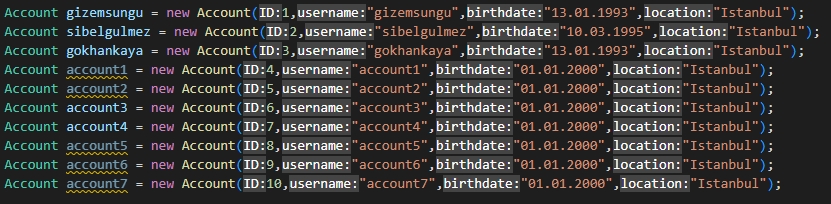
**LDLinkedList Problem Solution Approach**

In my first assignment I have to use only java basic arrays for containing elements. But for this assignment I need to use 2 java collection data structure ArrayList and LinkedList , 1 custom implementing data structure named LDLinkedList. ArrayList and LinkedList are easy to use so real problem was implementing LDLinkedList class. Firstly I need to implement inner “Node” class and basic methods like add,size,get and remove methods for lazy deletion. All methods excluding remove methods were easy to implement. But remove methods are a little bit confused.

I create three remove method. First two is overriden methods from List class java collections and last one is to remove lazy deleted nodes. First one takes object parameter and find its index by using “indexOf” method belong to AbstractList class. By using that index it calls another remove method that takes integer index variable. That method searchs all over the list to find node of the given index. After finding the node this method marks that node as a lazy deleted. And after two succesfull delete operation it calls another remove method named “removeDeleteds”. That method physicallly removes the lazy deleted nodes.

**Test Case(Scenario 4)**

* Creating 10 accounts

****

* Login as “gizemsungu”

****

* Following 4 accounts

**metin içeren bir resim

Açıklama otomatik olarak oluşturuldu**

* “gizemsungu” viewing her own profile

****

* Log out

****

* Login as sibelgulmez

****

* “sibelgulmez” shares two posts

****

* Log out

****

* Login as “gizemsungu”

****

* Liking and commenting second post of sibelgulmez

****

* Viewing “sibelgulmez”s profile

****

* Viewing “sibelgulmez” s post interactions

****

* Unliking and and deleting comment from second post of sibelgulmez

****

* Viewing “sibelgulmez”s profile

****

* Blocking “account4”

****

* Unfollowing “sibelgulmez” and “account3”

**metin içeren bir resim

Açıklama otomatik olarak oluşturuldu**

* Unblocking “account4”

****

* “gizemsungu” views her own profile

****

* “gizemsungu” views her own action history

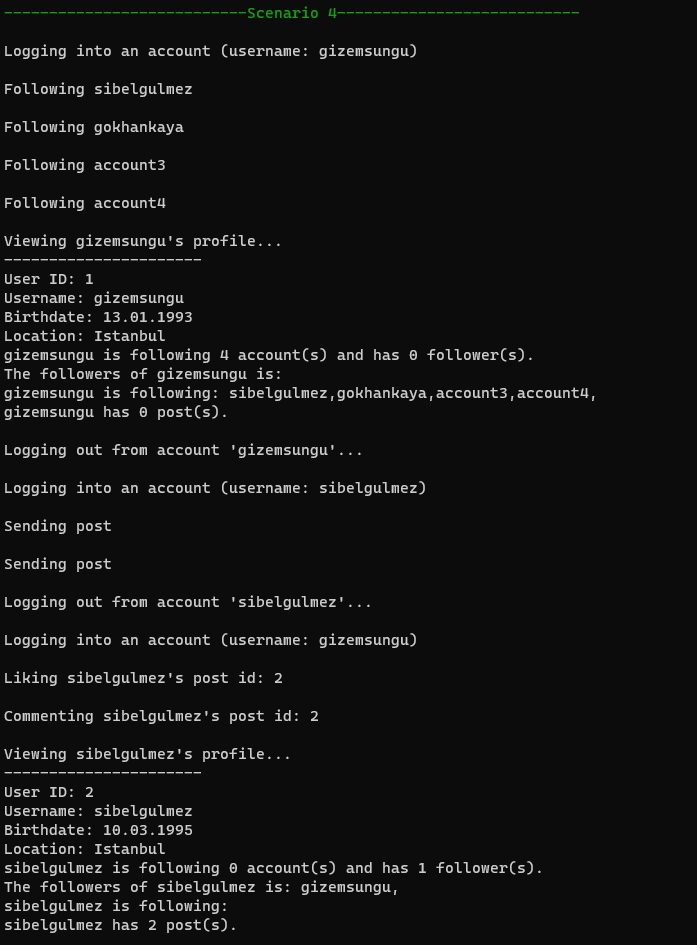
****

* Log out

****

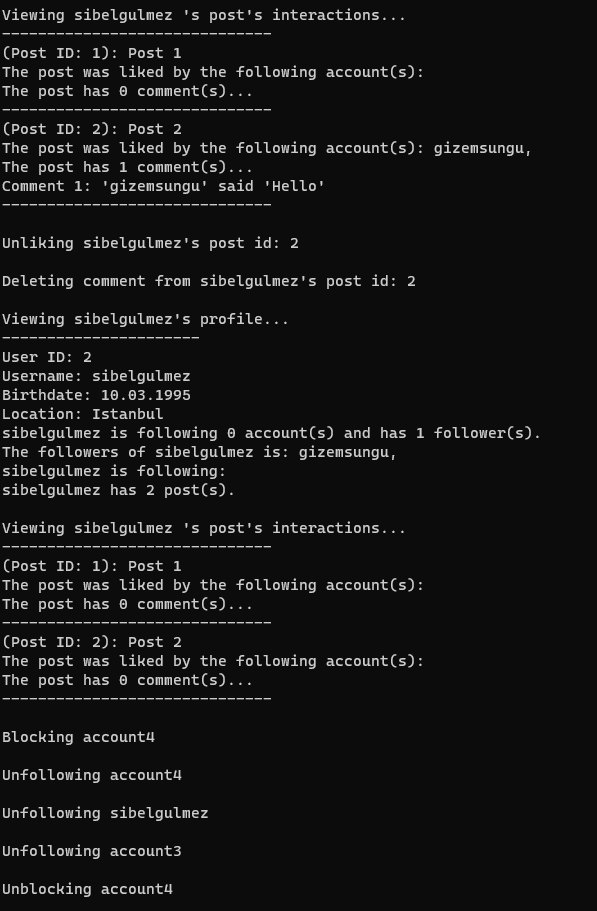
**Results**

* **Scenario 4**



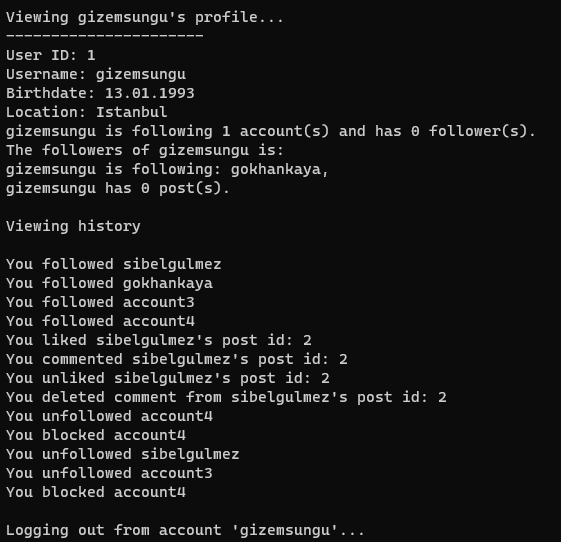
metin içeren bir resim

Açıklama otomatik olarak oluşturuldu



metin içeren bir resim

Açıklama otomatik olarak oluşturuldu



metin içeren bir resim

Açıklama otomatik olarak oluşturuldu