GTU DEPARTMENT OF COMPUTER ENGINERRING

CSE 222/505 – SPRING 2023

HOMEWORK #03 REPORT

Ahmet Yiğit

200104004066

1. Running Command and Results

a. *Creating 4 Account Object*

i.

```
LDLinkedList<Account> accounts = new LDLinkedList<>();
try {
    accounts.add(new Account(accountCount++, username: "a1", birthdate: "21/01/1990", location: "Ankara", accounts));
    accounts.add(new Account(accountCount++, username: "a2", birthdate: "12/12/1980", location: "Istanbul", accounts));
    accounts.add(new Account(accountCount++, username: "a3", birthdate: "11/05/1994", location: "Antalya", accounts));
    accounts.add(new Account(accountCount++, username: "a4", birthdate: "11/05/1994", location: "Antalya", accounts));
}
catch (Exception e){
    System.out.println(e.getMessage());
}
```

b. *Follow User*

i.

```
accounts.get(0).login(accounts);

accounts.get(0).follow(accounts, username: "a2");
accounts.get(0).follow(accounts, username: "a3");
```

c. *Profile Status*

i.

```
User ID: 0
Username : a1
Location: Ankara
Birth Date: 21/01/1990
a1 is following 2 account(s) and has 0 follower(s).
a1 is following a2, a3,
a1 has 0 post(s).

User ID: 1
Username : a2
Location: Istanbul
Birth Date: 12/12/1980
a2 is following 0 account(s) and has 1 follower(s).
The followers of a2 are: a1,
a2 has 0 post(s).

User ID: 2
Username : a3
Location: Antalya
Birth Date: 11/05/1994
a3 is following 0 account(s) and has 1 follower(s).
The followers of a3 are: a1,
a3 has 0 post(s).
```

d. *Unfollow User*

i.

```
accounts.get(0).unfollow(accounts, user: "a2");
```

e. *Profile Status*

i.

```
User ID: 0
Username : a1
Location: Ankara
Birth Date: 21/01/1990
a1 is following 1 account(s) and has 0 follower(s).
a1 is following a3,
a1 has 0 post(s).

User ID: 1
Username : a2
Location: Istanbul
Birth Date: 12/12/1980
a2 is following 0 account(s) and has 0 follower(s).
a2 has 0 post(s).

User ID: 2
Username : a3
Location: Antalya
Birth Date: 11/05/1994
a3 is following 0 account(s) and has 1 follower(s).
The followers of a3 are: a1,
a3 has 0 post(s).
```

f. *Posting post*

i.

```
accounts.get(0).sharePost(accounts, content: "This is my first post");
```

g. *Liking post*

i.

```
accounts.get(2).likePost(accounts, postID: 0);
```

h. *Comment post*

i.

```
accounts.get(2).commentPost(accounts, postID: 0, content: "That is good post!");
```

i. *Post Interactions*

i.

```
( PostID: 0) a1: This is my first post
The post was liked by the following account(s): a3,
The post has 1 comment(s) ...
Comment 1: a3 said 'That is good post!' .
```

j. *Unlike post*

i.
```
accounts.get(2).unlikePost(accounts, postID: 0);
```

k. *Uncomment Post*

i.
```
accounts.get(2).uncommentPost(accounts, postID: 0);
```

l. *Post Interactions*

i.
```
-------------
( PostID: 0) a1: This is my first post
The post has no likes.
The post has no comment.
```

m. *Block User*

i.
```
accounts.get(2).blockUsername(accounts, blockedUsername: "a1");
```

n. *Blocked User View*

i.
```
You cannot view this page because you are blocked!
```

o. *Unblock User*

i.
```
accounts.get(2).unBlockUsername(accounts, blockedUsername: "a1");
```

p. *UnBlocked User View*

i.
```
User ID: 2
Username : a3
Location: Antalya
Birth Date: 11/05/1994
a3 is following 0 account(s) and has 0 follower(s).
a3 has 0 post(s).
```

q. *LDLinkedList Add Method*

i.
```
list.add(0);
list.add(1);
list.add(2);
list.add(3);
list.add(4);
list.add(5);
list.add(6);
list.add(7);
list.add(8);
list.add(9);
for(Integer num : list) System.out.print(num + ", ");
```

ii.
```
0, 1, 2, 3, 4, 5, 6, 7, 8, 9,
```

r. *LDLinkedList Get Method*

i.
```
System.out.println("\n GET 2 IDX Number : "+list.get(2));
```

ii.
```
 GET 2 IDX Number : 2
```

s. *LDLinkedList Remove Method*

i.
```
list.remove( index: 2);

System.out.println("\n GET 2 IDX Number : "+list.get(2));
```

ii.
```
 GET 2 IDX Number : 3
```

t. *LDLinkedList Set Method*

i.
```
list.set(2,99);

System.out.println("\n GET 2 IDX Number : "+list.get(2));
```

ii.
```
GET 2 IDX Number : 99
```

u. *LDLinkedList Size Method*

i.
```
System.out.println("\n GET SIZE : "+list.size());
```

ii.
```
 GET SIZE : 9
```

v. *LDLinkedList Lazy Deletion*

i.
```
for(Integer num : list) System.out.print(num + ", "); // logical list
System.out.println("\n"+list); // physical list
```

ii.
```
0, 1, 99, 4, 5, 6, 7, 8, 9,
0 1 2 99 4 5 6 7 8 9
```

2. Time Complexity Analysis

Structure/Method	Add	Get	Remove	Size	Contains
Array	**Last =>O(1)** **Index=>O(N)**	**O(1)**	**O(N)**	**O(1)**	**O(N)**
ArrayList	**Last =>O(1)** **Index=>O(N)**	**O(1)**	**O(N)**	**O(1)**	**O(N)**
LinkedList	**Begin => O(1)** **End => O(1)** **Index => O(N)**	**O(N)**	**O(N)**	**O(1)**	**O(N)**
LDLinkedList	**Begin => O(1)** **End => O(1)** **Index => O(N)**	**O(N)**	**O(N+2)**	**O(1)**	**O(N+2)**

3. Experimental Running Time Analysis

Implementation Type	Scenario - 1	Scenario - 2	Scenario - 3	Scenario - 4
Basic Array Structure	32,668,100 nanoseconds	42,033,800 nanoseconds	36,264,000 nanoseconds	Not available
Array List Structure	34,625,000 nanoseconds	42,547,200 nanoseconds	39,287,300 nanoseconds	52,705,000 nanoseconds
Linked List Structure	33,282,400 nanoseconds	40,209,800 nanoseconds	46,313,200 nanoseconds	49,791,700 nanoseconds
LD Linked List Structure	38,948,400 Nanoseconds	37,247,500 Nanoseconds	37,787,400 Nanoseconds	53,351,800 nanoseconds

4. Problem Solution Approach for LDLinkedList

a. Import the necessary packages from the Java Collections Framework, such as List and AbstractList.
b. Define a private inner class named Node that represents a single node in the linked list. Each node should have a reference to the next node, a boolean value [isDeleted] to indicate if the node is lazily deleted or not, and an element of type E to store the actual data.
c. Define private instance variables to keep track of the head node, tail node, and the size of the linked list.
d. Implement the add() method to add an element to the end of the linked list. This method should create a new node and set it as the tail node
e. Implement the get() method to retrieve an element at a specific index. This method should iterate through the linked list to find the node at the given index and return the data stored in that node. If the node is lazily deleted, it should be skipped.
f. Most of the override function specialized for LAZY DELETION operations but I do it in getNode function because one logic in one function. So other functions does not depend on the lazy deletion logically.