CSE 222

System Programming

Report

Yavuz Selim İkizler

1901042617

Experimental Running Time

Implementation Type	Scenario 1	Scenario 2	Scenario 3	Scenario 4
Basic Array Structure (HW1)	real 0m0.409s user 0m0.766s sys 0m0.094s	real 0m0.426s user 0m0.750s sys 0m0.172s	-	-
Array List Structure (a)	real 0m0.426s user 0m0.875s sys 0m0.141s	real 0m0.544s user 0m1.000s sys 0m0.203s	real 0m0.578s user 0m0.969s sys 0m0.188s	real 0m0.901s user 0m1.188s sys 0m0.125s
Linked List Structure (b)	real 0m0.386s user 0m0.922s sys 0m0.109s	real 0m0.554s user 0m1.047s sys 0m0.141s	real 0m0.510s user 0m0.891s sys 0m0.156s	real 0m1.050s user 0m1.203s sys 0m0.172s
LD Linked List Structure (c)	real 0m0.436s user 0m0.844s sys 0m0.125s	real 0m0.607s user 0m1.141s sys 0m0.203s	real 0m0.531s user 0m0.922s sys 0m0.188s	real 0m0.931s user 0m1.172s sys 0m0.172s

Time Complexity Analysis

Login Account:

Basic Array Structure: O(1)

ArrayList: O(1)

LinkedList: O(1)

LDLinkedList: O(1)

addPost:

Basic Array Structure: O(n)

ArrayList: O(n)

LinkedList: O(n^2)

LDLinkedList: O(n^2)

follow:

Basic Array Structure: O(1)

ArrayList: O(1)

LinkedList: O(1)

LDLinkedList: O(1)

logout:

Basic Array Structure: O(1)

ArrayList: O(1)

LinkedList: O(1)

LDLinkedList: O(1)

view_profile:

Basic Array Structure: O(n)

ArrayList: O(n)

LinkedList: O(n^2)

LDLinkedList: O(n^2)

view_posts:

Basic Array Structure: O(n)

ArrayList: O(n)

LinkedList: O(n^2)

LDLinkedList: O(n^2)

addLike:

Basic Array Structure: O(1)

ArrayList: O(n)

LinkedList: O(n^2)

LDLinkedList: O(n^2)

addComment:

Basic Array Structure: O(1)

ArrayList: O(1)

LinkedList: O(1)

LDLinkedList: O(1)

addMessage:

Basic Array Structure: O(n)

ArrayList: O(n)

LinkedList: O(n^2)

LDLinkedList: O(n^2)

checkInbox:

Basic Array Structure: O(1)

ArrayList: O(1)

LinkedList: O(1)

LDLinkedList: O(1)

checkOutbox:

Basic Array Structure: O(1)

ArrayList: O(1)

LinkedList: O(1)

LDLinkedList: O(1)

ViewInbox:

Basic Array Structure: O(n^2)

ArrayList: O(n^2)

LinkedList: O(n^3)

LDLinkedList: O(n^3)

view_post_Interactions:

Basic Array Structure: O(n^2)

ArrayList: O(n^2)

LinkedList: O(n^4)

LDLinkedList: O(n^4)

unfollow:

Basic Array Structure: -

ArrayList: O(n)

LinkedList: O(n^2)

LDLinkedList: O(n^2)

unlike:

Basic Array Structure: -

ArrayList: O(n)

LinkedList: O(n^2)

LDLinkedList: O(n^2)

unComment:

Basic Array Structure: -

ArrayList: O(n)

LinkedList: O(n^2)

LDLinkedList: O(n^2)

unblock:

Basic Array Structure: -

ArrayList: O(n)

LinkedList: O(n^2)

LDLinkedList: O(n^2)

blockAccount:

Basic Array Structure: -

ArrayList: O(n^2)

LinkedList: O(n^3)

LDLinkedList: O(n^3)

Problem Solution Approach

Problem Defination

LDLinked list array should implement list interface and extend abstract list. LDLinkedList should implement lazy evaluation strategy. Node is deleted from its linked list logically but not physically. This is done by marking the node as “lazily deleted” (using a boolean value). The removed node is kept in the list with unremoved nodes until another node is removed from the same list. When two nodes are deleted from the list, then the list is traversed and two “lazily deleted” nodes are removed from the list pyhsically.

Solution

In the LDLinkedList, I made a class called LDLinked List that implements the list interface and extends the abstract list. LDLinkedList overrides get() ,set (object),remove(index) ,add() ,size() and isEmpty() methods and also has head (point the beginning of the list) ,tail (points the end of the list) objects integer size and integer del_count variable. I will explain them one by one. Also I made a node class as a helper class. This node class contains the components of each node to be kept in the list. These are next node (for connecting), element (data) and boolean mark variable (for lazy evaluation).

Code: Node Class

```
class Node<E> {
  E element;
  Node<E> next;
  boolean mark; // holds marked value.
  // Constructor
  public Node(E element) {
    this.element = element;
    this.next = null;
    mark=false;
  }
}
```

In order to implement the size and isEmpty method, I assigned an integer variable named size. This variable increases by 1 each time the add method is called and decreases by 1 when the remove method is called.

The size method returns the size variable, and the isEpty method returns true if size is equal to 0, otherwise flase.

Code:isEmpty

```
@Override
public boolean isEmpty() {
  return size == 0;
}
```

Code:size

```
@Override
public int size() {
  return size;
}
```

In some methods, such as get, set. We need to ignore the nodes we marked according to the lazy eveluation strategy. For this, I defined statements in these methods. If it sees the marked node, it will go to the next node since the marked values are deleted from the list but are physically present there.

Code:ignore mark

```
for (int i = 0; i < index; i++) {
  if(current.mark==true){
    current=current.next;
  }
  current = current.next;
  if(current.mark==true){
    current=current.next;
  }


}
if(index==0 && head.mark==true){
  current=current.next;
}
```

get() set():

The get method takes index as a parameter. There is a temporary current node in the method, this node is initially assigned the head node, then it is assigned to the next address in the for loop until it reaches the desired position in the given index, and if it encounters the marked nodes, it skips it. When the desired node is reached, the data of that node is returned.

The operation of the set method is the same as the get method, only the data from the parameter is assigned instead of returning.

Add():

The add method assigns the variable it takes as a parameter to the head node if the list is empty. if not, it connects it to the next node of the tail node and renews the tail node.

remove(index):

The remove method takes the index variable as a parameter. According to the mode from the del_count variable (if del_count is 1, it physically deletes the nodes, if 0, it applies 2 different operations).

If del_count is 0, if it is at the beginning of the list according to the value of the index, it assigns true to the mark of the leading node, if it is at the end, it assigns it to the tail node,

for intermediate values, it goes step by step to the address given in the index and marks the node in that index. del_count modified and decreased size by 1

if del_count is 1, the node at the given position in the index and the marked node must be deleted. first, it deletes the value given in the index. It applies the algorithm that used in the get method to reach the node in the index. It applies different deletion algorithms if the index is at the beginning, at the end, or in the middle of the list. After deleting the node at the address of the index, it finds the marked node and deletes it. It applies different deletion algorithms according to the location of the marked node again. del_count modified and decreased size by 1.

Running Comments and Results

Create account

```
System.out.println(x:"          ... Creating accounts...      ");
Account user1= new Account(user_id:12745,user_name:"gizemsüngü",location:"Istanbul",birth_date:"21.03.1993");
Account user2= new Account(user_id:12395,user_name:"sibelgülmez",location:"izmir",birth_date:"10.03.1995");
Account user3= new Account(user_id:12348,user_name:"gökhankaya",location:"Ankara",birth_date:"01.01.1985");
```

Login Account

```
System.out.println(x:"          ... Logging into an account (username: sibelgulmez)… ");
user2.login();
System.out.println(x:"");
```

Create Post / Share Po

```
System.out.println(x:"          ... Logging into an account (username: sibelgulmez)… ");
user2.login();
System.out.println(x:"");
```

```
Post post2=new Post(post_id:2,content:"Java the coffee...");
user2.addPost(post1);
```

st

Follow Account

```
System.out.println(x:"         ... Following gizemsungu and gokhankaya...");
user2.follow(user1);
```

Logout

```
System.out.println(x:"         ... Logging out from account 'sibelgulmez'…");
user2.logout();
```

View Profile

```
System.out.println(x:"         ... Viewing sibelgulmez's profile...");
user3.view_profile(user2);
```

View Post

```
System.out.println(x:"         ... Viewing sibelgulmez'posts...");
user3.view_posts(user2);
```

Create Like

```
Like like1=new Like(interaction:9,user3,post:1);
```

Create Comment

```
Comment comment1= new Comment(interaction:9,post:1,user3,comment:"me too");
```

Add Like

```
post1.addLike(like1);
```

Add Comment

```
post1.addComment(comment1);
```

Create Message

```
Message newmessage= new Message(message:7,sender:12348,reciever:12745,content:"This homework is too easy!");
```

Add Message

```
user3.addMessage(newmessage, account_list, account_size);
```

Check Inbox

```
user1.checkInbox();
```

Check Outbox

```
user1.checkOutbox();
```

View Inbox

```
user1.ViewInbox(account_list,account_size);
```

View Post Interaction

```
user1.view_post_interactions(user2);
```

Unfollow

```
user2.unfollow(user3);
```

Unlike post

```
post1.unlike(like1);
```

Uncomment Post

```
post1.unComment(comment1);
```

Unblock Account

```
user1.unblock(user2);
```

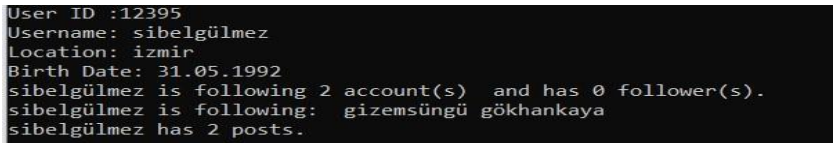
Block Account

```
user1.blockAccount(user2);
```

Create Account

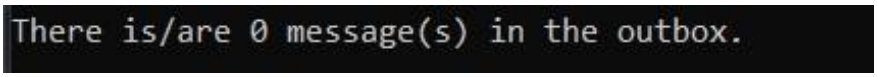
```
An account with username gizemsüngü has been created.
An account with username sibelgülmez has been created.
An account with username gökhankaya has been created.
```

View Profile

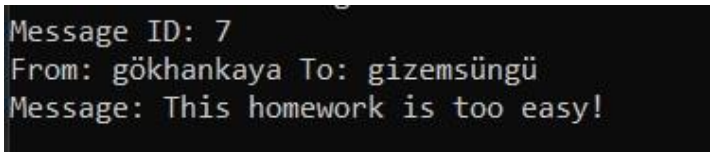


Inbox

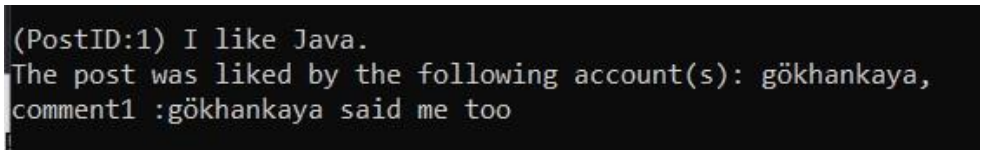
Outbox



View Inbox

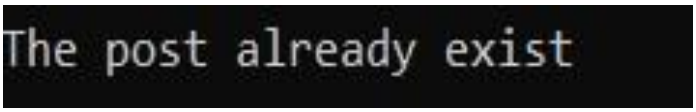


View Post Interaction

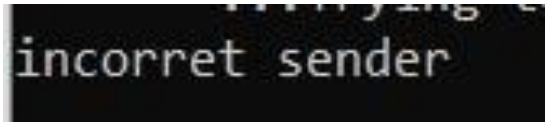


Not Logged in message

Post Exist Message



Incorrect Sender Message



User not exist message.

Show actions.

Block error message

the account: gizemsüngü blocked you. you can not view his/her profile.