Hacettepe University

Department Of Computer Engineering

BBM 203 Software Laboratory

Fall 2015 - Experiment 3

Final Report

Name and Surname : Yılmaz Baysal

Identity Number : 21327694

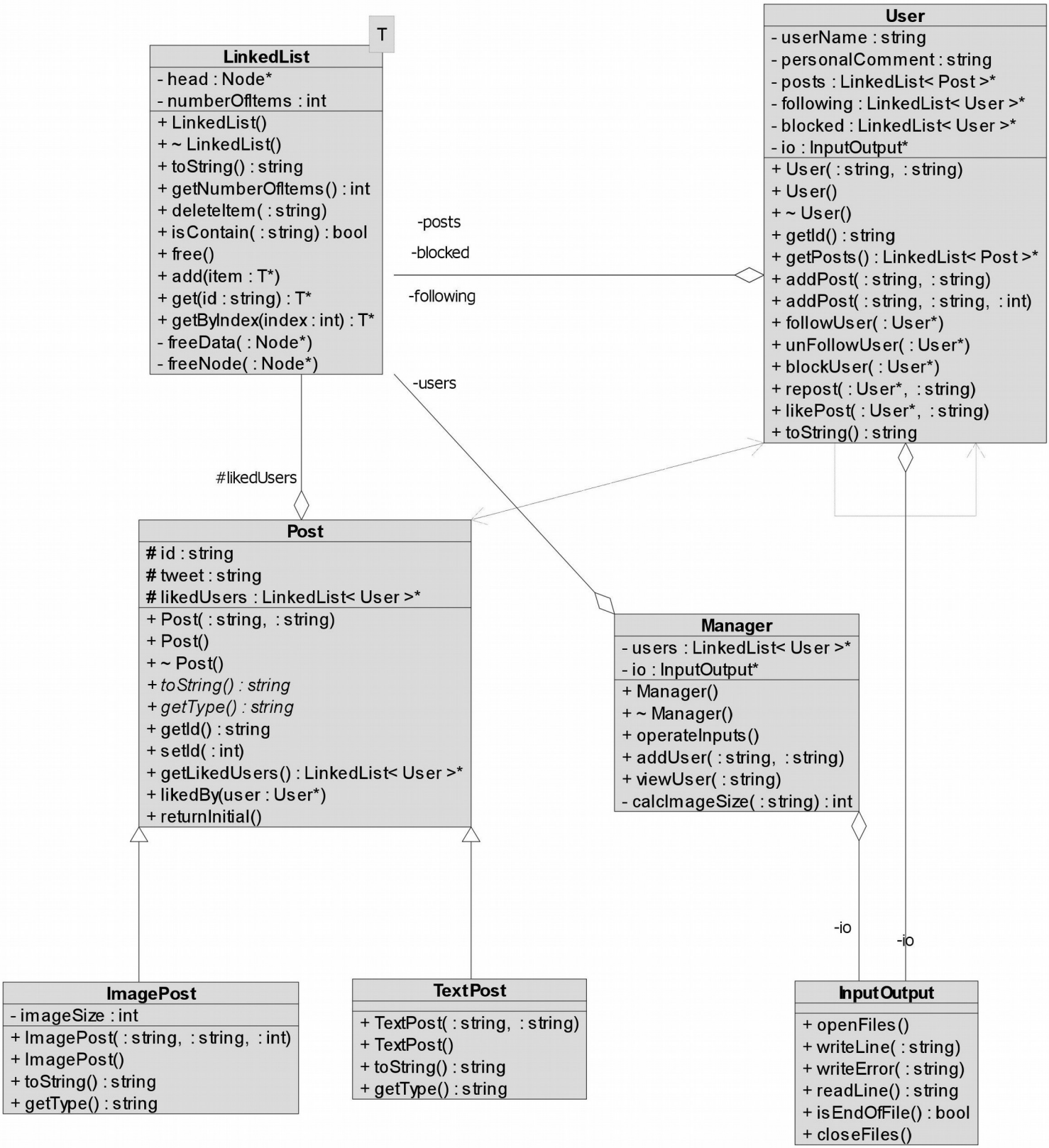
Subject : A hypothetic Twitter App / Linked Lists & OOP

Advisors : Asist. Prof. Adnan ÖZSOY

R.A. Ahmet Selman BOZKIR

Programming Language : C++

1. Class Diagram



2. Some Of The Algorithms That I Used

2.0 Main algorithm

First, read the inputs line by line. For each line, control that, is there anything to cause an error (the user exists or not, does user have given post or not). After that, call the corresponding method to process the input. Finally, print the result of the process to both file and console. If the the input causes an error, print a proper error message.

2.1 Adding an item to the list: void add(T* item)

Take a template parameter and add it to the head of the linked list, then increase the numberOfItems variable's value by one.

2.2 Deleting an item from the list: void delete(string Id)

Create two temporary node pointers to keep previous and current nodes. Then, iterate all list from head pointer to the NULL (end of the list). After each iteration, control whether the item's id is equal to the given id. When they are equal, assign the previous node's pointer as curent pointer's next pointer, then delete the current pointer.

2.3 Getting an item from the list: T* getItem(string Id)

Iterate the list from head to NULL. If the id of the current item is equal to the given id, return the current item as a pointer.

2.4 Reposting a post of the user: void repost(User* reposted, string postId)

First control the possible errors (does the user have a such post or blocked by the reposted user). Create a Post pointer. If the type of the reposted post is TextPost then assign the Post pointer as new TextPost. If not, assign the Post pointer as new ImagePost. After that, assign the value of new Post pointer as reposted post's value then delete the likes of the new post.

2.5 Blocking a user: void blockUser(User* user2)

First control the possible errors. Then add the user to the list of blocked users. After that, delete the likes of user2 from this user's posts. Finally, if the blocked user following this user, remove this connection (remove this user from blocked user's following list).

3. Data Structure

I used linked list because, there is no limit for number of users or posts that we need to store and it is able to perform adding and deleting operations more efficient. So, I had to choose a structure that can do the insertion and deletion operations without shifting and it can store limitless items in it (if there is enough memory). Also the structure that I should use have to store different types of datas in the same place. To do that, I wrote a template variable in the linked list's node that can store either User* or Post* values.

4. Problems

I had problems with avoiding memory leaks. Every time I tried to delete a pointer which is stored in heap, my program crashed. Finally, I got over this problem and could give the allocated memory to the system (I suppose).

5. Future And Personal Comments

First of all, I would separate the posts by subjects. If someone in my social platform do not want to see any political post for a while, he or she can hide all the political posts that his or her friends share. If some users want to see only scientific content, they could do that. Also I would add mini games (for example puzzle) at the right side of the screen so, users will spend much more time on the social platform.