HACETTEPE UNIVERSITY COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

NAME:NEHİL

SURNAME:DANIŞ

STUDENT ID:21327876

SUBJECT:A HYPOTHETIC TWITTER APP-LINKED LISTS & OOP/

EXPERIMENT 3/FINAL REPORT

PROGRAMMING LANGUAGE:C++(minGW)

ADVISOR:R.A AHMET SELMAN BOZKIR,DR. ADNAN ÖZSOY

1.Algorithm

1. Into the main function the program will call the readInput function.

1.1. The program will read input file line by line and then readInput function will call the split function.

1.1.1. Into the split function every line will be devided by using space delimeter.After dividing each line the program will call the getTheCommandDone function to implement the commands.

1.1.1.1. Depending on the command parameter ,the program will call the appropriate function.

1.1.1.1.1. If the income command is **'AddUser'** ,the program will call the adduser function that stays into the AddUser class.A user will be created in here.And then the program will control the existance of user.If this user had been created before,the program will give an error message.Else the program will create a new user.

1.1.1.1.2. If the income command is **'ViewUser'** ,the program will call the viewuser function that stays into the ViewUser class.At first the program will control the existance of user that we want to view by calling existanceOfUser function.If the user isn't in the user list the program will give an error message.Else the program will view the user.

1.1.1.1.3. If the income command is **'FollowUser'** ,the program will call the followSomeOne function that stays into the User class.At first there will be a block control and followed control.If follower had been blocked by following user,the program will give an error message.Else the program control the followed or not.If following user had already been followed by follower,the program will give an error message.Else The program will create a following user item and then this items put into the follower's followinguser list.

1.1.1.1.4. If the income command is **'AddPost'** ,the program will call the addPost function that stays into the User class.At first the program will control the existance of user that want to add a new post.There is no control except existance of user in this function.If the user is in the user list the program will find the user and then create a new post.This post could be image post or text post.For post I used inheritance.Because there are two different kinds of posts.Depending on the type of post new post will be created.And then the new post will be added to the user's post list.

1.1.1.1.5. If the income command is **'Repost'** ,the program will call the repost function that stays into the User class. At first there will be a block control and existance control.If both of these user is in the user list the program will continue to execution of this function else will give an error message.And then If reposter hadn't been blocked by reposted user,the program will continue to execute this function ,else will give an error message.And then the function will call the addRepostedPost function.Depending on the type of post the reposted post will be added to the reposter's post list.

1.1.1.1.6. If the income command is **'BlockUser'** ,the program will call the blockSomeOne function that stays into the User class.The program will control the existance of blocker and blocked one.If one of them wasn't in the user list,the program would give an error message.Else the program will control blocked or not.If blocked one had already been blocked by blocker,the program will give an error message.Else the porgram will implement the command.And then realtionship between blocked one and blocker will be cut.And all likes that come from blocker to the blocked one will be deleted.

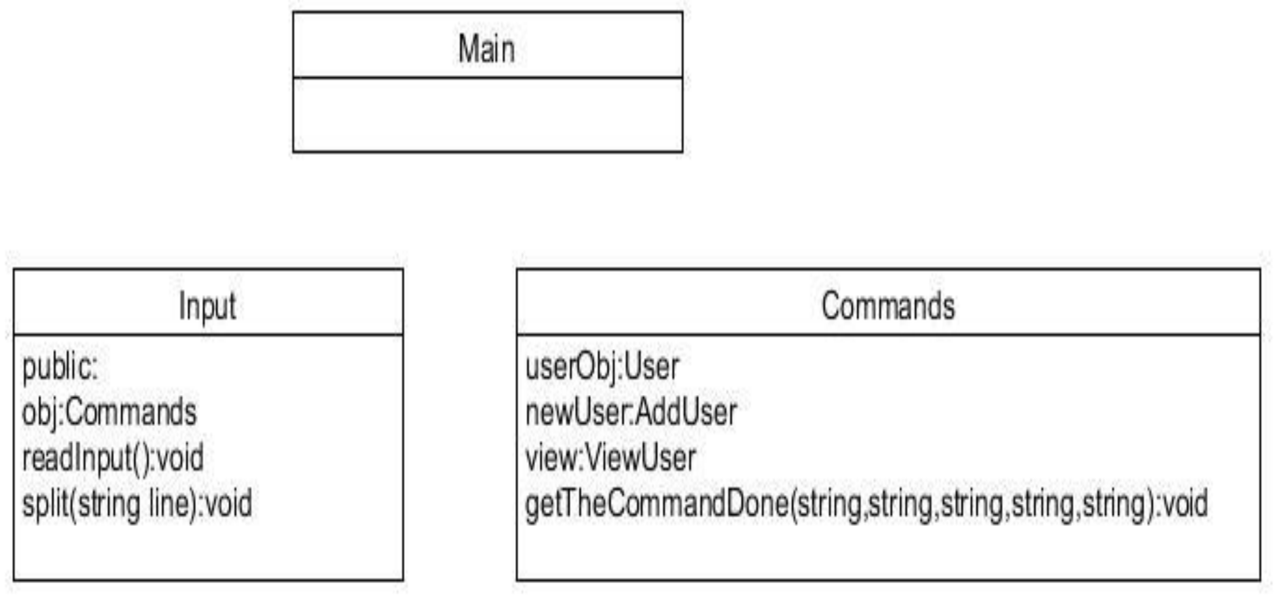
1.1.1.1.7. If the income command is **'UnfollowUser'** ,the program will call the unfollowSomeOne function that stays into the User class. The program will control the existance of unfollower and unfollowed one.If one of them wasn't in the user list,the program would give an error message.Else the program will control blocked or not.If unfollower had been blocked by unfollowed one,the program will give an error message.Else the porgram will control the unfollow.If unfollowed one had already been unfollowed by unfollower,the program will give an error message.Else the program will implement the command.

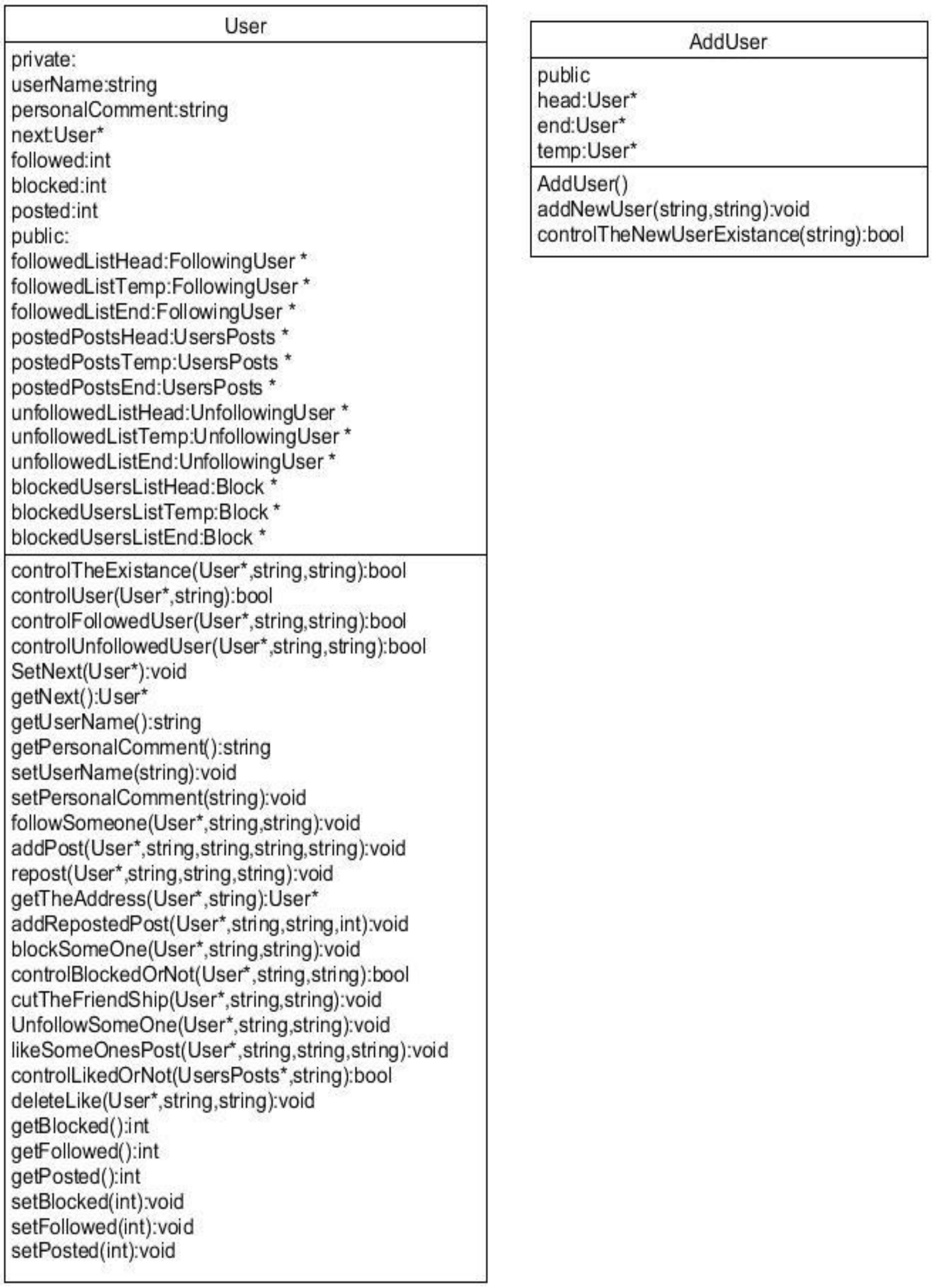
1.1.1.1.8. If the income command is **‘LikePost’** ,the program will call the likeSomeOnesPost function that stays into the User class. The program will control the existance of liker and liked one.If one of them wasn’t in the user list,the program would give an error message.Else the program will control blocked or not.If liked one had blocked liker before,the program will give an error message.Else the porgram will control the liked or not.If the liked one’s post had already been liked by liker,the program will give an error message.Else the program will implement the command.

1.2. The program will close the input file.

2. The program will finish.

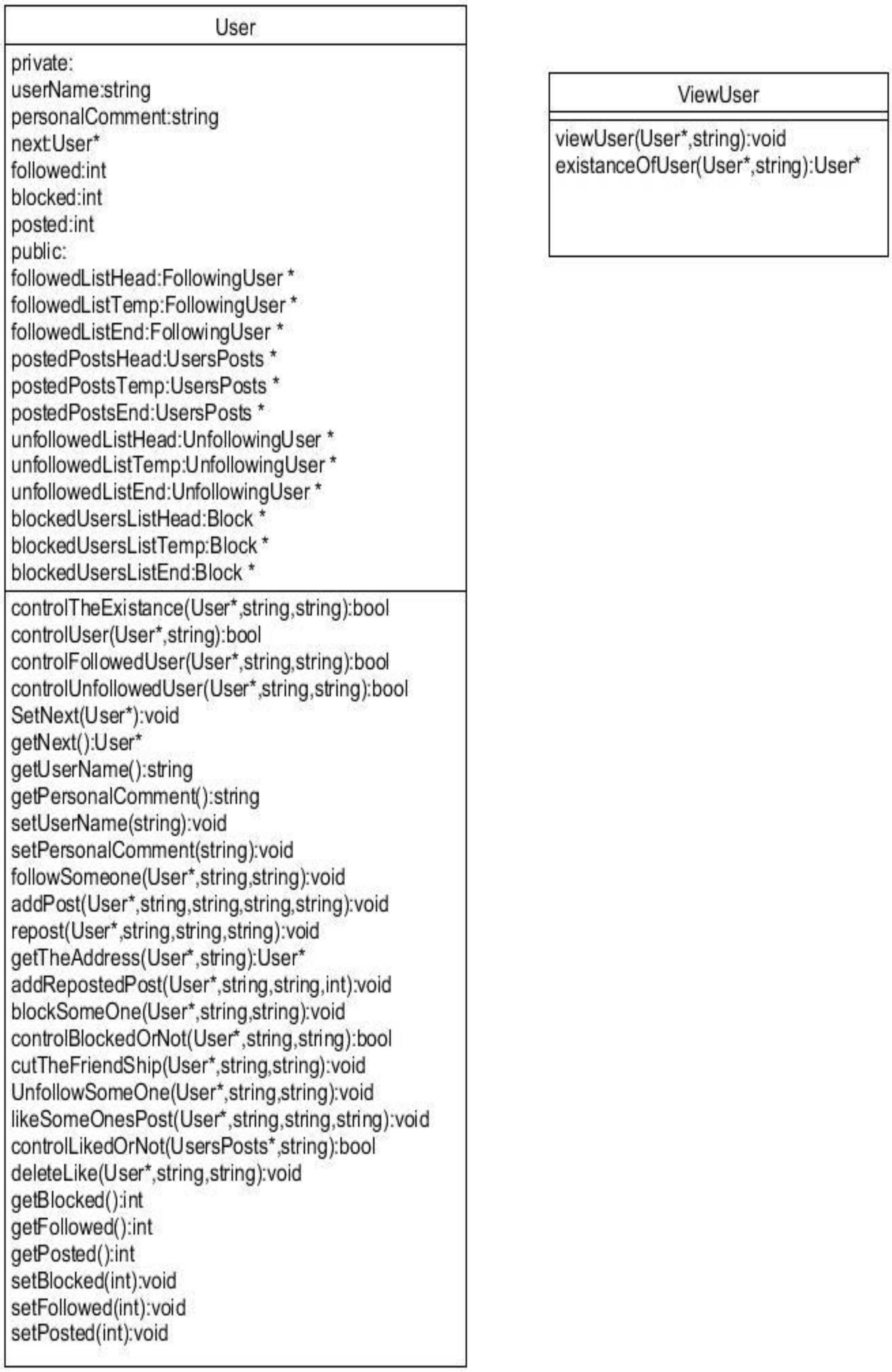
<u>2.Class Diagrams</u>

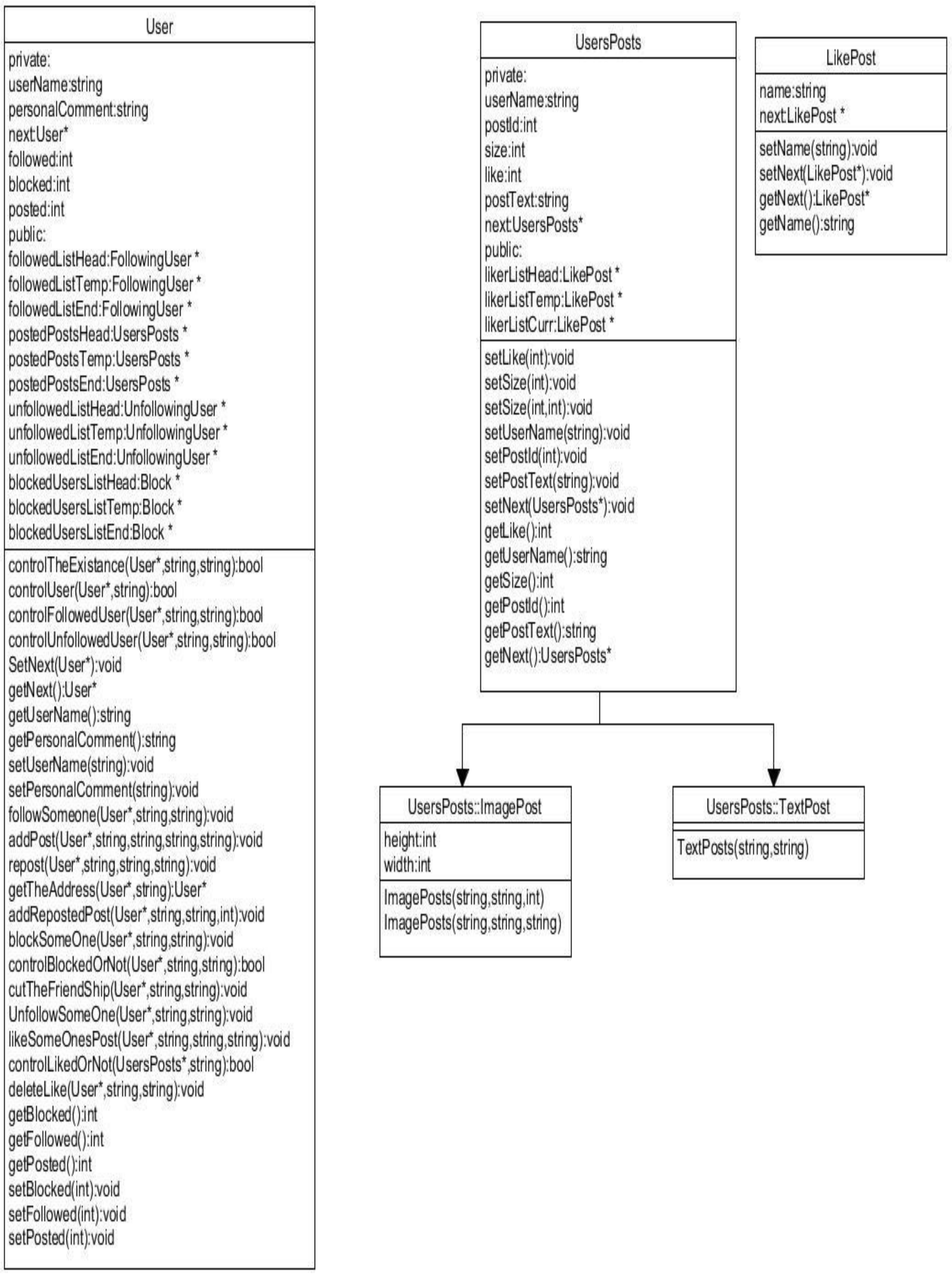


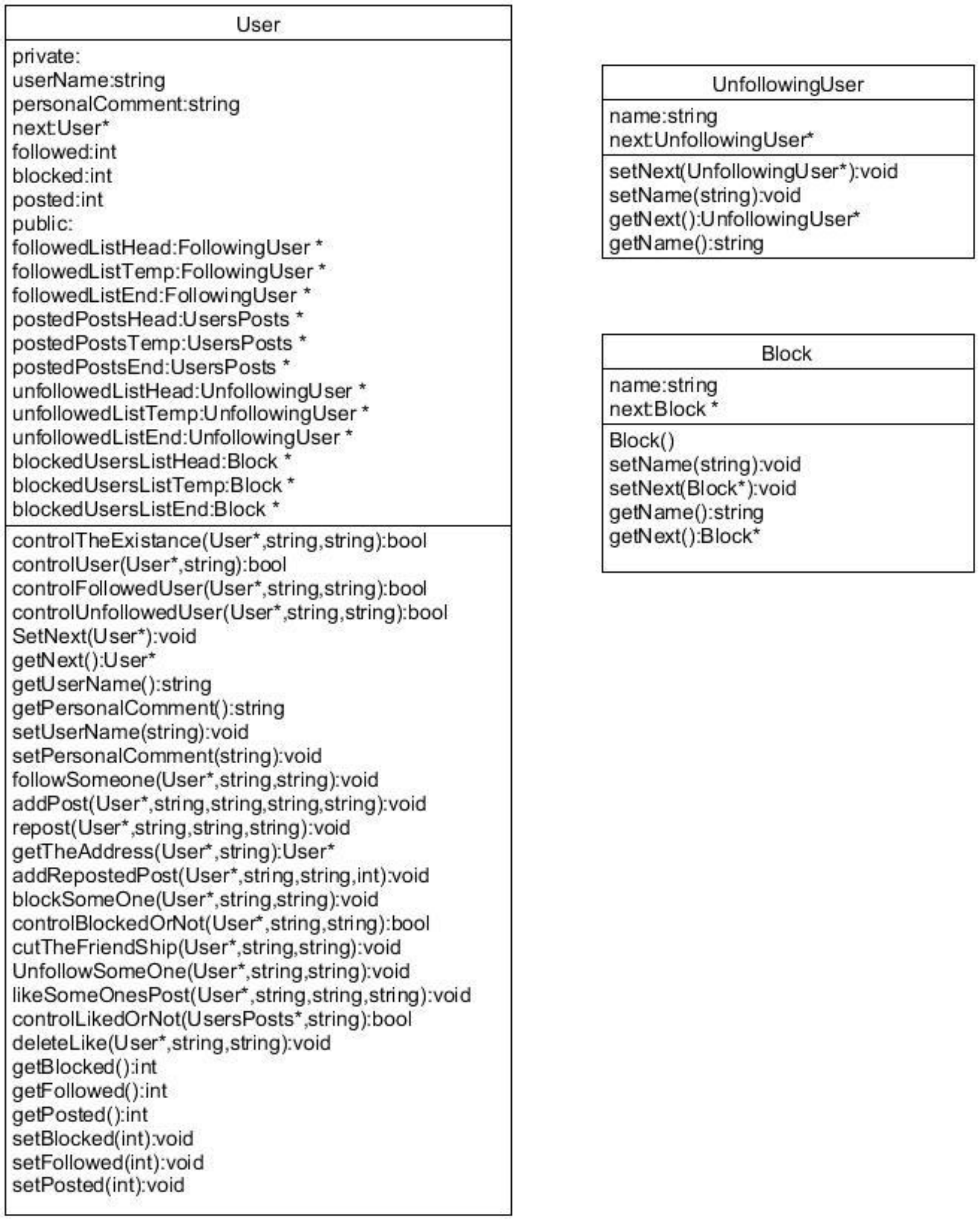


```
User
-----------------------------------------------
private:
userName:string
personalComment:string
next:User*
followed:int
blocked:int
posted:int
public:
followedListHead:FollowingUser *
followedListTemp:FollowingUser *
followedListEnd:FollowingUser *
postedPostsHead:UsersPosts *
postedPostsTemp:UsersPosts *
postedPostsEnd:UsersPosts *
unfollowedListHead:UnfollowingUser *
unfollowedListTemp:UnfollowingUser *
unfollowedListEnd:UnfollowingUser *
blockedUsersListHead:Block *
blockedUsersListTemp:Block *
blockedUsersListEnd:Block *
-----------------------------------------------
controlTheExistance(User*,string,string):bool
controlUser(User*,string):bool
controlFollowedUser(User*,string,string):bool
controlUnfollowedUser(User*,string,string):bool
SetNext(User*):void
getNext():User*
getUserName():string
getPersonalComment():string
setUserName(string):void
setPersonalComment(string):void
followSomeone(User*,string,string):void
addPost(User*,string,string,string,string):void
repost(User*,string,string,string):void
getTheAddress(User*,string):User*
addRepostedPost(User*,string,string,int):void
blockSomeOne(User*,string,string):void
controlBlockedOrNot(User*,string,string):bool
cutTheFriendShip(User*,string,string):void
UnfollowSomeOne(User*,string,string):void
likeSomeOnesPost(User*,string,string,string):void
controlLikedOrNot(UsersPosts*,string):bool
deleteLike(User*,string,string):void
getBlocked():int
getFollowed():int
getPosted():int
setBlocked(int):void
setFollowed(int):void
setPosted(int):void
```

```
AddUser
-----------------------------------------------
public
head:User*
end:User*
temp:User*
-----------------------------------------------
AddUser()
addNewUser(string,string):void
controlTheNewUserExistance(string):bool
```


User
private:
userName:string
personalComment:string
next:User*
followed:int
blocked:int
posted:int
public:
followedListHead:FollowingUser *
followedListTemp:FollowingUser *
followedListEnd:FollowingUser *
postedPostsHead:UsersPosts *
postedPostsTemp:UsersPosts *
postedPostsEnd:UsersPosts *
unfollowedListHead:UnfollowingUser *
unfollowedListTemp:UnfollowingUser *
unfollowedListEnd:UnfollowingUser *
blockedUsersListHead:Block *
blockedUsersListTemp:Block *
blockedUsersListEnd:Block *
controlTheExistance(User*,string,string):bool
controlUser(User*,string):bool
controlFollowedUser(User*,string,string):bool
controlUnfollowedUser(User*,string,string):bool
SetNext(User*):void
getNext():User*
getUserName():string
getPersonalComment():string
setUserName(string):void
setPersonalComment(string):void
followSomeone(User*,string,string):void
addPost(User*,string,string,string,string):void
repost(User*,string,string,string):void
getTheAddress(User*,string):User*
addRepostedPost(User*,string,string,int):void
blockSomeOne(User*,string,string):void
controlBlockedOrNot(User*,string,string):bool
cutTheFriendShip(User*,string,string):void
UnfollowSomeOne(User*,string,string):void
likeSomeOnesPost(User*,string,string,string):void
controlLikedOrNot(UsersPosts*,string):bool
deleteLike(User*,string,string):void
getBlocked():int
getFollowed():int
getPosted():int
setBlocked(int):void
setFollowed(int):void
setPosted(int):void
ViewUser
viewUser(User*,string):void
existanceOfUser(User*,string):User*


User
private:
userName:string
personalComment:string
next:User*
followed:int
blocked:int
posted:int
public:
followedListHead:FollowingUser *
followedListTemp:FollowingUser *
followedListEnd:FollowingUser *
postedPostsHead:UsersPosts *
postedPostsTemp:UsersPosts *
postedPostsEnd:UsersPosts *
unfollowedListHead:UnfollowingUser *
unfollowedListTemp:UnfollowingUser *
unfollowedListEnd:UnfollowingUser *
blockedUsersListHead:Block *
blockedUsersListTemp:Block *
blockedUsersListEnd:Block *
controlTheExistance(User*,string,string):bool
controlUser(User*,string):bool
controlFollowedUser(User*,string,string):bool
controlUnfollowedUser(User*,string,string):bool
SetNext(User*):void
getNext():User*
getUserName():string
getPersonalComment():string
setUserName(string):void
setPersonalComment(string):void
followSomeone(User*,string,string):void
addPost(User*,string,string,string,string):void
repost(User*,string,string,string):void
getTheAddress(User*,string):User*
addRepostedPost(User*,string,string,int):void
blockSomeOne(User*,string,string):void
controlBlockedOrNot(User*,string,string):bool
cutTheFriendShip(User*,string,string):void
UnfollowSomeOne(User*,string,string):void
likeSomeOnesPost(User*,string,string,string):void
controlLikedOrNot(UsersPosts*,string):bool
deleteLike(User*,string,string):void
getBlocked():int
getFollowed():int
getPosted():int
setBlocked(int):void
setFollowed(int):void
setPosted(int):void
UsersPosts
private:
userName:string
postId:int
size:int
like:int
postText:string
next:UsersPosts*
public:
likerListHead:LikePost *
likerListTemp:LikePost *
likerListCurr:LikePost *
setLike(int):void
setSize(int):void
setSize(int,int):void
setUserName(string):void
setPostId(int):void
setPostText(string):void
setNext(UsersPosts*):void
getLike():int
getUserName():string
getSize():int
getPostId():int
getPostText():string
getNext():UsersPosts*
LikePost
name:string
next:LikePost *
setName(string):void
setNext(LikePost*):void
getNext():LikePost*
getName():string
UsersPosts::ImagePost
height:int
width:int
ImagePosts(string,string,int)
ImagePosts(string,string,string)
UsersPosts::TextPost
TextPosts(string,string)



```
User
-------------------------------------------
private:
userName:string
personalComment:string
next:User*
followed:int
blocked:int
posted:int
public:
followedListHead:FollowingUser *
followedListTemp:FollowingUser *
followedListEnd:FollowingUser *
postedPostsHead:UsersPosts *
postedPostsTemp:UsersPosts *
postedPostsEnd:UsersPosts *
unfollowedListHead:UnfollowingUser *
unfollowedListTemp:UnfollowingUser *
unfollowedListEnd:UnfollowingUser *
blockedUsersListHead:Block *
blockedUsersListTemp:Block *
blockedUsersListEnd:Block *
-------------------------------------------
controlTheExistance(User*,string,string):bool
controlUser(User*,string):bool
controlFollowedUser(User*,string,string):bool
controlUnfollowedUser(User*,string,string):bool
SetNext(User*):void
getNext():User*
getUserName():string
getPersonalComment():string
setUserName(string):void
setPersonalComment(string):void
followSomeone(User*,string,string):void
addPost(User*,string,string,string,string):void
repost(User*,string,string,string):void
getTheAddress(User*,string):User*
addRepostedPost(User*,string,string,int):void
blockSomeOne(User*,string,string):void
controlBlockedOrNot(User*,string,string):bool
cutTheFriendShip(User*,string,string):void
UnfollowSomeOne(User*,string,string):void
likeSomeOnesPost(User*,string,string,string):void
controlLikedOrNot(UsersPosts*,string):bool
deleteLike(User*,string,string):void
getBlocked():int
getFollowed():int
getPosted():int
setBlocked(int):void
setFollowed(int):void
setPosted(int):void
```

```
UnfollowingUser
-------------------------------------------
name:string
next:UnfollowingUser*
-------------------------------------------
setNext(UnfollowingUser*):void
setName(string):void
getNext():UnfollowingUser*
getName():string
```

```
Block
-------------------------------------------
name:string
next:Block *
-------------------------------------------
Block()
setName(string):void
setNext(Block*):void
getName():string
getNext():Block*
```

3.Data Structures

I used linked list for implementing this project.Because in my oppinion we cannot use the base patterns.We have to create something to implement this project.Because if I used vector,I didn't learn the logic of linked list.In this way I learned lots of thing about linked list.I wrote all my functions by myself.I didn't use any base pattern like vector delete function.There is a big linked list that keep all users records.And every user has their own linked list.Because of that reason,my structure is something like nested classed.

4.Problem

At first I didn't come up with any problem.But then,I copy my code and paste to the visual studio compiler.And then the program was crashed.I wrote my first program into the Eclipse and everything looked okay,but then I realized that I didn't assign NULL to some variables that I have to assign.And my add user function worked in wrong way.Fixing all these problems,my program compiled and executed.

Note:I added new notation to the code if in input file ,the user doesn't use an underscore charactare,the program writes **'ERROR'** to the console.

5.Personal Comments

Every command that has taken two arguments for user,could be implemented for both of these user.Especially block command could be mutual.Because everything would less complicated then now.If I used everything that I can ,I would add send message command,delete post command, delete user command.Except this notations ,the project is really helpful for me to learn linked lists deeply.