Gebze Technical University
Department of Computer Engineering
CSE 222/505 - Spring 2023
PA 1 Report

AHMET ALPER UZUNTEPE

1901042669

✓ SYSTEM REQUIREMENTS

1)First,Create Parameters For Use in Methods(Array size 100)

I determined the necessary parameters and created them. I stored the objects in the array. In this way, I could count and change them whenever I wanted.

```
public class Message extends Interaction {
    1 usage
    protected int messageID;
    1 usage
    protected Account senderID;
    1 usage
    protected Account receiverID;

    5 usages
    public String content ;
```

```
public class Post {
    12 usages
    protected final int postId;
    3 usages
    protected int accountID ;

    5 usages
    protected String Content ;

    6 usages
    protected Account[] likes = new Account[100];
    7 usages
    protected Account[] comments = new Account[100];
```

```
public class Like extends Interaction {
    2 usages
    protected boolean IsItliked ;
```

```
public class Comment extends Interaction{
    2 usages
    protected boolean Hascomment;
    1 usage
```

```
protected Account[] likes = new Account[100];
7 usages
protected Account[] comments = new Account[100];
```

```
17 usages
private Post[] posts = new Post[100];

7 usages
private Message[] Inbox = new Message[100];
7 usages
private Message[] Outbox = new Message[100];
6 usages
private Account[] following = new Account[100];
6 usages
private Account[] followers = new Account[100];
```

```
public class Account {
    3 usages
    protected int accountID;
    23 usages
    protected String username;
    2 usages
    protected String birthdate;
    2 usages
    protected String location;

    2 usages
    protected String Content ;
    12 usages
    private boolean IsItOpen;
```

2) Create Constructures(Create Default and more constructure (We do not need to all of them))

```
protected Post(int postIDs, String contents, int accountIDs) {
    this.postId = postIDs;
    this.Content = contents;
    this.accountID = accountIDs;
}
protected Post(int postIDs, String contents) {
    this.postId = postIDs;
    this.Content = contents;
}
private Post(int postIDs, int accountIDs) {
    this.accountID = accountIDs;
    this.postId = postIDs;

}
2 usages
protected Post(int postIDs) {
    this.postId = postIDs;

}
```

I created the constructs within the specified parameters. I also added the Default constructure, but I can qualify it as non-functional.

3) Create Methods

- ✓ **ADD/DELETE LIKE**

```
public void addLike(Account obj){
    int i = 0 ;
    for(;i<100;i++){
        if(this.likes[i]== null) {
            this.likes[i] = obj;
            break;
        }
    }
}
```

```
public void disLike(Account obj){
    int i = 0 ;
    for(;i<100;i++){
        if(this.likes[i] == obj){
            this.likes[i]= null;
        }
    }
}
```

Our method saves object in like array.

- ✓ ADD/DELETE COMMENT

```
public void addComment(Account obj,String chat){
    int i = 0 ;
    for(;i<100;i++){
        if(this.comments[i]==null) {
            this.comments[i] = obj;
            this.comments[i].Content= chat ;
            break;
        }
    }
}
```

```
public void deleteComment(Account obj,String chat){
    int i = 0 ;
    for(;i<100;i++){
        if(this.comments[i]==obj) {
            this.comments[i]= null;
            this.Content = null ;
            break;
        }
    }
}
```

Our method saves object in array and we will save the comment in the content section.

✓ View Likes/Comments

```java
public void viewLikes(){

    int i = 0 ,count = 0 ;
    StringBuilder names = new StringBuilder();
    for (;i<100;i++){
        if(this.likes[i] != null){
            count++;
            //builder yardımı ile stringleri birbirine ekliyoruz
            names = names.append(this.likes[i].username+"\t");
        }
    }
    if(count==0){
        System.out.println("PostID: "+this.postId+"\tThere is no Likes.");
    }
    else{
        System.out.println("PostID: "+this.postId+"\tliked by\t"+count+" person \tAccount usernames;\t"+names);
    }

}
```

```java
public void viewComments(){

    int i = 0 ,count = 0 ;
    //StringBuilder names = new StringBuilder();
    for (;i<100;i++){
        if(this.comments[i] != null){
            count++;
            //builder yardımı ile stringleri birbirine ekliyoruz
            //names = names.append(this.comments[i].accountID);
        }
    }
    if(count==0){
        System.out.println("PostID: "+this.postId+"\tThere is no comments.");
    }
    else {
        System.out.println("PostID: " +this.postId + "\thave comment by\t" + count + " person :");
        for (i = 0; i < 100; i++) {
            if (this.comments[i] != null) {
                System.out.println("PostID: " + this.postId + "\thave comment by\t" + this.comments[i].username + ":\t" + this.comments[i].Content);
            }
        }
    }
```

We count the total likes and comments with the help of the Count variable. I used the String Builder to see the likes. We print the names of the objects in the printing process.

✓ Get Following/Followers Information

```
2 usages
public String getUsername(Account obj) { return obj.username; }
2 usages
public int getAccountID(Account obj) {  return obj.accountID; }
2 usages
public String getBirthdate(Account obj) { return obj.birthdate; }
2 usages
public String getLocation(Account obj) { return obj.location; }

2 usages
public void getFollowingInfo(Account obj){
    int i = 0 ,count = 0 ;
    StringBuilder names = new StringBuilder();
    for (;i<100;i++){
        if(obj.following[i] != null){
            count++;
            //builder yardımı ile stringleri birbirine ekliyoruz
            names = names.append(obj.following[i].username+"\t");
        }
    }
    System.out.println(obj.username+"\tfollowing\t"+count+"\taccounts;"+names);
}
```

```
public void getFollowerInfo(Account obj){
    int i = 0 ,count = 0 ;
    StringBuilder names = new StringBuilder();
    for (;i<100;i++){
        if(obj.followers[i] != null){
            count++;
            names = names.append(obj.followers[i].username+"\t");
        }
    }
    System.out.println(obj.username+"\tfollowed by\t"+count+"\taccounts;\t"+names);
}
```

We get the information with the get methods. We add the follower following information obtained with Stringbuilder to each other. We print at the end.

✓ View Profile

```
public void viewProfile(Account obj){
    System.out.println(getAccountID(obj));
    System.out.println(getUsername(obj));
    System.out.println(getBirthdate(obj));
    System.out.println(getLocation(obj));
    getFollowingInfo(obj);
    getFollowerInfo(obj);
}
6 usages
public void viewProfile(){
    System.out.println(getAccountID( obj: this));
    System.out.println(getUsername( obj: this));
    System.out.println(getBirthdate( obj: this));
    System.out.println(getLocation( obj: this));
    getFollowingInfo( obj: this);
    getFollowerInfo( obj: this);
}
```

We access and print profile information with the help of other methods.

✓ LogIN/OUT

```
public void login(){
    this.IsItOpen = true;

}
6 usages
public void logout(){
    //logini true false olarak tut
    this.IsItOpen = false;
}
```

Change parameter We will use it in methods check the logged in already.

✓ ADD/DELETE POSTS

```java
public void addPost(Post obj){
    if(this.IsItOpen == false){
        System.out.println("You did not logged in.");
    }
    else{
        int i = 0 ;
        for(;i<100;i++){
            if(this.posts[i]== null){
                this.posts[i]= obj;
                obj.accountID = this.accountID;
                break;
            }
        }
    }
}

public void deletePost(Post obj){
    if(this.IsItOpen == false){
        System.out.println("You did not logged in.");
    }
    else {
        int i = 0;
        for (; i < 100; i++) {
            if (this.posts[i] == obj) {
                this.posts[i] = null;
                break;
            }
        }
    }
}
```

We check that there is an account logged in. We are throwing the object into an empty part of the array.

✓ FOLLOW/UNFOLLOW

```java
public void follow(Account obj){//We add our object to the end of our arraylist.
    if(this.IsItOpen == false){
        System.out.println("You did not logged in.");
    }
    else{
        int i = 0 ;
        for(;i<100;i++){
            if(this.following[i]== null){
                this.following[i]= obj;
                break;
            }
        }
        for(i = 0;i<100;i++){
            if(obj.followers[i]== null){
                obj.followers[i] = this;
                break;
            }
        }
    }
}

1 usage
public void unfollow(Account obj){//We add our object to the end of our arraylist.
    if(this.IsItOpen == false){
        System.out.println("You did not logged in.");
    }
    else{
        int i = 0 ;
        for(;i<100;i++){
            if(this.following[i]== obj){
                this.following[i]= null;
                break;
            }
        }
        for(i = 0;i<100;i++){
            if(obj.followers[i]== this){
                obj.followers[i] = null;
                break;
```

When followed, the following user is recorded as the followed followers.

✓ View Post

```
public void viewPosts(){
    if(this.IsItOpen == false){
        System.out.println("You did not logged in.");
    }
    else{
        int i = 0 ;
        for(;i<100;i++){
            if(this.posts[i]!= null){
                if(this.posts[i].Content==null){
                    System.out.println(this.posts[i].postId+"\t"+this.username+": There is no contents");

                }
                else{
                    System.out.println(this.posts[i].postId+"\t"+this.username+":"+this.posts[i].Content);

                }

            }
        }
    }
}

1 usage
public void viewPosts(Account obj){
    if(this.IsItOpen == false){
        System.out.println("You did not logged in.");
    }
    else{
        int i = 0 ;
        int j = 0 ;
        for(;i<100;i++){
            if(obj.posts[i]!= null) {
                System.out.println(obj.posts[i].postId + "\t" + obj.username + ":" + obj.posts[i].Content);
                j++;
            }
            if(i==99 && obj.posts[i]== null&& j==0){
                System.out.println("There is no post have shared by "+obj.username);
            }
```

It shows the posts made by the users together with the comments about the post. If there is no about section, it indicates this.

✓ View Interactions

```
public void viewInteractions(){
    int j = 0 ;
    int i = 0 ;
    System.out.println(this.username+" Interactions....");
    for(;i<100;i++){
        if(this.posts[i]!= null) {
            this.posts[i].viewLikes();
            this.posts[i].viewComments();
            j++;
        }
        if(i==99 && this.posts[i]== null&& j==0){
            System.out.println("There is no interactions have shared by "+this.username);
        }
    }
}
/**/
```

It provides information about interactions using the viewlike and viewcomments methods. If there is no interaction, it indicates this.

✓ SEND MESSAGES

```
public void sendToInbox(int messageIDs,Message senderIDs,String contents ){
    /*
    if(this.IsItOpen == false){
        System.out.println("You did not logged in.");
    }*/
    //else{
    int i = 0 ;
    for(;i<100;i++){
        if(this.Inbox[i]== null){
            this.Inbox[i] = senderIDs ;
            this.Inbox[i].content=contents;
            break;
            }
        }
    //}
}
1 usage
public void sendToOutbox(int messageIDs,Message receiverIDs ,String contents ){
    else{
        int i = 0 ;
        for(;i<100;i++){
            if(this.Outbox[i]== null){
                this.Outbox[i] = receiverIDs ;
                this.Outbox[i].content=contents;

                break;
            }
        }
    }
```

I am using two methods as sent and received. I complete the incoming messages to the sender and the sender of the outgoing messages by throwing objects into the arrays.

✓ View INBOX and OUTBOX

```
public void viewInbox(){
    if(this.IsItOpen == false){
        System.out.println("You did not logged in.");
    }
    else{
        int j = 0 ;
        int i = 0 ;
        System.out.println(this.username+" inbox Messages....");
        for(;i<100;i++){
            if(this.Inbox[i]!= null) {
                j++;
            }
            if(i==99 && this.Inbox[i]== null&& j==0){
                System.out.println("Inbox is empty for\t"+this.username);
            }
        }
        if(j!=0){
            System.out.println("There is "+j+"Messages for\t"+this.username);
        }
        for(i=0;i<100;i++){
            if(this.Inbox[i]!= null) {
                System.out.println("One Messages from\t"+this.username+":\t"+this.Inbox[i].content);
                j++;
            }
        }
    }

}
```

```
public void viewOutbox(){
    if(this.IsItOpen == false){
        System.out.println("You did not logged in.");
    }
    else{
        int j = 0 ;
        int i = 0 ;
        System.out.println(this.username+" outbox Messages....");
        for(;i<100;i++){
            if(this.Outbox[i]!= null) {
                j++;
            }
            if(i==99 && this.Outbox[i]== null&& j==0){
                System.out.println("outbox is empty for\t"+this.username);
            }
        }
        if(j!=0){
            System.out.println("There is "+j+"Messages to\t"+this.username);
        }
        for(i=0;i<100;i++){
            if(this.Outbox[i]!= null) {
                System.out.println("One Messages to\t"+this.username+":\t"+this.Outbox[i].content);
                j++;
            }
```

The user is informed with the help of the objects kept in the arrays by the login person. This information is given if the boxes are empty.

USE CASE DIAGRAM