Photonest

Prepared for

SE302 Software Engineering



Object Design

1.0

15.05.2020

Feyzullah Berkay Danış

Özay Ezerceli

Nursena Karakulah

Dilara Ünbay

Contents

[1. Introduction 4](#_Toc40083981)

[1.1. Object Design Trade-offs 5](#_Toc40083982)

[1.2. Interface Documentation Guidelines 6](#_Toc40083983)

[1.3. Definitions, Acronyms, and Abbreviations 6](#_Toc40083984)

[1.4. References 6](#_Toc40083985)

[2. Packages 6](#_Toc40083986)

[2.1. User Interface Package Definition 7](#_Toc40083987)

[2.1.1. File organization of code for User Interface Package 7](#_Toc40083988)

[2.1.2. Overview of User Interface package 7](#_Toc40083989)

[2.1.3. Dependencies with other packages 7](#_Toc40083990)

[2.1.4. Usage of User Interface package 7](#_Toc40083991)

[2.2. Profile Management Package 7](#_Toc40083992)

[2.2.1. File Organization of code for Profile Management Package 7](#_Toc40083993)

[2.2.2. Overview of Profile Management Package 8](#_Toc40083994)

[2.2.3. Dependencies with other Packages 8](#_Toc40083995)

[2.2.4. Usage of the Package 8](#_Toc40083996)

[2.3. Account Management Package 8](#_Toc40083997)

[2.3.1. File Organization of code for Account Management Package 8](#_Toc40083998)

[2.3.2. Overview of Account Management Package 8](#_Toc40083999)

[2.3.3. Dependencies with other Packages 8](#_Toc40084000)

[2.3.4. Usage of the Package 9](#_Toc40084001)

[2.4. Post Management Package 9](#_Toc40084002)

[2.4.1. File Organization of code for Post Management Package 9](#_Toc40084003)

[2.4.2. Overview of Post Management Package 9](#_Toc40084004)

[2.4.3. Dependencies with other packages 9](#_Toc40084005)

[2.4.4. Usage of the Package 9](#_Toc40084006)

[2.5. Comment Management Package 9](#_Toc40084007)

[2.5.1. File Organization of code for Comment Management Package 9](#_Toc40084008)

[2.5.2. Overview of Comment Management Package 10](#_Toc40084009)

[2.5.3. Dependencies with other Packages 10](#_Toc40084010)

[2.5.4. Usage of the Package 10](#_Toc40084011)

[2.6. Post Storage Package 10](#_Toc40084012)

[2.6.1. File Organization of code for Post Storage Package 10](#_Toc40084013)

[2.6.2. Overview of Post Storage Package 10](#_Toc40084014)

[2.6.3. Dependencies with other packages 10](#_Toc40084015)

[2.6.4. Usage of the Package 11](#_Toc40084016)

[2.7. Account Storage Package 11](#_Toc40084017)

[2.7.1. File Organization of code for Account Storage Package 11](#_Toc40084018)

[2.7.2. Overview of Account Storage Package 11](#_Toc40084019)

[2.7.3. Dependencies with other Packages 11](#_Toc40084020)

[2.7.4. Usage of the Package 11](#_Toc40084021)

[3. Class Interfaces 11](#_Toc40084022)

[3.1. Post Class 11](#_Toc40084023)

[3.1.1. Attributes : 12](#_Toc40084024)

[3.1.2. Operations : 12](#_Toc40084025)

[3.1.3. Dependencies with other classes and packages: 13](#_Toc40084026)

[3.1.4. Exceptions Raised and Exception Handling: 14](#_Toc40084027)

[3.2. Location Class 14](#_Toc40084028)

[3.2.1. Attributes: 14](#_Toc40084029)

[3.2.2. Operations: 14](#_Toc40084030)

[3.2.3. Dependencies with other classes: 15](#_Toc40084031)

[3.2.4. Exceptions Raised and Exception Handling: 15](#_Toc40084032)

[3.3. Coordinate Class 15](#_Toc40084033)

[3.3.1. Attributes : 16](#_Toc40084034)

[3.3.2. Operations: 16](#_Toc40084035)

[3.3.3. Dependencies with other classes: 16](#_Toc40084036)

[3.3.4. Exceptions Raised and Exception Handling: 16](#_Toc40084037)

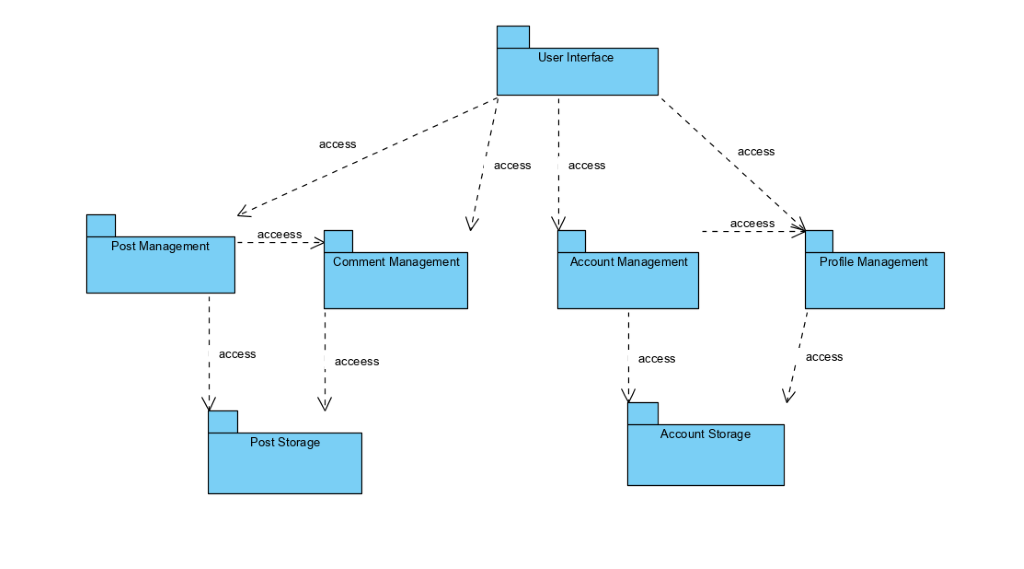
Photonest Object Design Document

# Introduction

Photonest was programmed for people to share pictures online with other people. It is a social sharing application where users can upload and share posts and view posts uploaded by other users, a post would need to include an image and also users can add location information and hashTags to the Post. Users can also see others users posts, make comments on them or like their posts using like function. Users can follow each other which this would inform the users when the other user shares a new post. Each users will have a profile in the system which they can edit their profile information or delete their profile.

Objective of this system is to provide an application that satisfies user needs and requests, and provide every function we have mentioned in the RAD document. The application needs to be simple and enjoyable for users but still effective, therefore user interface of the system has been simplified as much as possible. GUI is made of page, activity and fragment views and functions. The application will not be usable for unregistered users, after creating an account and logging in to the system, registered users will be able to use the application at full functionality. Account information will be kept in firebase realtime database in private, the application has security measures to ensure that the privacy constraints are satisfied according to universal privacy policy.

The subsystem decomposition from which the systems and objects in this document were developed is summarized in the figure below:



## Object Design Trade-offs

During the object design of our project we had make several design trade-offs. One of them was deciding between writing the code ourselves or outsource it. We have written almost all the code by ourselves, we used firebase methods during authentication, change password and forgot password functions.

Another trade-off decision we made was between memory space and response time of the application, since we wanted to make this application enjoyable by users we decided to choose response time over memory space because in a mobile application like Photonest a slow response time could make user dislike the app.

We had to decide between delivery time and functionality and we decided to give delivery time more priority. Since we had a certain delivery date, we decided to add as much functionality to the system as we can and deliver it on time.

## Interface Documentation Guidelines

In order to create better communication between developers during object design we developed some guidelines for interface documentation. We used singular nouns or phrases for class names that included activity or fragment at the end of the name to indicate weather the class is an activity or fragment and every word in the name started with a capital letter. All the objects created were derived from the class diagram and use cases in the RAD. When an exception is caught, the error is displayed to the user.

## Definitions, Acronyms, and Abbreviations

RAD : Requirement Analysis Document

Post : An object that contains an image, hashTags, location which users can view.

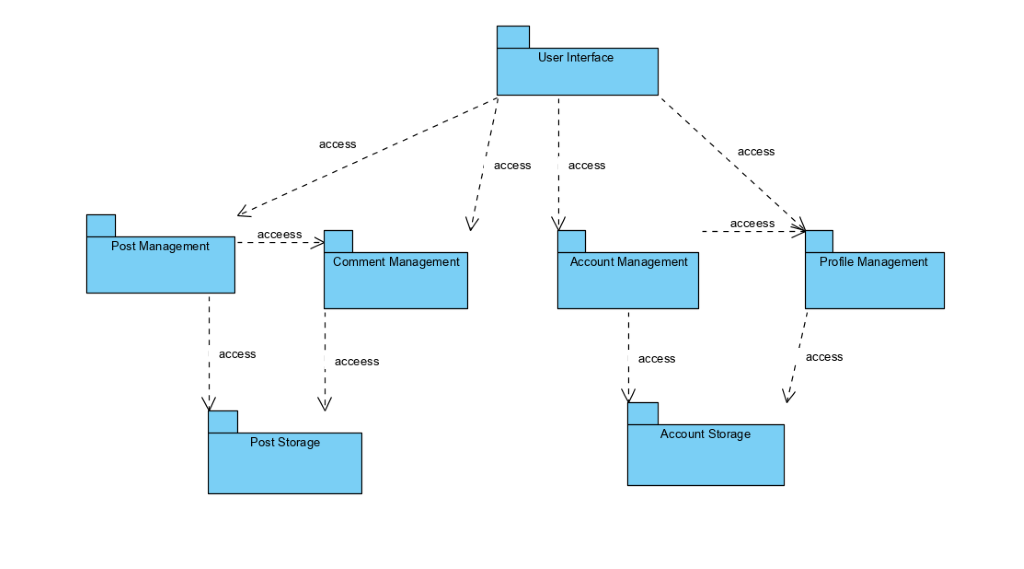
Location : Information of place where a Post was created at.

GUI : Graphical User Interface.

## References

* Instagram

# Packages



## User Interface Package Definition

### File organization of code for User Interface Package

This package contains the following files,

res/

,which this file includes the files,

* + anim/ : This file contains the xml files for animations needed in package.
  + drawable/ : This file contains the xml/png/jpeg files for icons and pictures needed in the package.
  + layout/ : This file contains xml files for templates of pages needed in the package.
  + menu/ : This file contains xml files of menu designs’ needed in the package.
  + mipmap/ : This file contains design files of launcher icon.
  + values/ : This file contains references to reach values of multiple used string, color, styles.

### Overview of User Interface package

This package is responsible of showing reactions received from controller subsystem after the user performed a function. Any data in the model subsystem will be displayed to the user through this package.

### Dependencies with other packages

User interface package contains layout files that are independent from other packages, this package will display reactions received from other packages.

### Usage of User Interface package

User interface package will contain the entire GUI and it will provide design layout of every page, activity and fragment of the application that user interacts with. Also it will provide images and icons that were used in application which makes the application visually interesting, menus that users use to navigate through the activities and packages.

## Profile Management Package

### File Organization of code for Profile Management Package

This package will include the following files:

* + com.302.photonest/

This file contains activity and fragment classes that has the functions provided by the Profile management package in them.

### Overview of Profile Management Package

The Profile Management Package is responsible with handling functions that are related with profile information. It retrieves all user information from firebase database.

### Dependencies with other Packages

The Profile Management Package is related with Account Management and User Interface. When a user needs to be reached from firebase database user information is first received from Account Management. When a user wants to perform a function provided by this package user interacts with the function through the User Interface Package.

### Usage of the Package

Profile Management Package is used for user profile functions such as edit profile, change profile photo, delete profile photo. This package is also used to retrieve user data from fireabase and display it to user.

## Account Management Package

### File Organization of code for Account Management Package

This package will contain the following files:

* + com.302.photonest/

This file includes activity and fragment classes that contain code for functions provided by this package.

### Overview of Account Management Package

Account management handles authentication and other account related functions such as register, login, forgot password, change password logout. It retrieves user’s account information from Account Storage Package.

### Dependencies with other Packages

Account Management package is dependent on the User Interface Package and Account Storage Package. Users interact with the functions through User Interface Package and Account Management Package retrieves user account information from Account Storage.

### Usage of the Package

Account Management Package handles account related functions, it retrieves user input from User Interface package and updates data of Account Storage Package and perform appropriate action according to user input.

## Post Management Package

### File Organization of code for Post Management Package

Post Management Package contains the following files,

* + com.302.photonest/

This file contains activity and fragment classes that contains functions provided by this package.

### Overview of Post Management Package

Post Management Package handles functions related to posts and applies changes to post data on firebase realtime database.

### Dependencies with other packages

Post Management package is dependent on User Interface Package and Post Storage Package. It retrieves user input from User Interface to perform appropriate function and accesses post data from Post Storage Package.

### Usage of the Package

Post Management Package performs functions of creating, deleting and editting posts. It receives the required post data coming from user through the User Interface package and updates the post data on Post Storage Package.

## Comment Management Package

### File Organization of code for Comment Management Package

Comment Management Package contains the following files,

* + com.302.photonest/

This file contains activity and fragment classes that contains functions provided by this package.

### Overview of Comment Management Package

Comment Management Package handles comment related functions and accesses to comment data on Post Storage Package.

### Dependencies with other Packages

Comment Management Package dependent on Post Storage, User Interface and Post Management. Comment Management Package receives comment data from Post Storage Package, it receives user input from User Interface Package before applying appropriate function, while using functions provided by the package comments reaches to post data through Post Management Package.

### Usage of the Package

Comment Management Package handles comment related functions such as adding or deleting comment.

## Post Storage Package

### File Organization of code for Post Storage Package

Files related to this package does not exist in application they are held on firebase realtime database.

### Overview of Post Storage Package

Post Storage Package holds data about posts and comments that are added by users. Application acceses this package for post and data.

### Dependencies with other packages

Post Storage Package is dependent on Comment Management Package and Post Management Package. Post Storage Package receives input from Comment Management before updating comment data and receives input from Post Management before updating post data.

### Usage of the Package

This package holds all the information about posts and comments, other packages reaches to Post Storage Package for post and comment data.

## Account Storage Package

### File Organization of code for Account Storage Package

Files related to this package does not exist in application they are held on firebase realtime database.

### Overview of Account Storage Package

Account Storage Package holds data about user accounts and profiles that are created by users such as username, email, password-are kept as encrypted-, full name.

### Dependencies with other Packages

Account Storage package is dependent on Profile Management and Account Management packages. This package receives input from Profile Management Package before updating profile data of user, it receives input from Account Management Package before updating account data.

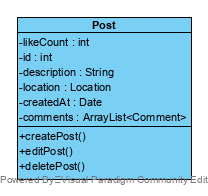
### Usage of the Package

Account Storage is used for holding profile and account related data. When a data about profile or account needs to be reached other packages access this Package.

# Class Interfaces

## Post Class

The Post class allowes users to create new posts , edit existing posts and change their information or delete existing posts. Post class is as follows :



**Class Name : Post**

### Attributes :

* + *-Likecount : int*

Likecount is value of type int and a private attribute. It shows the number of likes the Post object has.

* + -id : int

Id is an integer type value, it is an identification number, it is unique for every post object. It is a private attribute. It distinguishes every post object created from each other.

* + -description : String

Description is a string type value and a private attribute. It holds the hashTags the post has.

* + -location : Location

location is Location type value and a private attribute. It holds the location the post was created at.

* + -createdAt : Date

createdAt holds a Date type value and it is a private attribute. It holds the date that the post created at.

* + -comments : ArrayList<Comment>

Comments is an ArrayList for comments and it is a private attribute. It holds the list of comments that were made for the post.

### Operations :

* + +createPost() :

CreatePost() is a void type of function, it is a public function. When this function is invoked, a new post will be created and added to the firebase database.

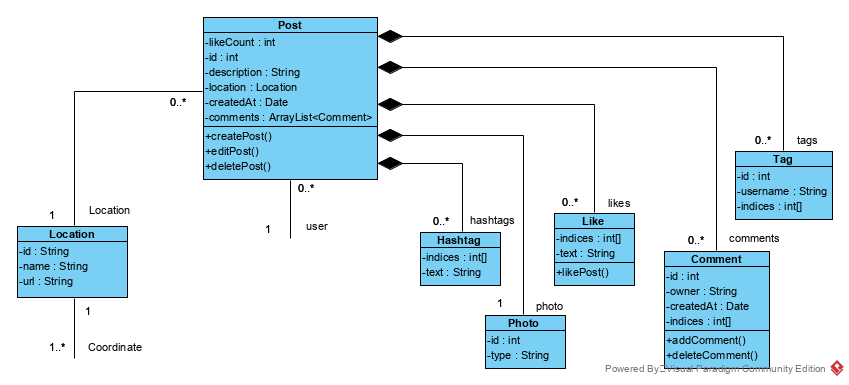
* + +editPost() :

EditPost() is a void type of function that does not return any value. It is a public function. When this function is invoked the information of the choosen post such as it’s description and location is changed.

* + +deletePost() :

DeletePost() is a void type of function that does not return any value. It is a public function. When this function is invoked, the choosen post will be deleted from the firebase database based on the given post id.

### Dependencies with other classes and packages:



The Post class interacts with Location class, Hashtag class, Photo class, Like class, Comment class, Tag class and Post Storage. When a user creates a post if the user wants to add a hashTag to the post user creates a new Hashtag or uses an existing one, therefore Post interacts with the Hashtag class.While a user is creating a post, if the user wants he/she can add location to the post, therefore Post class interacts with the Location class. While a user is creating a post, user needs to pick a photo from his/her device for post image, therefore Post class interacts with Photo class. When a user likes a post, a like object is created with reference to the post, therefore Post class interacts with the Like class. When a user makes a comment to a post, a comment objects is created with reference to the post, therefore Post class interacts with the Comment class. All the information about the posts are kept in Post Storage package, so for a user to do a Post related function such as creating, editting or deleting, Post class needs to interact with the Post Storage package.

### Exceptions Raised and Exception Handling:

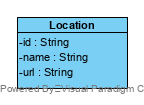
When unexpected(not due to user error) failures occures during creating or deleting post process, a message is displayed on the screen informing user of the failure.

While creating a post, location and caption data can be empty, this was handled using NullPointer exception handling.

While adding location if current location is null, a message is displayed informing the user that since current location is null, a default location will be used.

## Location Class

Location class is reached during post creation. If a user wants to add location information to the post he/she created, a Location object is created for the procedure. Location class is as it seem below.



**Class Name : Location**

### Attributes:

* + -id : int

A private attribute of String type. This attribute holds the ID number of the Location object, used for distinguishing Location objects and it is unique for every Location object.

* + -name : String

A private attribute of String type. This attribute holds the name of the Location.

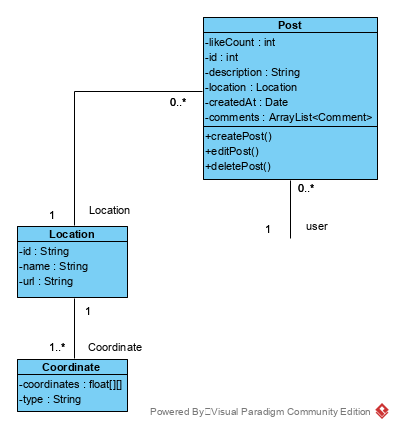
* + -url : String

A private attribute of String type. This attribute holds the URL of the post image that Location will be added to.

### Operations:

This class does not have any operations, since it’s only use is to be added to Post object.

### Dependencies with other classes:



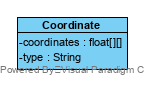
The Location class is in interaction with Post class and Coordinate class. Location objects are created during post creation if user wants to add location to the post. Therefore Location object needs a Post object for creation. The Location class interacts with Coordinate class. While the location object is being created, it retrieves coordinate information from a Coordinate object.

### Exceptions Raised and Exception Handling:

This class does not raise any exceptions because it does not have any functions that provide service inside the class.

## Coordinate Class

Coordinate class is used when a Location object is created for getting the coordinates of location. Coordinate class is given below.



**Class Name : Coordinate**

### Attributes :

* + -coordinates : float[][]

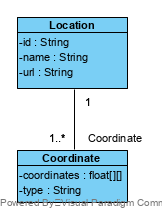
It is a private attribute of float array type. It holds the coordination values of the object.

* + -type : String

### Operations:

This class does not contain any functions inside.

### Dependencies with other classes:



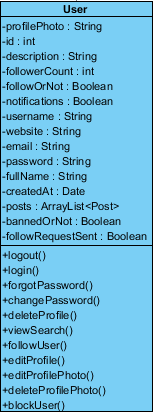
Coordinate class interacts with Location class, while a Location object is being created, coordinate information is gathered using a Coordinate class, therefore a Location class needed before creating a Coordinate class.

### Exceptions Raised and Exception Handling:

There is no exception handling in this class since it does not provide any functions.

## User Class

The user class allows a user to perform certain operations in order to run the system. A user can edit his/her profile, view profile, login and logout, change password, delete account, search other users and hashtags, follow other users and block users. The User class is as follows:



**Class Name: User**

### Attributes :

* *-profilePhoto : String*

profilePhoto is one of the user’s information of type string and it is a private attribute. Every user can have his/her own profile photo for his/her account.

* *-id : int*

Id is a user identification number of type positive integer. It is a private attribute. It is used in order to distinguish the user from other users.

* *-description : String*

Description is one of the user’s information of type string and it is a private attribute. Every user can have his/her own description on his/her profile and every user can see each other’s description by visiting user’s profile.

* *-followerCount : int*

followerCount is one of the user’s information of type positive int and it is a private attribute. This information is shown at his/her profile and updated when user follows other users.

* *-followOrNot : Boolean*

followOrNot is one of the user’s information of type boolean and it is a private attribute. With this information user’s follower and following information can be showed on user profile and other users can see each other’s follower and following information.

* *-notifications : Boolean*

Notifications is one of the user’s information of type boolean and it is a private attribute. This information provides users to get notification when user’s any post or comment liked.

* *-username : String*

Username is a user identification number of type string. It is a private attribute. It is used in order to authenticate the user with the system.

* *-website : String*

Website is one of the user’s information of type string and it is a private attribute. It is showed on user profile and can be edited. This text is cliable and directs user to that link.

* *-email : String*

Email is one of the user’s information of type string and it is a private attribute. It is used for some tasks like when user forgot the password and in order to authenticate the user with the system. He/she can get reset password link via his/her email.

* *-password : String*

Password is a user identification number of type string. It is a private attribute. It is used in order to authenticate the user with the system.

* *-fullName : String*

fullName is one of the user’s information of type string and it is a private attribute. It is showed user’s profile, on posts, comments and search pages.

* *-createdAt : Date*

createdAt is one of the user’s information of type Date and it is a private attribute. When visitor is registered and become a registered user, there is an information that stores the date the creation of user account.

* *-posts : ArrayList<Post>*

Posts is one of the user’s attributes of type arraylist takes a Post object. It is a private attribute. It keeps the user posts and these are showed on each user’s profile. Also if you follow any user, this provides user to see posts of users that person follows.

* *-bannedOrNot : Boolean*

bannedOrNotis one of the user’s information of type boolean and it is a private attribute. It provides user to block other users and so banned person can’t see this user’s post and without unlocking the block, banned user can’t follow this user.

* *-followRequestSent : Boolean*

This attribute is deprecated. In our mobile application anyone can follow someone without sending a follow request.

### Operations:

* *+logout()*

logout() is a public function which is used for finalize the user’s session. When a user wants to exit from the system. He/she must use this function to finalize his/her session.

* *+login()*

login() is a public function which is used for authentication of user. When a user wants to enter the system, he/she must use this function. To access user related functions, first user is needed to be logged in.

* *+forgotPassword()*

forgotPassword() is a public function which is used for adjusting new password when user forget his/her account password. It can be done by tying an email and clicking the link in the email and then user can determine his/her new password for account.

* *+changePassword()*

changePassword() is a public function that does not return a value. When this function is called, user’s password would be change according to his/her input for new password.

* *+deleteProfile()*

deleteProfile() is a public function which is used for deleting user account. When a user wanted to delete his/her account, this can be done on user profile menu by choosing delete account and clicking okay button and then user related information would be deleted on the system.

* *+viewSearch()*

deleteProfile() is a public function which is used for searching user or hashtag. Then user can go directly to searched users profile or see the list of posts that has specific hashtag.

* *+followUser()*

followUser() is a public function which is used for following other users on the system. So, user who is following one another can see the followed user’s posts on his main page.

* *+editProfile()*

editProfile() is a public function which is used for editing editable user information that is showed on user profile. Bio, website, name and username can be changed by using edit profile function.

* *+editProfilePhoto()*

editProfilePhoto() is a public function which is used for adding new profile photo or changing the current one.

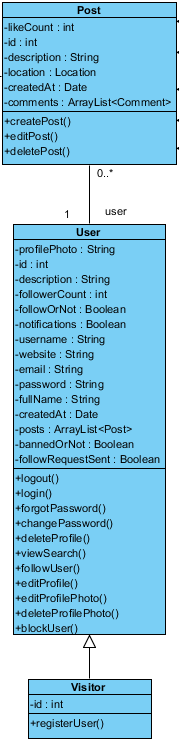
* *+deleteProfilePhoto()*

deleteProfilePhoto() is a public function which is used for deleting current profile photo. When a user delete his/her profile photo default photo would be adjusted instead until he/she put the new profile photo.

* *+blockUser()*

blockUser() is a public function which is used for blocking any user for that account so banned user can’t see the post’s of user who banned himself/herself and can’t follow that user either.

### Dependencies with other classes:



The User class interacts with Post and Visitor class. User is a class that visitor type of user is inherited from this class and could be created in the system. The User class can login, logout, change password, edit account, view account, delete account, follow user, block user, view search and view users profie. By intereacting with Post class, users can do functions related with post class such as creating a post and uploading this post to system, editing and deleting the post.

### Exceptions Raised and Exception Handling:

There are several exceptions that can be raised by the User class. For example, when user wants to login with username and password that doesn’t exist in user database, system display an error message to user that says username or password is not current.

When a user wants to edit his/her account and the information which is going to be changed may not be valid or wrong. So, User is notified by system says, ‘information that you’re trying to change is not valid’.

## Visitor Class

Visitor class is related with the user. Unregistered user is counted as a visitor and to login and perform other user related functions they need to be register to the system. The Visitor class is as follows:



**Class Name: Visitor**

### Attributes :

* -id : int

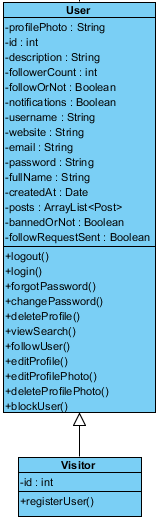
Id is a user identification number of type positive integer. It is a private attribute. It is used in order to distinguish the user from other users.

### Operations:

* +registerUser()

registerUser() is a public function which is used for registiring visitor type of user so when user register to the system he/she can able to use other user related and system functions.

### Dependencies with other classes:



The Visitor class interacts with User class. Visitor class is inherited from user class and have only one function that this type of users can register to system. After registration that user can be able to use other user related functions such as login, create post, edit profile, follow other users etc.

### Exceptions Raised and Exception Handling:

There is an exception that can be raised by the Visitor class when using register function. For example, when user wants to register with information such as username, password, fullname and email, some of them may raise an exception and system display an error messeage to user that says password or email is not valid.

## Hashtag Class

Hashtag class is related with Post class that users can put hashtags to the posts and also users can search specific hashtag to use the posts that has the specific hashtag. The Hashtag class is as follows:



**Class Name: Hashtag**

### Attributes :

* -indices : int[]

indices is one of the hashtag’s attribute of type integer array. It is a private attribute. It keeps the number of hashtags for posts or comments.

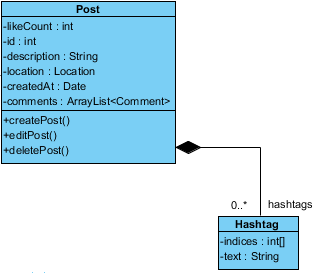
* -text : String

text is one of the hashtag’s attribute of type String. It is a private attribute. It keeps the text of the hashtags in the posts or comments.

### Operations:

There is no operation that specified for this class.

### Dependencies with other classes:



The Hashtag class interacts with Post class. When a user wants to create a post or comment to any post he/she can put hashtags related with post or comment. Also users can click over the hashtags on posts or comments. Thus, hashtag needs to be a part of post class to be created. There is no limit for hashtags but since the caption of post is limited with the exact number of word so hashtag would be as well.

### Exceptions Raised and Exception Handling:

Actually there is no need to exception handling for hashtag class because there is no related function about it. In application just hashtag’s context and number of hashtags are calculated and being used.

## Photo Class

Photo class is related with the Post class that users can select a photo and by creating post they can share their photos. People can comment and like these posts. The Photo class is as follows:



**Class Name: Photo**

### Attributes :

* -id : int

Id is a photo identification number of type positive integer. It is a private attribute. It is used in order to distinguish the photos from other photos.

* -type : String

Type is one of the photo’s attribute of type string that used for determining the type of photo. There could be two possible, either photo is post photo or profile photo.

### Operations:

There is no operation that specified for this class.

### Dependencies with other classes:

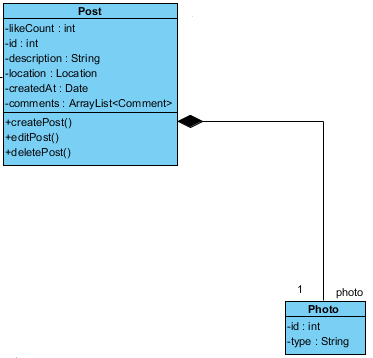


Photo class interacts with Post Class. When a user wants to create a post he/she can select a photo from his/her phone to put the photo into post. Photo class is a part of Post class so post is needed for photo to be showed. One post can only has just one photo.

### Exceptions Raised and Exception Handling:

Actually there may be one exception that can be raised while using the application. Photos of the posts are getting from database to be shown on the main and profile page. So, sometimes since program is faster than database, it needs to be handled by putting progress bar while photo is loading on these pages.

## Like Class

Like class is related with the Post class that users can able to like a post or a comment of the user that followed. The Like class is as follows:



**Class Name: Like**

### Attributes :

* -indices : int[]

indices is one of the hashtag’s attribute of type integer array. It is a private attribute. It keeps the number of likes for posts or comments.

* -text : String

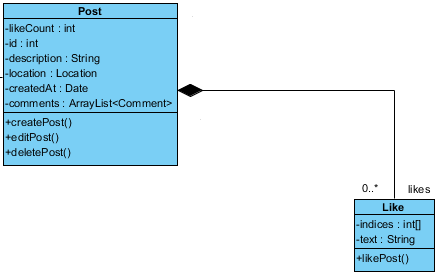
text is one of the like’s attribute of type String. It is a private attribute. It keeps the text of the number of likes in the posts or comments.

### Operations:

* +likePost()

likePost() is a public function which is used for liking posts or comments. A user can like posts of people who is followed by that user. Also this user can like any comment written for these posts.

### Dependencies with other classes:



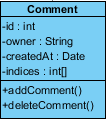
Like class interacts with Post class. When a user wants to like post or comment there is needed to have at least a post. So, like needs to be a part of post class to be created. Users can see the list of likes to see who liked the post. One post can have unlimited likes so there is no limit for likes.

### Exceptions Raised and Exception Handling:

There is no exception that is raised by like class but there is case that we need to check if the position of the post is true with the one we liked because it can be create such problem that post we actually liked and the one is liked could be different. By adding little checking before liking the post we could avoid from this problem.

## Comment Class

Comment class is related with the Post class. It provides users to write a comment to a post as much as the user wants. Also, users can see each others comments. The Comment class is as follows:



**Class Name: Comment**

### Attributes :

* -id : int

Id is a comment identification number of type positive integer. It is a private attribute. It is used in order to distinguish the comment from other comments.

* -owner : String

Owner is one of the comment’s attributes of type string and it’s private. It keeps the information of which user is commented and whose this comment. So, we can list the comments according to users. This information is also useful for the comment like.

* -createdAt : Date

createdAt is one of the comment’s attributes of type Date and it’s private. It keeps the information of when the comment is created so that we can showed the list of comments in a way of sorted.

* -indices : int[]

indices is one of the comment’s attribute of type integer array. It is a private attribute. It keeps the number of comments for posts.

### Operations:

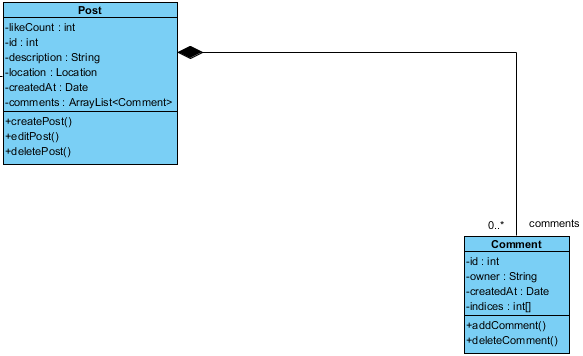
* +addComment()

addComment() is a public function which is used for adding comment to any user’s posts that the user follows. A user can make countless comment to post and other people can see these comments.

* +deleteComment()

deleteComment() is a public function which is used for deleting his/her comment from any user’s posts that the user follows. When a user delete his/her comment from a post then, other people can’t able to see the comment as well.

### Dependencies with other classes:



Comment class interacts with Post class. When a user wants to write a comment to a post there is needed to have at least one post. So, comment needs to be a part of post class to be created. Users can see the list of comments to see who commented to the post. One post can have unlimited comments so there is no limit for the number of comment that a user can write.

### Exceptions Raised and Exception Handling:

There is one exception that comment class may raise which is null pointer exception error. When a user want to like a comment in a comment list it’s important to get a correct position of comment to like comment we would. For position value we need to handle null pointer exception by using try catch.

## Tag Class

Tag class is related with Post class. It provides to users to tag other users to the post with their username. The Tag class is as follows:



**Class Name: Tag**

### Attributes :

* -id : int

Id is a tag identification number of type positive integer. It is a private attribute. It is used in order to distinguish the tag from other tags.

* -username : String

Username is one of the tag’s attributes of type String and it’s private. It keeps the username of the tagged user in the photo that on the post.

* -indices : int[]

indices is one of the tag’s attribute of type integer array. It is a private attribute. It keeps the number of tags for posts.

### Operations:

There is no operation that specified for this class.

### Dependencies with other classes:



Tag class interacts with the Post class. User can tag people who is following by this user to the post. So, tag needs to be a part of post class to be created. One post can have many tags at most the number of people the user is following. This may be needed to be limited sometime.

### Exceptions Raised and Exception Handling:

It is same with like and comment function that we may get an exception about the position of the tag that we choose. So we need to handle it by using try catch. Thus, even the position comes null the application won’t be crashed because of this exception.