Mohamed Abdul Rahman

CS 5800 - Software Engineering
Prof. Nima Davarpanah

Final Exam

Question 1 : UML Diagram for Twitter

1. Design Patterns : The following design patterns are used to create the UML diagram for Twitter.

- Proxy
  - **AuthenticationServiceProxy** – In order to check the user login status, we have to call the AuthenticationService frequently, it will be good idea to have a proxy between this to cache the status.
- Mediator
  - **TweetServer** – User cannot directly communicate to other users, they have to use **TweetServer** as a mediator to perform all the operations.
- Observer
  - **User –** can receive tweets in real time since the **User** is a **TweetObserver**, **TweetServer** is the **TweetSubject** which notifies users.
- Factory
  - **TweetFactory –** This will serve as a factory class to build instance or object of **Tweet**
- Handler
  - **AuthenticationHandler (Abstract)** → extended by → **UserRegistrationStatusHandler , UserAuthenticationHandler :** To process the authentication process of a user, the first handler checks the registration state of the user, if its all good then proceeds to the second handler which authenticates the users.
- I have also used following patterns Association / Aggregation / Composition / Inheritance.

2. Functionalities : The following features are available

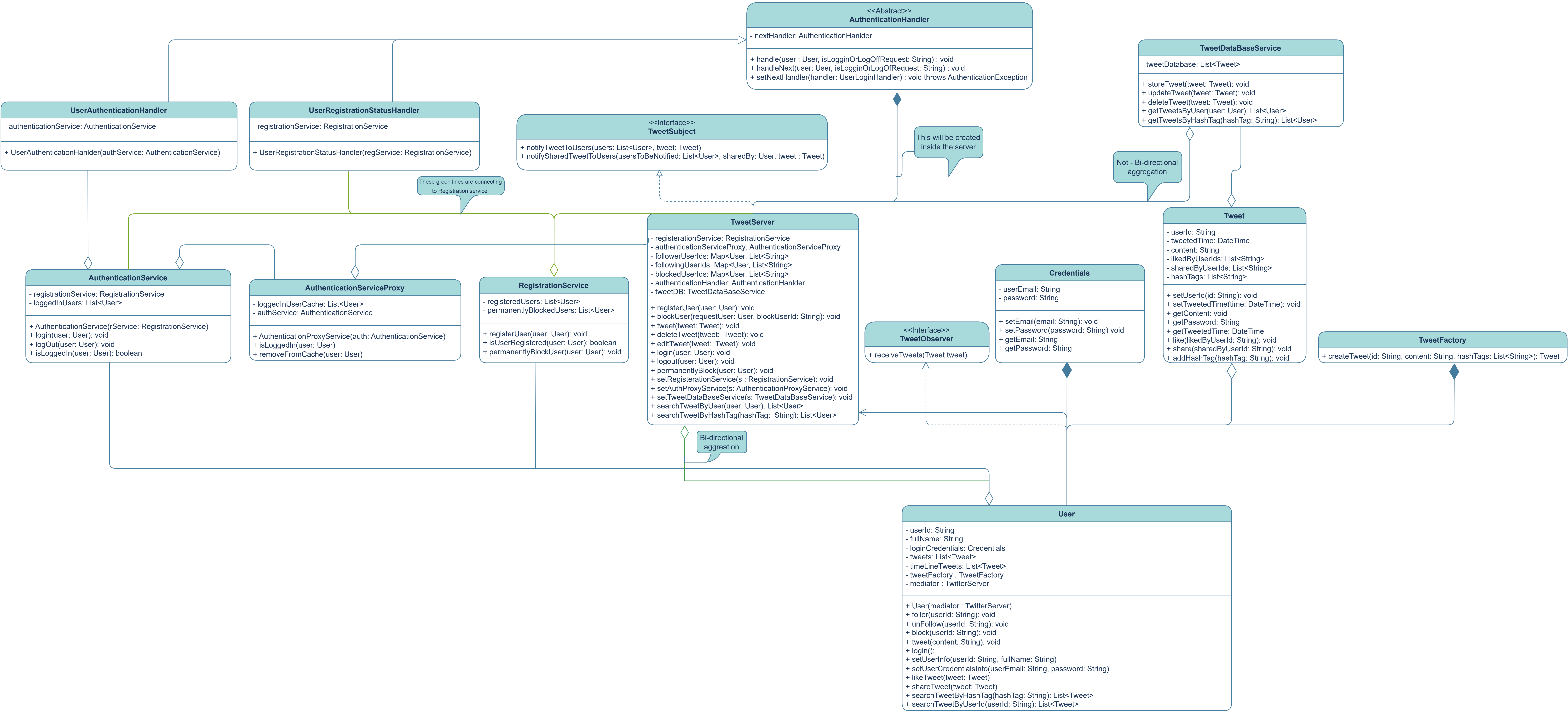
- Send Tweet
- Receive Tweet (real time)
- Delete tweet
- Edit Tweet
- Follow User
- Un-follow User
- Block User
- Register User
- Permanently Block User
- Authentication
- Search By User
- Search By HashTag

Question 1: UML Diagram for Twitter

Following patterns are Used

- Observer
- Proxy
- Factory
- Handler
- Mediator
- Association / Aggregation / Composite / Inheritance

by Mohamed Abdul Rahman



<<Abstract>>
AuthenticationHandler
- nextHandler: AuthenticationHanlder
+ handle(user : User, isLogginOrLogOffRequest: String) : void
+ handleNext(user: User, isLogginOrLogOfRequest: String) : void
+ setNextHandler(handler: UserLoginHandler) : void throws AuthenticationException

TweetDataBaseService
- tweetDatabase: List<Tweet>
+ storeTweet(tweet: Tweet): void
+ updateTweet(tweet: Tweet): void
+ deleteTweet(tweet: Tweet): void
+ getTweetsByUser(user: User): List<User>
+ getTweetsByHashTag(hashTag: String): List<User>

UserAuthenticationHandler
- authenticationService: AuthenticationService
+ UserAuthenticationHanlder(authService: AuthenticationService)

UserRegistrationStatusHandler
- registrationService: RegistrationService
+ UserRegistrationStatusHandler(regService: RegistrationService)

<<Interface>>
TweetSubject
+ notifyTweetToUsers(users: List<User>, tweet: Tweet)
+ notifySharedTweetToUsers(usersToBeNotified: List<User>, sharedBy: User, tweet : Tweet)

This will be created inside the server

Not - Bi-directional aggregation

These green lines are connecting to Registration service

TweetServer
- registerationService: RegistrationService
- authenticationServiceProxy: AuthenticationServiceProxy
- followerUserIds: Map<User, List<String>
- followingUserIds: Map<User, List<String>
- blockedUserIds: Map<User, List<String>
- authenticationHandler: AuthenticationHanlder
- tweetDB: TweetDataBaseService
+ registerUser(user: User): void
+ blockUser(requestUser: User, blockUserId: String): void
+ tweet(tweet: Tweet): void
+ deleteTweet(tweet: Tweet): void
+ editTweet(tweet: Tweet): void
+ login(user: User): void
+ logout(user: User): void
+ permanentlyBlock(user: User): void
+ setRegisterationService(s : RegistrationService): void
+ setAuthProxyService(s: AuthenticationProxyService): void
+ setTweetDataBaseService(s: TweetDataBaseService): void
+ searchTweetByUser(user: User): List<User>
+ searchTweetByHashTag(hashTag: String): List<User>

Tweet
- userId: String
- tweetedTime: DateTime
- content: String
- likedByUserIds: List<String>
- sharedByUserIds: List<String>
- hashTags: List<String>
+ setUserId(id: String): void
+ setTweetedTime(time: DateTime): void
+ getContent: void
+ getPassword: String
+ getTweetedTime: DateTime
+ like(likedByUserId: String): void
+ share(sharedByUserId: String): void
+ addHashTag(hashTag: String): void

AuthenticationService
- registrationService: RegistrationService
- loggedInUsers: List<User>
+ AuthenticationService(rService: RegistrationService)
+ login(user: User): void
+ logOut(user: User): void
+ isLoggedIn(user: User): boolean

AuthenticationServiceProxy
- loggedInUserCache: List<User>
- authService: AuthenticationService
+ AuthenticationProxyService(auth: AuthenticationService)
+ isLoggedIn(user: User)
+ removeFromCache(user: User)

RegistrationService
- registeredUsers: List<User>
- permanentlyBlockedUsers: List<User>
+ registerUser(user: User): void
+ isUserRegistered(user: User): boolean
+ permanentlyBlockUser(user: User): void

Credentials
- userEmail: String
- password: String
+ setEmail(email: String): void
+ setPassword(password: String) void
+ getEmail: String
+ getPassword: String

<<Interface>>
TweetObserver
+ receiveTweets(Tweet tweet)

TweetFactory
+ createTweet(id: String, content: String, hashTags: List<String>): Tweet

Bi-directional aggreation

User
- userId: String
- fullName: String
- loginCredentials: Credentials
- tweets: List<Tweet>
- timeLineTweets: List<Tweet>
- tweetFactory : TweetFactory
- mediator : TwitterServer
+ User(mediator : TwitterServer)
+ follor(userId: String): void
+ unFollow(userId: String): void
+ block(userId: String): void
+ tweet(content: String): void
+ login():
+ setUserInfo(userId: String, fullName: String)
+ setUserCredentialsInfo(userEmail: String, password: String)
+ likeTweet(tweet: Tweet)
+ shareTweet(tweet: Tweet)
+ searchTweetByHashTag(hashTag: String): List<Tweet>
+ searchTweetByUserId(userId: String): List<Tweet>