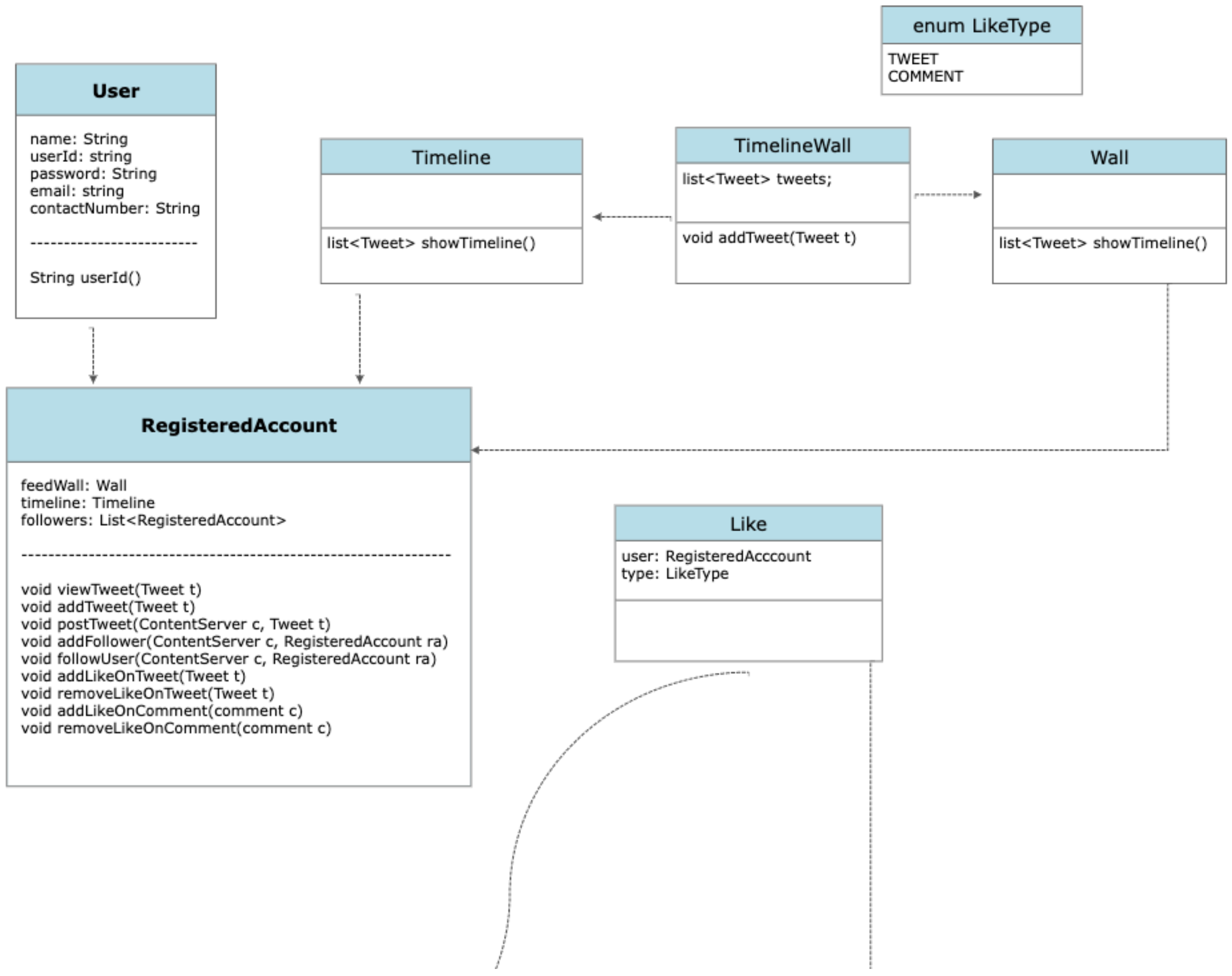
Summary

INTRODUCTION

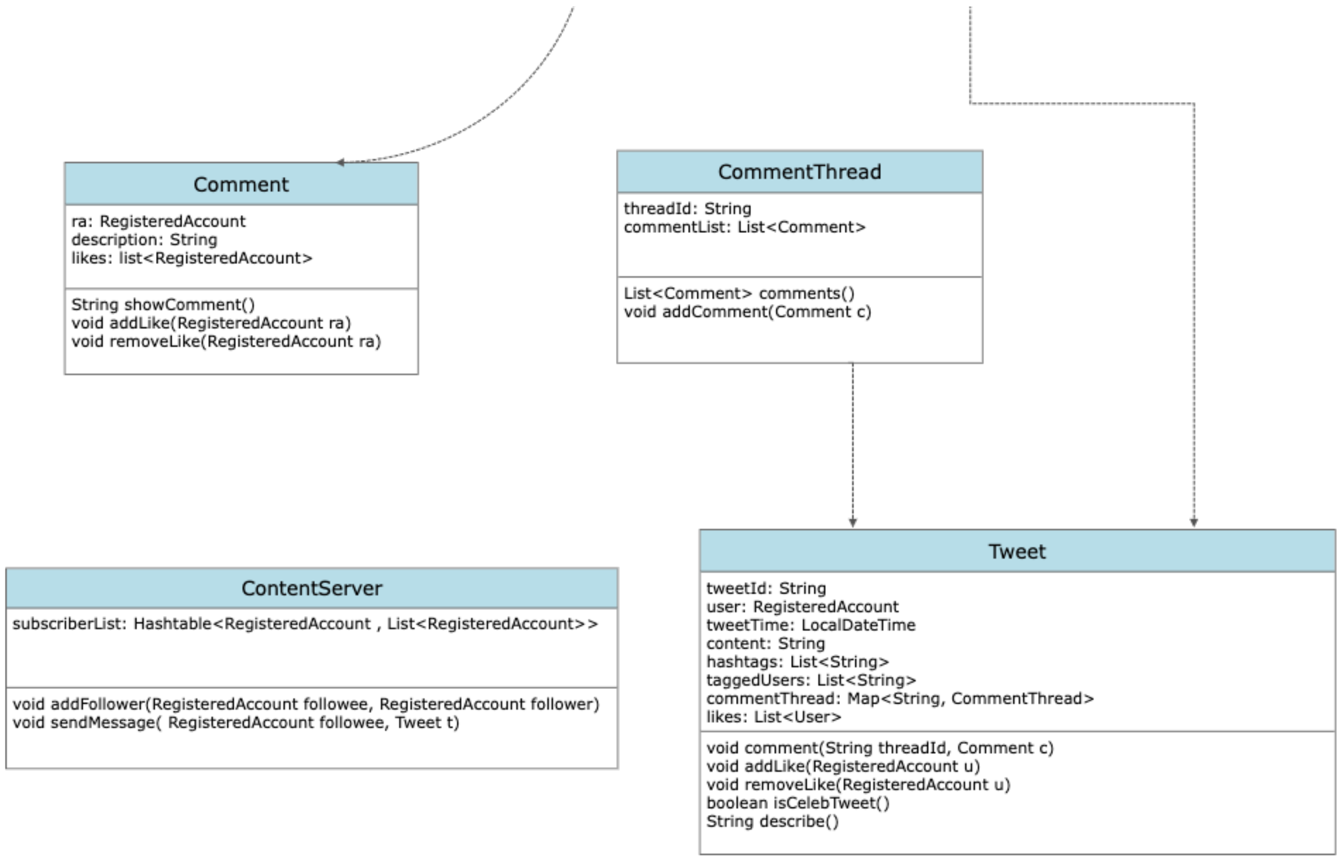
This social media design is very much similar to

1. TWITTER
2. FACEBOOK
3. INSTAGRAM
4. TIC-TOC
5. YOUTUBE
6. LINKEDIN

CLASS DIAGRAM


enum LikeType
TWEET
COMMENT
User
name: String
userId: string
password: String
email: string
contactNumber: String

String userId()
Timeline
list<Tweet> showTimeline()
TimelineWall
list<Tweet> tweets;
void addTweet(Tweet t)
Wall
list<Tweet> showTimeline()
RegisteredAccount
feedWall: Wall
timeline: Timeline
followers: List<RegisteredAccount>
void viewTweet(Tweet t)
void addTweet(Tweet t)
void postTweet(ContentServer c, Tweet t)
void addFollower(ContentServer c, RegisteredAccount ra)
void followUser(ContentServer c, RegisteredAccount ra)
void addLikeOnTweet(Tweet t)
void removeLikeOnTweet(Tweet t)
void addLikeOnComment(comment c)
void removeLikeOnComment(comment c)
Like
user: RegisteredAcccount
type: LikeType


Comment
ra: RegisteredAccount
description: String
likes: list<RegisteredAccount>
String showComment()
void addLike(RegisteredAccount ra)
void removeLike(RegisteredAccount ra)
CommentThread
threadId: String
commentList: List<Comment>
List<Comment> comments()
void addComment(Comment c)
ContentServer
subscriberList: Hashtable<RegisteredAccount , List<RegisteredAccount>>
void addFollower(RegisteredAccount followee, RegisteredAccount follower)
void sendMessage(RegisteredAccount followee, Tweet t)
Tweet
tweetId: String
user: RegisteredAccount
tweetTime: LocalDateTime
content: String
hashtags: List<String>
taggedUsers: List<String>
commentThread: Map<String, CommentThread>
likes: List<User>
void comment(String threadId, Comment c)
void addLike(RegisteredAccount u)
void removeLike(RegisteredAccount u)
boolean isCelebTweet()
String describe()

ENUMS

1. A User can like a Comment and can also like a Post. So below we have a PostType

```
public enum PostType{
  TWEET,COMMENT
}

public enum LikeType{
  HEART,THUMBS_UP,AWESOME,SAD
}
```

CLASSES

USER

These are the basic entities of a user which can be extended wherever we want to.

```
public class User {
        private String name;
        private String userId;
        private String password;
        private String email;
        private String contactNumber;
}
```

LIKE

A like is an object that belongs to a registeredAccount and can be given to any type: TWEET or Comment

```
public class Like{
  private RegisteredAccount user;
  private PostType type;
  private LikeType likeType;

  public Like(RegisteredAccount u, PostType type, LikeType likeType)
  {
    this.user=u;
    this.type=type;
    this.likeType=likeType;
  }
}
```

COMMENT

This is a comment object that a registeredAccount can post.

```
public class Comment {
  private RegisteredAccount ra;
  private String description;
  private List<Like> likes= new List<Like>;

  public Comment(RegisteredAccount ra, String description){
    this.ra= ra;
    this.description= description;
  }

  public String showComment(){
    return description;
  }

  public void addLike(RegisteredAccount u, LikeType likeType){
                likes.add( new Like(u,COMMENT,likeType) );
        }
        public void removeLike(RegisteredAccount u){
                likes.remove(  ); // remove like where user of like is
u;
        }
}
```

COMMENT THREAD

This is a class because we can have comments on comments. So that is why we have a list of comments.

```
public class CommentThread{
  private String threadId;
  ArrayList<Comment> commentList;


  public CommentThread(String threadId){
    this.threadId= threadId;
    commentList= new ArrayList<Comment>;
  }
  public List<Comment> comments(){
    return this.commentList;
  }

  public void addComment(Comment c){
    commentList.add(c);
  }
}
```

TWEET

This tweet class can be tweaked to any type of post, be it an image-type of post on Facebook, or a video-type post on Instagram. But here we are considering a text-type post on Twitter.

```
public class Tweet{
    private String tweetId;
        private RegisteredAccount user;
        private LocalDateTime tweetTime;
        private String content;
        private List<String> hashtags;
        private List<String> taggedUsers;
        private Map<String, CommentThread> commentThreads = new
HashMap<String, CommentThread>();
        private List<RegisteredAccount> likes= new
List<RegisteredAccount>;

        public Tweet(String content,List<String> hashtags,List<String>
taggedUsers){
                this.content= content;
                this.hashtags=hashtags;
                this.taggedUsers=taggedUsers;
        }

        public void comment(String threadId, description d,
RegisteredUser ra) {
                commentThreads.putIfAbsent(threadId, new CommentThread
(threadId));
                commentThreads.get(threadId).addComment( new Comment(
ra, d ) );
        }

        public void addLike(RegisteredAccount u, LikeType likeType){
                likes.add( new Like(u,TWEET, likeType) );
        }
        public void removeLike(RegisteredAccount u){
                likes.remove(  ); // remove like where user of like is
u;
        }

        public boolean isCelebTweet() {
                return this.user.isCelebrity();
        }

        public String describe() {
                StringBuilder br = new StringBuilder();
                br.append(user.userId() + Constants.LINE_BREAK);
                br.append("Tweeted At : " + tweetTime + Constants.
LINE_BREAK);
                br.append(content + Constants.LINE_BREAK);
                br.append(hashtags + Constants.LINE_BREAK);
                br.append(taggedUsers + Constants.LINE_BREAK);
                br.append(commentThreads.values());
                return br.toString();
        }
}
```

TIMELINEWALL

This is a common class that a timeline and wall can extend. So later on we need not to re-declare them again.

```
public class TimelineWall{
  list<Tweet> tweets;
  public timelineWall(){
    tweets= new List<Tweet>;
  }
  void addTweet(Tweet t){
    tweets.add(t);
  }
}
```

TIMELINE

This is a timeline that we see when we open an app. This contains tweets that I will be able to see. These tweets will be the tweets that the people whom I follow have posted.

```
public class Timeline extends TimelineWall {
  public list<Tweet> showTimeline()
  {
    return tweets;
  }
  public addCelebTweets( list<Tweet> celebTweets ){
    tweets.addAll(celebTweets)
  }
}
```

WALL

This is the personal wall that has all the tweets which I posted.

```
public class Wall extends TimelineWall {
  public list<Tweet> showWall()
  {
    return tweets;
  }
}
```

CONTENT SERVER

This is the contentServer class

```
public class ContentServer{
  private Hashtable<RegisteredAccount , List<RegisteredAccount> >
subscriberList;
  private Hashtable<RegisteredAccount, List<Tweet>  > celebritiesPost

  public void addFollower(RegisteredAccount followee, RegisteredAccount
follower  ){
    subscriberList.get(followee).add(follower);
  }

  public void sendMessage( RegisteredAccount followee, Tweet t){
    if(!followee.isCelebrity()){
      subscriberList.get(followee).stream().forEach( follower->
follower.getTimeline().addTweet(t) );
    }
    else {
      celebritiesPost[followee].push(t);
    }
  }
}
```

REGISTERED ACCOUNT

This is the registeredAccount that extends the User. They can post a tweet. Then can comment on a tweet and can like a comment and a tweet.

```
public class RegisteredAccount extends User{
    private Wall feedWall;
        private TimeLine timeline;
        private List<RegisteredAccount> followers;
        private List<RegisteredAccount> followings;
        private Boolean isCelebrity;

        public checkCelebrity(){
                return followers.size()> 1000? true : false;
        }
        public void viewTweet(Tweet tweet)
        {
                tweet.describe();
        }
        public void addTweet(Tweet t){
                Timeline.addTweet(t);
        }

        public void postTweet(ContentServer c, Tweet t){
                feedWall.addTweet(t);
                c.sendMessage(this,t);
        }

        public void showTimeline(ContentServer c){
                var nonCelebTweets = timeline.getTweets();
                var celebTweets = [];
                for(var i=0;i<followings.size();i++)
                {
                        if(followings[i].getIsCelebrity){  // getter
function for isCelebrity
                                celebTweets.addAll( c.celebPosts[i]);
                        }
                }
                return nonCelebTweets.addAll(celebTweets);
        }

        public void addFollower(ContentServer c,  RegisteredAccount ra)
        {
                c.addFollower(this,ra);
                followers.add(ra);
        }

        public void followUser(ContentServer c,RegisteredAccount ra){
                c.addFollower(ra, this);
                followings.add(ra);
        }

        // same functions with different LikeTypes
        public addLikeOnTweet(Tweet t , LikeType type ){
                t.addLike(this,TWEET,type);
        }
        public removeLikeOnTweet(Tweet t){
                t.removeLike(this,TWEET);
        }

        // same functions with different LikeTypes
        public addLikeOnComment(comment c, LikeType type){
                c.addLike(this,COMMENT,type);
        }
        public removeLikeOnComment(comment c){
                c.removeLike(this,COMMENT);
        }
}
```