ecm1410_coursework\src\socialmedia\SocialMedia.java

```java
package socialmedia;

import java.io.IOException;
import java.io.File;
import java.io.FileWriter;
import java.util.*;

import java.io.BufferedWriter;
import java.io.BufferedReader;
import java.io.FileReader;




public class SocialMedia implements SocialMediaPlatform {

    ArrayList<Account> accounts = new ArrayList<>(); //List of Accounts
    ArrayList<FeedDenizen> orphanage = new ArrayList<>(); // Comments where the post has been deleted


    //Returns an Account object of an account from an ID given
    public Account findAccountFromID(int id) throws AccountIDNotRecognisedException{

        for (Account i : accounts){
            if(i.getID() == id){
                return i;
            }
        }
        throw new AccountIDNotRecognisedException();
    }

    //Returns an Account object of an account from a given Handle
    public Account findAccountFromHandle(String handle) throws HandleNotRecognisedException{

        for (Account i : accounts){
            if(handle.equals(i.getHandle())){
                return i;
            }
        }
        throw new HandleNotRecognisedException();
    }


    //Returns an Account object of an account from an ID given, only to be used if account ID is known to exist
    public Account findAccountFromKnownID(int id){

        for (Account i : accounts){
            if(i.getID() == id){
                return i;
            }
        }
        return new Account(0, null, null);
    }


    // Unpacks an ID into Account ID and Post ID
    public int[] unpackID(int id){

        int accountID = (int)Math.floor(id/10000);
        int postID = id - (accountID * 10000);
        int[] result = {accountID, postID};
        return result;
    }


    // Returns a post from a given ID
    public FeedDenizen findPostFromID(int id) throws PostIDNotRecognisedException{

        int[] ids = unpackID(id);
        Account account = findAccountFromKnownID(ids[0]);
        FeedDenizen post = account.getPost(ids[1]);
        return post;
    }

    // Returns a post from a given ID, only to be used if account ID is known to exist
    public FeedDenizen findPostFromKnownID(int id){

        int[] ids = unpackID(id);
        Account account = findAccountFromKnownID(ids[0]);
        FeedDenizen post = account.getKnownPost(ids[1]);
        return post;
    }


    // Creates an account with a blank description by calling createAccount with a blank description
    @Override
    public int createAccount(String handle) throws IllegalHandleException, InvalidHandleException {

        return createAccount(handle, "");
    }

    // Validates if the handle is unique and the description follows the rules, and if so creates an account with the next ID and adds it to the account list, then returns the ID
    @Override
    public int createAccount(String handle, String description) throws IllegalHandleException, InvalidHandleException {

        if (!(0 < handle.length() && handle.length() < 31)){
            throw new InvalidHandleException();
        }
        char[] array = handle.toCharArray();
        for (char i : array ){
            int ascii = i;
            if (!(64 < ascii && ascii < 123)){
                throw new InvalidHandleException();
            }
        }
        for (Account i : accounts){
            if (handle.equals(i.getHandle())) {
            throw new IllegalHandleException();
```

```java
                }
        }
        int next_index = accounts.size();
        accounts.add(new Account(next_index, handle, description));
        return next_index;
    }

    // Calls findAccountFromID, if an account is found, calls the delete method on that account, and stores all the children post in the orphanage list, then removes the account from the account list
    @Override
    public void removeAccount(int id) throws AccountIDNotRecognisedException {

        Account i = findAccountFromID(id);
        List<Integer> orphans = i.delete();
        for (Integer childID : orphans){
            orphanage.add(findPostFromKnownID(childID));
        }
        accounts.remove(i);
    }

    // Finds an account and adds all the comments of this account's posts to the orphanage then removes the account and calls the delete account method
    @Override
    public void removeAccount(String handle) throws HandleNotRecognisedException {

        Account i = findAccountFromHandle(handle);
        List<Integer> orphans = i.delete();
        for (Integer childID : orphans){
            orphanage.add(findPostFromKnownID(childID));
        }
        accounts.remove(i);
    }

    // Validates the handle is legal, then iterates through the list of accounts and checks their handle.
    // If the handle matches the new handle, the new handle can't be used so IllegalHandleException is Thrown
    // If the handle matches the old handle, this is the account we want to change the handle of so we store the index for later
    // Once the loop is done, if an index was found for the account, the handle is changed, else it throws HandleNotRecognisedException
    @Override
    public void changeAccountHandle(String oldHandle, String newHandle)
            throws HandleNotRecognisedException, IllegalHandleException, InvalidHandleException {

        if (!(0 < newHandle.length() && newHandle.length() < 31)){
            throw new InvalidHandleException();
        }
        int acc = -1;
        for (int i = 0; i < accounts.size(); i++){
            String currentHandle = accounts.get(i).getHandle();
            if (newHandle.equals(currentHandle)) {
                throw new IllegalHandleException();
            }
            if (oldHandle.equals(currentHandle)) {
                acc = i;
            }
        }

        if (acc > -1) {
            accounts.get(acc).setHandle(newHandle);
            }
        else{
            throw new HandleNotRecognisedException();
        }
    }


    // Uses the findAccountFromHandle method to find the account, and changes the description if one is found
    @Override
    public void updateAccountDescription(String handle, String description) throws HandleNotRecognisedException {

        findAccountFromHandle(handle).setDescription(description);
    }

    // Uses the findAccountFromHandle method to find the account, then returns a formatted string to display the information abut the account
    @Override
    public String showAccount(String handle) throws HandleNotRecognisedException {

        Account i = findAccountFromHandle(handle);
        return  "ID: " + i.getID() + System.lineSeparator() +
                "Handle: " + handle + System.lineSeparator() +
                "Description: " + i.getDescription() + System.lineSeparator() +
                "Post count: " + i.getNumberOf("Post") + System.lineSeparator() +
                "Endorse count: " + i.getNumberOf("Endorsement");
    }

    // Validates the message length, attempts to find the account, calls the post method on the account, then returns the posts ID
    @Override
    public int createPost(String handle, String message) throws HandleNotRecognisedException, InvalidPostException {

        if (!(0 < message.length() && message.length() < 100)){
            throw new InvalidPostException();
        }
        Post post = findAccountFromHandle(handle).post(message);
        return post.getID();
    }

    // Attempts to find the post and account, then validates that it is a post (not comment/endorsement) then calls the endorsee method on the post and returns the endorsement ID
    @Override
    public int endorsePost(String handle, int id) throws HandleNotRecognisedException, PostIDNotRecognisedException, NotActionablePostException {

        FeedDenizen post = findPostFromID(id);
        if (post.getClass().getSimpleName().equals("Post")){
            Endorsement end = findAccountFromHandle(handle).endorse(id);
            post.endorse();
            return end.getID();
        }
        throw new NotActionablePostException();
    }

    // Validates the message length, attempts to find the account, then validates that it is not a endorsement, then calls the comment method on the account, then returns the posts ID
    @Override
    public int commentPost(String handle, int id, String message) throws HandleNotRecognisedException, PostIDNotRecognisedException, NotActionablePostException, InvalidPostException {

        if (!(0 < message.length() && message.length() < 100)){
            throw new InvalidPostException();
        }
        FeedDenizen post = findPostFromID(id);
        if (!post.getClass().getSimpleName().equals("Endorsement")){
            Comment comment = findAccountFromHandle(handle).comment(message, id);

            return comment.getID();
        }
        throw new NotActionablePostException();
    }

    // Finds the post, then calls the delete method on it, and adds all the child comments to the orphanage
    @Override
    public void deletePost(int id) throws PostIDNotRecognisedException {
        int[] ids = unpackID(id);
        Account account = findAccountFromKnownID(ids[0]);
        List<Integer> orphans = account.deletePost(ids[1]);
        for (Integer childID : orphans){
            orphanage.add(findPostFromKnownID(childID));
        }
    }

    // Formats a string to contain all the information for a single post: ID, account, total endorsements, total comments, post text
    @Override
    public String showIndividualPost(int id) throws PostIDNotRecognisedException {
        int[] ids = unpackID(id);
        Account account = findAccountFromKnownID(ids[0]);
        FeedDenizen post = findPostFromID(id);
        return  "ID: " + post.getID() + System.lineSeparator() +
                "Account: " + account.getHandle() + System.lineSeparator() +
                "No. endorsements: " + post.getTotalEndorsements()  + "| No. comments: " + post.getTotalComments() + System.lineSeparator() +
                post.getText();
    }

    // Finds a post from an ID and adds it to a new List. Then it iterates through the list and for each post adds the formatted string from the showIndividualPost method
    // The for each post it checks if it has children, if so it adds the children to the list directly after their parent post and adds one to the indent level
    // It also adds the int 0 to the list after the children it just added. The 0 signifies when to go back down an indent level.
    // When the list is exhausted, the formatted string is returned
    @Override
    public StringBuilder showPostChildrenDetails(int id)throws PostIDNotRecognisedException, NotActionablePostException {

        List<Integer> posts = new ArrayList<>();
        posts.add(id);

        StringBuilder str = new StringBuilder();
        String indent = "   ";

        int indentLevel = 0;
        int index = 0;
        while (index < posts.size()){

            if (posts.get(index) != 0){
                String[] comment = showIndividualPost(posts.get(index)).split(System.lineSeparator());
                for (String line : comment){
                    if ((indentLevel > 0) && (line.substring(0,3).equals("ID:"))){
                        str.append(indent.repeat(indentLevel -1));
                        str.append("| > ");
                        str.append(line);
                        str.append(System.lineSeparator());
                    } else {
                        str.append(indent.repeat(indentLevel));
                        str.append(line);
                        str.append(System.lineSeparator());
                    }
                }


                FeedDenizen post = findPostFromID(posts.get(index));
                List<Integer> children = post.getChildren();
                if (children.size() > 0){
                    str.append(indent.repeat(indentLevel));
                    str.append("|");
                    str.append(System.lineSeparator());

                    children.add(0);
                    posts.addAll(index+1, children);
                    indentLevel += 1;
                }

            } else {
                indentLevel -=1;
            }
            index += 1;
        }
        return str;
    }

    // Returns the number of accounts
    @Override
    public int getNumberOfAccounts() {

        return accounts.size();
    }

    // Iterates through the accounts, and sums the number of post type posts
    @Override
    public int getTotalOriginalPosts() {

        int count= 0;
        for (Account i : accounts){
            count += i.getNumberOf("Post");
        }
        return count;
    }

    // Iterates through the accounts, and sums the number of endorsements type posts
    @Override
    public int getTotalEndorsementPosts() {

        int count= 0;
        for (Account i : accounts){
            count += i.getNumberOf("Endorsement");
        }
        return count;
    }

    // Iterates through the accounts, and sums the number of comment type posts
    @Override
    public int getTotalCommentPosts() {

        int count= 0;
        for (Account i : accounts){
            count += i.getNumberOf("Comment");
        }
        return count;
    }

    // Iterates through the accounts, and finds the account with the most endorsed post
    @Override
    public int getMostEndorsedPost() {

        int previous = 0;
        int previousID = 0;
        for (Account account : accounts){
            int[] current = account.mostEndorsements();
            if (current[0] - previous > 0){
                previous = current[0];
                previousID = current[1];
            }
        }
        return previousID;
    }

    // Iterates through the accounts, and finds the account with the most endorsements
    @Override
    public int getMostEndorsedAccount() {

        int previous = 0;
        int previousID = 0;
        for (Account account : accounts){
            int current = account.totalEndorsements();
            if (current > previous){
                previous = current;
                previousID = account.getID();
            }
        }
        return previousID;
    }

    // Removes all references to accounts and therefore posts. Also removes all
reference to orphan posts
    @Override
    public void erasePlatform() {

        accounts.removeAll(accounts);
        orphanage.removeAll(orphanage);

    }

    // Starts by deleting any existing files with the same filename in the same directory if they exist
    // To the txt file, it writes:
    // For each account it writes the account: ID, handle, description followed by an opening bracket
    //      For each post from that account's timeline it writes post: type, ID, text, total comments, total endorsements, parent ID followed by an opening bracket
    //          For each comment for each post it writes the ID followed by a line (|)
    //      Ends the post
    // Therefore the formatting for one account look like:
    // 34,Maurice Moss,IT support at Reynholm Industries,(P,340009,Did you see that ludicrous display last night?,1,3,0,(560023|)/C,340010,thing about Arsenal is they always try and walk it in,2,2,560023,(560024|470143|)/)
    @Override
    public void savePlatform(String filename) throws IOException {

        try{
            File f = new File(filename);
            if (f.exists()){
                f.delete();
            }
            f.createNewFile();
            FileWriter fWriter = new FileWriter(filename);
            BufferedWriter bWriter = new BufferedWriter(fWriter);
            for (Account account : accounts){
                StringBuffer line = new StringBuffer();
                String[] chars =  {String.valueOf(account.getID()), account.getHandle(), account.getDescription()};
                for (String i : chars){
                    line.append(i + ",");
                }
                line.append("(");
                for (FeedDenizen post : account.getTimeline()){
                    char type = post.getClass().getSimpleName().charAt(0);
                    line.append(type + "," + post.getID() + "," + post.getText() + "," + post.getTotalComments() + "," + post.getTotalEndorsements() + "," + post.getParent() + ",");

                    line.append("(");
                    for (Integer child : post.getChildren()){
                        line.append(child + "|");
                    }
                    line.append(")/");
                }
                line.append(")");
                bWriter.write(line.toString());
                bWriter.flush();
            }
            bWriter.close();
        } catch (IOException e){
            throw new IOException();
        }
    }

    // For each line in the file, it creates an account. For each post on the account it creates the relevant post and adds it to the timeline.
    @Override
    public void loadPlatform(String filename) throws IOException, ClassNotFoundException {

        BufferedReader reader;
        String line = null;
        try{
            reader = new BufferedReader(new FileReader(filename));

            while ((line = reader.readLine()) != null) {
                String[] accDesc = line.split(",", 4);
                String strPosts = accDesc[3].substring(1, accDesc[3].length()-1);
                String[] posts = strPosts.split("/");
                int id = Integer.parseInt(accDesc[0]);
                Account account = new Account(id, accDesc[1], accDesc[2]);
                for (String p : posts){
                    String[] splitPost = p.split(",");

                    if (splitPost[0].equals("P")){
                        Post newPost = account.post(splitPost[2]);
                        int endorsements = Integer.parseInt(splitPost[4]);
                        newPost.endorsements = endorsements;
                        int noComments = Integer.parseInt(splitPost[3]);
                        if (noComments > 0){
                            String strComments = splitPost[6].substring(1, splitPost[6].length() - 1);
                            String[] comments = strComments.split("|");
                            List<Integer> children = new ArrayList<>();
                            for (String i : comments){
                                int y = Integer.parseInt(i);
                                children.add(y);
                            }
                            newPost.children = children;
                        }
                    }
                    if (splitPost[0].equals("C")){
                        int parent = Integer.parseInt(splitPost[5]);
                        Comment newPost = account.comment(splitPost[2], parent);
                        int endorsements = Integer.parseInt(splitPost[4]);
                        newPost.endorsements = endorsements;
                        int noComments = Integer.parseInt(splitPost[3]);
                        if (noComments > 0){
                            String strComments = splitPost[6].substring(1, splitPost[6].length() - 1);
                            String[] comments = strComments.split("|");
                            List<Integer> children = new ArrayList<>();
                            for (String i : comments){
                                int y = Integer.parseInt(i);
                                children.add(y);
                            }
                            newPost.children = children;
                        }
                    }
                    if (splitPost[0].equals("E")){
                        int parent = Integer.parseInt(splitPost[5]);
                        account.endorse(parent);
                    }
                }
            }

            reader.close();

        } catch (IOException e){
            throw new IOException();
        }


    }

}

```

ecm1410_coursework\src\socialmedia\Account.java

```java
package socialmedia;

import java.util.ArrayList;
import java.util.List;

public class Account {
    int ID;
    String handle;
    String description;
    List<FeedDenizen> timeline;

    public Account(int id, String y, String z){
        this.ID = id;
        this.handle = y;
        this.description = z;
        this.timeline = new ArrayList<>();

    }

    public int getID() {
        return this.ID;
    }

    public String getHandle() {
        return this.handle;
    }

    public void setHandle(String str) {
        this.handle = str;
    }

    public String getDescription() {
        return this.description;
    }

    public List<FeedDenizen> getTimeline(){
        return this.timeline;
    }

    // Iterates through the timeline and fins the most endorsed post
    public int[] mostEndorsements(){
        int previous = 0;
        int previousID = 0;
        for (FeedDenizen i : this.timeline){
            if (i.getTotalEndorsements() > previous){
                previous = i.getTotalEndorsements();
                previousID = i.getID();
            }
        }
        return new int[] {previous, previousID};
    }

    // Sums the total amount of endorsements the account has received
    public int totalEndorsements(){
        int sum = 0;
        for (FeedDenizen i : timeline){
            sum += i.getTotalEndorsements();
        }
        return sum;
    }

    public FeedDenizen getPost(int id) throws PostIDNotRecognisedException{
        for (FeedDenizen i : this.timeline){
            if (id == i.getPostID()){
                return i;
            }
        }
        throw new PostIDNotRecognisedException();
    }

    public FeedDenizen getKnownPost(int id){
        for (FeedDenizen i : this.timeline){
            if (id == i.getPostID()){
                return i;
            }
        }
        return null;
    }

    public void setDescription(String str) {
        this.description = str;
    }

    // Finds the total number of elements in the timeline with the same type as the
argument
    public int getNumberOf(String arg){
        int count = 0;
        for (FeedDenizen i : this.timeline){
            String type = i.getClass().getSimpleName();
            if (type.equals(arg)){
                count += 1;
            }
        }
        return count;
    }

    // Deletes all of the posts and collects their children, then clears all the user
data - handle, description and timeline and returns the children
    public List<Integer> delete(){
        List<Integer> orphans = new ArrayList<>();
        for (FeedDenizen i : this.timeline){
            orphans.addAll(1, i.getChildren());
            i.delete();
        }
        this.handle = "[DELETED USER]";
        this.description = "";
        this.timeline = new ArrayList<>();
        return orphans;
    }

    // Creates a post on the timeline
    public Post post(String message){
        Post post = new Post(this.timeline.size(), this.ID, message);
        this.timeline.add(post);
        return post;
    }

    // Creates an endorsement on the timeline
    public Endorsement endorse(int id){
        Endorsement endorsement = new Endorsement(this.timeline.size(), this.ID, id);
        this.timeline.add(endorsement);
        return endorsement;
    }

    // Creates a comment on the timeline
    public Comment comment(String message, int postID){
        Comment comment = new Comment(this.timeline.size(), this.ID, postID, message);
        this.timeline.add(comment);
        return comment;
    }

    // Deletes a post given specific ID
    public List<Integer> deletePost(int id) throws PostIDNotRecognisedException{
        FeedDenizen post = getPost(id);
        List<Integer> orphans = post.getChildren();
        post.delete();
        return orphans;
    }
}
```

ecm1410_coursework\src\socialmedia\FeedDenizen.java

```java
package socialmedia;

import java.util.List;

abstract class FeedDenizen {
    int ID;
    int accountID;

    public void init(int id, int accID){
        this.ID = id;
        this.accountID = accID;
    }

    public int getID(){
        return this.accountID * 10000 + this.ID;
    }

    public int getPostID(){
        return this.ID;
    }

    public int getAccount(){
        return this.accountID;
    }

    abstract List<Integer> getChildren();

    abstract void delete();

    abstract void endorse();

    abstract int getTotalComments();

    public int getParent(){
        return 0;
    }

    public int getTotalEndorsements(){
        return 0;
    }

    public String getText(){
        return null;
    }
}

```

ecm1410_coursework\src\socialmedia\Post.java

```java
package socialmedia;

import java.util.ArrayList;
import java.util.List;

public class Post extends FeedDenizen {
    int endorsements;
    String text;
    List<Integer> children;

    public Post(int id, int accID, String message){
        init(id, accID);
        this.endorsements = 0;
        this.text = message;
        this.children = new ArrayList<>();
    }

    public String getText(){
        return this.text;
    }

    public void endorse(){
        this.endorsements += 1;
    }

    public void addComment(int commentID){
        this.children.add(commentID);
    }

    public void removeComment(int commentID){
        boolean found = false;
        for (int i = 0; (i < children.size()) & !found; i++){
            if (children.get(i) == commentID){
                children.remove(i);
                found = true;
            }
        }
    }

    public int getTotalEndorsements(){
        return this.endorsements;
    }

    public int getTotalComments(){
        return this.children.size();
    }

    @Override
    public List<Integer> getChildren(){
        return this.children;
    }

    @Override
    public void delete(){
        this.accountID = 0;
        this.text = "[POST DELETED]";
    }
}

```

ecm1410_coursework\src\socialmedia\Comment.java

```java
package socialmedia;
import java.util.ArrayList;
import java.util.List;

public class Comment extends FeedDenizen{
    int parentID;
    int endorsements;
    String text;
    List<Integer> children;

    public Comment(int id, int accID, int postID, String message){
        init(id, accID);
        this.parentID = postID;
        this.text = message;
        this.children = new ArrayList<>();
        this.endorsements = 0;
    }

    public int getParent(){
        return this.parentID;
    }

    public int getTotalComments(){
        return this.children.size();
    }

    public void endorse(){
        this.endorsements += 1;
    }

    public String getText(){
        return this.text;
    }

    @Override
    public List<Integer> getChildren(){
        return this.children;
    }

    @Override
    public void delete(){
        this.accountID = 0;
        this.text = "[POST DELETED]";
    }
}

```

ecm1410_coursework\src\socialmedia\Endorsement.java

```java
package socialmedia;
import java.util.ArrayList;
import java.util.List;


public class Endorsement extends FeedDenizen{
    int postID;

    public Endorsement(int id, int accID, int targetID){
        init(id, accID);
        this.postID = targetID;

    }

    public void endorse(){
    }

    public int getParent() {
        return this.postID;
    }

    public int getTotalComments(){
        return 0;
    }


    @Override
    void delete() {
        this.accountID = 0;
    }

    @Override
    List<Integer> getChildren(){
        return new ArrayList<>();
    }
}
```