```java
package socialmedia;

/**
 * Class to create Post-comment objects
 */
public class Comment extends PostWithChildren {
    private int parentID;

    /**
     * method to obtain comments parents ID
     * @return parentID
     */
    public int getParentID() {
        return parentID;
    }

    /**
     * Sets super and parentID
     * @param ID
     * @param userID
     * @param content
     * @param parentID
     */
    public Comment(int ID, int userID, String content, int parentID) {
        super(ID, userID, content);
        this.parentID = parentID;
    }
}

```

```java
package socialmedia;

import java.io.Serializable;

/**
 * Class that represents an account object
 * adds all account functionality
 */
public class Account implements Serializable {
    private String handle;
    private int ID;
    private int postCount;
    private int endorseCount;

    /**
     * method to increment post count by 1
     */
    public void incrementPostCount() {
        this.postCount ++;
    }

    /**
     * method to decrement post count by 1
     */
    public void decrementPostCount() {
        this.postCount --;
    }

    /**
     * method to increment endorse count by 1
     */
    public void incrementEndorseCount() {
        endorseCount ++;
    }

    /**
     * method to increment endorse count by 1
     */
    public void decrementEndorseCount() {
        endorseCount --;
    }
```

```java

    /**
     * method to set the description of an account
     */
    public void setDescription(String description) {
        this.description = description;
    }

    private String description;

    /**
     * method to set handle of an account
     */
    public void setHandle(String handle) {
        this.handle = handle;
    }

    /**
     * method to set accounts ID, handle, postCount and endorseCount
     */
    public Account(int ID, String handle) {
        this.ID = ID;
        this.handle = handle;
        this.postCount = 0;
        this.endorseCount = 0;
    }

    /**
     * method to set accounts ID, handle, description, postCount and endorseCount
     */
    public Account(int ID, String handle, String description) {
        this.ID = ID;
        this.handle = handle;
        this.description = description;
        this.postCount = 0;
        this.endorseCount = 0;
    }
```

```java

    /**
     * method to obtain post count
     * @return postCount
     */
    public int getPostCount() {
        return postCount;
    }

    /**
     * method to obtain endorse count
     * @return endorseCount
     */
    public int getEndorseCount() {
        return endorseCount;
    }

    /**
     * method to obtain handle
     * @return handle
     */
    public String getHandle() {
        return handle;
    }

    /**
     * method to obtain ID
     * @return ID
     */
    public int getID() {
        return ID;
    }

    /**
     * method to obtain description
     * @return description
     */
    public String getDescription() {
        return description;
    }
}
```

```java
package socialmedia;

import java.util.ArrayList;

/**
 * Class to create deleted post objects
 */
public class DeletedPost extends PostWithChildren {
    private static String content = "The original content was removed from the system and is no longer available.";

    /**
     * method to set deleted post variables
     * @param ID
     * @param children
     */
    public DeletedPost(int ID, ArrayList<Integer> children){
        super(ID, 0);
        this.setChildren(children);
    }

    /**
     * method to obtain content
     * @return content
     */
    @Override
    public String getContent() {
        return content;
    }
}
```

```java
package socialmedia;

/**
 * Class to create a Post-endorsement object
 */
public class Endorsement extends Post{
    private static String content = "Post Endorsed";
    private int parentID;

    /**
     * Method to obtain content
     * @return content
     */
    @Override
    public String getContent() {
        return content;
    }

    /**
     * Method to obtain parent ID
     * @return parentID
     */
    public int getParentID() {
        return parentID;
    }

    /**
     * method to set endorsement variables
     * @param ID
     * @param userID
     * @param parentID
     */
    public Endorsement(int ID, int userID, int parentID) {
        super(ID, userID);
        this.parentID = parentID;
    }
}

```

```java
package socialmedia;

/**
 * Class to create original objects
 */
public class Original extends PostWithChildren {

    /**
     * Method to set original variables
     * @param ID
     * @param userID
     * @param content
     */
    public Original(int ID, int userID, String content) {
        super(ID, userID, content);
    }
}

```

```java
package socialmedia;

import java.io.Serializable;

/**
 * Class to create post objects
 */
abstract public class Post implements Serializable {
    private final int ID;
    private final int userID;

    private String content;

    /**
     * Method to obtain ID
     * @return ID
     */
    public int getID() {
        return ID;
    }

    /**
     * Method to obtain user ID
     * @return userID
     */
    public int getUserID() {
        return userID;
    }

    /**
     * Method to obtain content
     * @return content
     */
    public String getContent() {
        return content;
    }

    /**
     * Method to set post variables
     * @param ID
```

```
     * @param userID
     */
    public Post(int ID, int userID){
        this.ID = ID;
        this.userID = userID;
    }

    /**
     * Method to set post variables with content
     * @param ID
     * @param userID
     * @param content
     */
    public Post(int ID, int userID, String content
){
        this.ID = ID;
        this.userID = userID;
        this.content = content;
    }
}


```

```java
package socialmedia;

import java.util.ArrayList;

/**
 * abstract class to create Post-PostWithChildren object
 */
public abstract class PostWithChildren extends Post {
    public ArrayList<Integer> children;

    /**
     * Method to delete child posts
     * @param id
     */
    public void deleteChild(Integer id) {
        children.remove(id);
    }

    /**
     * Method to output children
     * @return children
     */
    public ArrayList<Integer> getChildren() {
        return children;
    }

    /**
     * Method to add a child to post
     * @param childId
     */
    public void addChild(int childId){
        children.add(childId);
    }

    /**
     * Method to set children to post
     * @param children
     */
    public void setChildren(ArrayList<Integer>
```

```
children) {
        this.children = children;
    }

    /**
     * Method to set variables for PostWithChildren with content
     * @param ID
     * @param userID
     * @param content
     */
    public PostWithChildren(int ID, int userID, String content) {
        super(ID, userID, content);
        this.children = new ArrayList<>();
    }

    /**
     * Method to set variables for PostWithChildren
     * @param ID
     * @param userID
     */
    public PostWithChildren(int ID, int userID) {
        super(ID, userID);
        this.children = new ArrayList<>();
    }
}
```

```java
package socialmedia;

import java.io.*;
import java.util.*;

public class SocialMedia implements SocialMediaPlatform {
    private ArrayList<Account> accounts;
    private HashMap<Integer, Post> posts;
    private int accountIDGeneratorCounter;
    private int postIDGeneratorCounter;
    // constructor
    public SocialMedia() {
        accounts = new ArrayList<>();
        posts = new HashMap<>();
        accountIDGeneratorCounter = 1;
        postIDGeneratorCounter = 1;
    }

    /**
     * The method returns an account using an account handle
     * @param handle of the account wanted
     * @return account details
     * */
    private Account getAccountByHandle(String handle) {
        for(Account account : accounts) {
            if (account.getHandle().equals(handle)) {
                return account;
            }
        }
        return null;
    }

    /**
     * The method returns an account using an account id
     * @param id of account wanted
     * @return account details
```

```
     */
    private Account getAccountById(int id) {
        for(Account account : accounts) {
            if (account.getID() == id) {
                return account;
            }
        }
        return null;
    }
    @Override
    public int createAccount(String handle) throws IllegalHandleException, InvalidHandleException {
        if(handle.equals("") || handle.length() > 30 || handle.contains(" ")) {
            throw new InvalidHandleException();
        }
        if(getAccountByHandle(handle) != null) {
            throw new IllegalHandleException();
        }
        accounts.add(new Account(accountIDGeneratorCounter, handle));
        accountIDGeneratorCounter += 1;

        return accountIDGeneratorCounter - 1;
    }

    @Override
    public void removeAccount(int id) throws AccountIDNotRecognisedException {
        for(int i = 0; i < accounts.size(); i++){
            if (accounts.get(i).getID() == id){
                accounts.remove(i);
                return;
            }
        }
        throw new AccountIDNotRecognisedException();
    }

    @Override
    public void changeAccountHandle(String
```

```
oldHandle, String newHandle) throws HandleNotRecognisedException, IllegalHandleException, InvalidHandleException {
        if(newHandle.equals("") || newHandle.length() > 30 || newHandle.contains(" ")){
            throw new InvalidHandleException();
        }
        Account account = getAccountByHandle(newHandle);
        if(account != null) {
            throw new IllegalHandleException();
        }
        account = getAccountByHandle(oldHandle);
        if(account != null) {
            account.setHandle(newHandle);
            return;
        }
        throw new HandleNotRecognisedException();

    }

    @Override
    public String showAccount(String handle) throws HandleNotRecognisedException {
        Account account = getAccountByHandle(handle);
        if (account != null) {
            return String.format(
                    """
                    <pre>
                    ID: %d
                    Handle: %s
                    Description: %s
                    Post count: %d
                    Endorse count: %d
                    </pre>
                    """, account.getID(), account.getHandle(),
                    account.getDescription(), account.getPostCount(),
                    account.getEndorseCount());
```

```java

        }
        throw new HandleNotRecognisedException();
    }

    @Override
    public int createPost(String handle, String message) throws HandleNotRecognisedException, InvalidPostException {
        Account account = getAccountByHandle(handle);
        if(account == null){
            throw new HandleNotRecognisedException();
        }
        if(message.equals("") || message.length() > 100) {
            throw new InvalidPostException();
        }
        posts.put(postIDGeneratorCounter, new Original(postIDGeneratorCounter, account.getID(), message));
        account.incrementPostCount();
        postIDGeneratorCounter += 1;
        return postIDGeneratorCounter - 1;

    }

    @Override
    public int endorsePost(String handle, int id) throws HandleNotRecognisedException, PostIDNotRecognisedException, NotActionablePostException {
        Account account = getAccountByHandle(handle);
        if(account == null) {
            throw new HandleNotRecognisedException();
        }
        if(!posts.containsKey(id)) {
            throw new PostIDNotRecognisedException
```

```
();
        }
        if(posts.get(id) instanceof Endorsement || posts.get(id) instanceof DeletedPost) {
            throw new NotActionablePostException();
        }
        Endorsement endorsement = new Endorsement(postIDGeneratorCounter, account.getID(), id);
        ((PostWithChildren)posts.get(id)).addChild(postIDGeneratorCounter);
        getAccountById((posts.get(id)).getUserID()).incrementEndorseCount();
        account.incrementPostCount();
        posts.put(postIDGeneratorCounter, endorsement);
        postIDGeneratorCounter ++;
        return postIDGeneratorCounter-1;
    }

    @Override
    public int commentPost(String handle, int id, String message) throws HandleNotRecognisedException, PostIDNotRecognisedException, NotActionablePostException, InvalidPostException {
        Account account = getAccountByHandle(handle);
        if(account == null) {
            throw new HandleNotRecognisedException();
        }
        if(!posts.containsKey(id)) {
            throw new PostIDNotRecognisedException();
        }
        if(posts.get(id) instanceof Endorsement || posts.get(id) instanceof DeletedPost) {
            throw new NotActionablePostException();
        }
```

```
        if(message.equals("") || message.length() > 100) {
            throw new InvalidPostException();
        }
        Comment comment = new Comment(postIDGeneratorCounter, account.getID(), message, id);
        ((PostWithChildren)posts.get(id)).addChild(postIDGeneratorCounter);
        account.incrementPostCount();
        posts.put(postIDGeneratorCounter, comment);
        postIDGeneratorCounter ++;
        return postIDGeneratorCounter-1;
    }

    @Override
    public void deletePost(int id) throws PostIDNotRecognisedException {
        if(!posts.containsKey(id)) {
            throw new PostIDNotRecognisedException();
        }
        Post post = posts.get(id);
        if(post instanceof DeletedPost){
            throw new PostIDNotRecognisedException();
        }
        if (post instanceof Endorsement){
            Post parentPost = posts.get(((Endorsement)post).getParentID());
            ((PostWithChildren)parentPost).deleteChild(id);
            getAccountById(parentPost.getUserID()).decrementEndorseCount();
            getAccountById(post.getUserID()).decrementPostCount();
            posts.remove(id);
        }
        if (post instanceof PostWithChildren) {
            if (post instanceof Comment){
```

```
                ((PostWithChildren)posts.get(((Comment)post).getParentID())).deleteChild(id);
            }
            ArrayList<Integer> children = new ArrayList<>();
            Account postOwnerAccount = getAccountById(post.getUserID());
            for(Integer childId : ((PostWithChildren)post).getChildren()) {
                if(posts.get(childId) instanceof Endorsement) {
                    postOwnerAccount.decrementEndorseCount();
                    getAccountById(posts.get(childId).getUserID()).decrementPostCount();
                    posts.remove(id);
                }
                else {
                    children.add(childId);
                }
            }
            posts.put(id, new DeletedPost(id, children));
            getAccountById(post.getUserID()).decrementPostCount();
        }

    }

    /**
     * method to get string that shows all account info
     * @param id of the account requested
     * @return string formatted with all info on the account
     */
    private String getIndividualPostString(int id){
        Post post = posts.get(id);
        int numberOfComments = 0;
        int numberOfEndorsements = 0;
```

```
        Account account;
        if(post instanceof Endorsement){
            account = getAccountById(post.getUserID());
        } else{
            for(int postId : ((PostWithChildren) post).getChildren()){
                if(posts.get(postId) instanceof Comment){ numberOfComments ++; }
                if(posts.get(postId) instanceof Endorsement){ numberOfEndorsements ++; }
            }
            account = getAccountById(post.getUserID());
        }
        return String.format(
                """
                ID: %d
                Account: %s
                No. endorsements: %d | No. comments: %d
                %s
                """, post.getID(), account != null ? account.getHandle() : "null",
                numberOfEndorsements, numberOfComments,
                post.getContent());
    }

    @Override
    public String showIndividualPost(int id) throws PostIDNotRecognisedException {
        if (posts.containsKey(id)) {
            return "<pre>\n" + getIndividualPostString(id) + "\n</pre>";
        }
        throw new PostIDNotRecognisedException();    }

    /**
     * Takes a post and finds all child elements
```

```
of that post recursivly
     * @param post
     * @param depth of comments
     * @return StringBuilder (post and children)
     */
    private StringBuilder recursivePostTraversal(PostWithChildren post, int depth){
        if(post.getChildren().stream()
                .filter(element -> !(posts.get(element) instanceof Endorsement || posts.get(element) instanceof DeletedPost))
                .toList().size() == 0){
            return new StringBuilder();
        }

        StringBuilder stringBuilder = new StringBuilder("    ".repeat(depth) + "|");
        for(int childId:post.getChildren()){
            if(!(posts.get(childId) instanceof Endorsement || posts.get(childId) instanceof DeletedPost)){
                String postString = getIndividualPostString(childId);
                String[] lines = postString.split("\n");
                lines[0] = "| > " + lines[0];
                lines[1] = "    " + lines[1];
                lines[2] = "    " + lines[2];
                lines[3] = "    " + lines[3];

                postString = String.join("\n", lines);

                postString = "\n" + postString.replaceAll("(?m)^", "    ".repeat(depth));

                stringBuilder.append(postString);

                StringBuilder childrenString = recursivePostTraversal((PostWithChildren)posts.get(childId), depth + 1).insert(0, "\n");
```

```java

                stringBuilder.append(childrenString);

            }
        }
        return stringBuilder;
    }

    @Override
    public StringBuilder showPostChildrenDetails(int id) throws PostIDNotRecognisedException, NotActionablePostException {
        if(!posts.containsKey(id)) {
            throw new PostIDNotRecognisedException();
        }
        if(posts.get(id) instanceof Endorsement) {
            throw new NotActionablePostException();
        }
        StringBuilder stringBuilder = new StringBuilder("<pre>\n");

        stringBuilder.append(getIndividualPostString(id));
        stringBuilder.append(recursivePostTraversal((PostWithChildren)posts.get(id), 0));

        stringBuilder.append("<pre>");

        return stringBuilder;
    }

    @Override
    public int getMostEndorsedPost() {
        int highestNumberOfEndorsements = 0;
        Post mostEndorsedPost = null;
        for(Post post:posts.values()) {
            if(!(post instanceof Endorsement ||
```

```
post instanceof DeletedPost)){
                int numberOfEndorsments = 0;
                for(Integer id:((PostWithChildren) post).getChildren()){
                    if(posts.get(id) instanceof Endorsement){
                        numberOfEndorsments ++;
                    }
                }
                if(highestNumberOfEndorsements < numberOfEndorsments){
                    highestNumberOfEndorsements = numberOfEndorsments;
                    mostEndorsedPost = post;
                }
            }
        }
        if(mostEndorsedPost == null){
            return 0;
        }
        return mostEndorsedPost.getID();
    }

    @Override
    public int getMostEndorsedAccount() {
        Account account = accounts.stream().max(Comparator.comparingInt(Account::getEndorseCount)).orElse(null);
        return account != null ? account.getID() : 0;
    }

    @Override
    public void erasePlatform() {
        postIDGeneratorCounter = 1;
        accountIDGeneratorCounter = 1;
        posts = new HashMap<>();
        accounts = new ArrayList<>();
    }

    @Override
```

```
    public void savePlatform(String filename) throws IOException {
        FileOutputStream f = new FileOutputStream(filename + ".txt");
        ObjectOutputStream o = new ObjectOutputStream(f);

        // Write objects to file
        o.writeObject(this);

        o.close();
        f.close();
    }

    @Override
    public void loadPlatform(String filename) throws IOException, ClassNotFoundException {
        FileInputStream fi = new FileInputStream(filename + ".txt");
        ObjectInputStream oi = new ObjectInputStream(fi);

        SocialMedia socialMedia = (SocialMedia) oi.readObject();

        postIDGeneratorCounter = socialMedia.postIDGeneratorCounter;
        accountIDGeneratorCounter = socialMedia.accountIDGeneratorCounter;
        posts = socialMedia.posts;
        accounts = socialMedia.accounts;

        oi.close();
        fi.close();
    }

    @Override
    public int createAccount(String handle, String description) throws IllegalHandleException, InvalidHandleException {
        if(handle.equals("") || handle.length() >
```

```
30 || handle.contains(" ")){
            throw new InvalidHandleException();
        }
        if(getAccountByHandle(handle) != null) {
            throw new IllegalHandleException();
        }
        accounts.add(new Account(accountIDGeneratorCounter, handle, description));
        accountIDGeneratorCounter += 1;

        return accountIDGeneratorCounter - 1;
    }

    @Override
    public void removeAccount(String handle) throws HandleNotRecognisedException {
        for(int i = 0; i < accounts.size(); i++){
            if (accounts.get(i).getHandle().equals(handle)){
                // create a shallow copy of posts for iteration
                ArrayList<Post> postsImage = new ArrayList<>(posts.values());
                for(Post post:postsImage){
                    if(post.getUserID() == accounts.get(i).getID()){
                        try {
                            deletePost(post.getID());
                        } catch (PostIDNotRecognisedException e) {
                            throw new RuntimeException(e);
                        }
                    }
                }
                accounts.remove(i);
                return;
            }
        }
        throw new HandleNotRecognisedException();
```

```
    }

    @Override
    public void updateAccountDescription(String handle, String description) throws HandleNotRecognisedException {
        for (Account account : accounts) {
            if (account.getHandle().equals(handle)) {
                account.setDescription(description);
                return;
            }
        }
        throw new HandleNotRecognisedException();
    }

    @Override
    public int getNumberOfAccounts() {
        return accounts.size();
    }

    @Override
    public int getTotalOriginalPosts() {
        int totalOriginalPosts = 0;
        for(Post post: posts.values()){
            if(post instanceof Original) totalOriginalPosts ++;
        }
        return totalOriginalPosts;
    }

    @Override
    public int getTotalEndorsmentPosts() {
        int totalEndorsmentPosts = 0;
        for(Post post: posts.values()){
            if(post instanceof Endorsement) totalEndorsmentPosts ++;
        }
        return totalEndorsmentPosts;
    }
```

```java

    @Override
    public int getTotalCommentPosts() {
        int totalCommentPosts = 0;
        for(Post post: posts.values()){
            if(post instanceof Comment) totalCommentPosts ++;
        }
        return totalCommentPosts;
    }
}

```