```java
package socialmedia;

import javax.security.auth.login.AccountNotFoundException;
import java.io.*;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * SocialMedia is an implementor of the SocialMediaPlatform interface.
 */
public class SocialMedia implements SocialMediaPlatform {

    private static final Logger LOGGER = Logger.getLogger( SocialMedia.class.getName());
    private List<Account> accounts = new ArrayList<>();
    private ArrayList<Post> posts = new ArrayList<>();
    private StringBuilder childrenPostContent = new StringBuilder();
    private Account DELETED_USER = new Account();


    @Override
    public int createAccount(String handle) throws IllegalHandleException, InvalidHandleException {
        Account account = new Account(handle);
        accounts.add(account);
        return account.getId();
    }

    @Override
    public int createAccount(String handle, String description) throws IllegalHandleException, InvalidHandleException {
        try {
            Account account = new Account(handle, description);
            accounts.add(account);
            return account.getId();
        }catch (IllegalHandleException ihe){
            LOGGER.log( Level.SEVERE, ihe.toString(), ihe );
            return -1;

        }catch (InvalidHandleException ine){
            LOGGER.log( Level.SEVERE, ine.toString(), ine );
            return -1;
        }
    }

    @Override
    public void removeAccount(int id) throws AccountIDNotRecognisedException {
        Account a = getAccountById(id);
        a.removeAccount();
        accounts.remove(a);
    }

    @Override
    public void removeAccount(String handle) throws
```

```java
HandleNotRecognisedException {
        Account a = getAccountByHandle(handle);
        a.removeAccount();
        // TODO: remove corresponding posts and likes too
        accounts.remove(a);
    }

    @Override
    public void changeAccountHandle(String oldHandle, String newHandle)
            throws HandleNotRecognisedException, IllegalHandleException, InvalidHandleException {
        Account a = getAccountByHandle(oldHandle);
        a.changeAccountHandle(newHandle);
    }

    /**
     * Finds account with a given handle.
     * @param handle handle to identify the account.
     * @return account with a given handle.
     * @throws HandleNotRecognisedException if there is no matching account.
     */
    public Account getAccountByHandle(String handle) throws HandleNotRecognisedException {
        boolean found = false;
        int foundIndex = -1;
        for (int i = 0; i < accounts.size() && !found; i++) {
            if (accounts.get(i).getHandle() == handle){
                found = true;
                foundIndex = i;
            }
        }
        if (found){
            return accounts.get(foundIndex);
        }
        else {
            throw new HandleNotRecognisedException("Account with handle: " + handle + " not found in the system.");
        }
    }

    /**
     * Finds account with a given id.
     * @param id id to identify the account.
     * @return account with a given id.
     * @throws AccountIDNotRecognisedException
     */
    public Account getAccountById(int id) throws AccountIDNotRecognisedException {
        boolean found = false;
        int foundIndex = -1;
        for (int i = 0; i < accounts.size() && !found; i++) {
            if (accounts.get(i).getId() == id){
                found = true;
                foundIndex = i;
            }
        }
        if (found){
            return accounts.get(foundIndex);
        }
        else {
            throw new AccountIDNotRecognisedException("Account with id: " + id + " not found in the system.");
        }
    }

    @Override
    public void updateAccountDescription(String handle, String description) throws HandleNotRecognisedException {
        Account a = getAccountByHandle(handle);
        a.setDescriptionField(description);
    }

    @Override
    public String showAccount(String handle) throws HandleNotRecognisedException {
        Account a = getAccountByHandle(handle);
        return a.toString();
    }

    @Override
    public int createPost(String handle, String message) throws HandleNotRecognisedException, InvalidPostException {
        Account a = getAccountByHandle(handle);
        Post p = new Post(message, a);
        posts.add(p);
        return p.getId();
    }

    @Override
    public int endorsePost(String handle, int id)
            throws HandleNotRecognisedException, PostIDNotRecognisedException, NotActionablePostException {
        Post p = getPostById(id);
        Account a = getAccountByHandle(handle);
        if (p instanceof Endorsement){
            throw new NotActionablePostException("Post with id: '" + id + "' is an endorsement " +
                    "so it cannot be endorsed.");
        }
        Endorsement endorsement = new Endorsement(p,a);
        posts.add(endorsement);
        return endorsement.getId();
    }


    public Post getPostById(int id) throws PostIDNotRecognisedException{
        for (int i = 0; i < posts.size(); i++) {
            if (posts.get(i).getId() == id){
                return posts.get(i);
            }
        }
        throw new PostIDNotRecognisedException("Post with id: '" + id + "' not found.");
    }

    @Override
    public int commentPost(String handle, int id, String message) throws HandleNotRecognisedException,
            PostIDNotRecognisedException, NotActionablePostException, InvalidPostException {
        Account a = getAccountByHandle(handle);
        Post p = getPostById(id);
        if (p instanceof Endorsement){
            throw new NotActionablePostException("Post with id: '" + id + "' is an endorsement " +
                    "so it cannot be commented.");
        }
        Comment comment = new Comment(a,p,message);
        posts.add(comment);
        return comment.getId();
    }

    @Override
    public void deletePost(int id) throws PostIDNotRecognisedException {
        Post targetPost = new Post();
        for(Post post: posts){
            if(post.getId() == id){
                targetPost = post;
                break;
            }
        }
        targetPost.setPostContent("<The initial content was deleted from the system, therefore it is not available anymore.>");
        targetPost.setDeleted(true);
        for(Post Endorsement: targetPost.getEndorsements()){
            for(Post post: posts){
                if(post.getId() == Endorsement.getId()){
                    posts.remove(post);
                    break;
                }
            }
            Endorsement.clearAll();
        }
    }

    @Override
    public String showIndividualPost(int id) throws PostIDNotRecognisedException {
        boolean postIDRecognised = false;
        Post targetPost = new Post();
        String individualPost;
        try {
            for (Post post : posts) {
                if (post.getId() == id) {
                    postIDRecognised = true;
                    targetPost = post;
                    break;
                }
            }
            if (!postIDRecognised) {
                throw new PostIDNotRecognisedException("Post Id not recognised exception: Please try again entering a valid Id.");
            }
            individualPost = "Id: " + targetPost.getId() + "\nAccount: " + targetPost.getAccount().getHandle()
                    + "\nNo. endorsements: " + targetPost.getEndorsementCount() + " | No. comments: " + targetPost.getCommentCount() + "\n" + targetPost.getPostContent() + " \n";
            return individualPost;
        } catch(PostIDNotRecognisedException e){
            String message = "Post Id not recognised exception: Please try again entering a valid Id.";
            return message;
        }
    }
    public void clearStringBuilder(){
        childrenPostContent.setLength(0);
    }
    public void FormatStringBuilder(Post post) throws PostIDNotRecognisedException {
        try{
            if(post != null){
                String individualPost;
                if(post.isComment()){
                    childrenPostContent.append(("    ").repeat(Math.max(0,post.getDepth()) - 1)).append("| >");
                    individualPost = showIndividualPost(post.getId()).replace("\n","\n" + ("    ").repeat(Math.max(0,post.getDepth())));
                }else{
                    individualPost = showIndividualPost(post.getId());
                }
                childrenPostContent.append(individualPost).append("|\n");
                for(Post child: post.getPostChildrenList()){
                    FormatStringBuilder(child);
                }
            }}
        catch(PostIDNotRecognisedException e){
            System.out.println("PROBLEM WITH showingIndividualPosts()");
        }
    }


    @Override
    public StringBuilder showPostChildrenDetails(int id)
            throws PostIDNotRecognisedException, NotActionablePostException {
        // TODO Auto-generated method stub
        return null;
    }

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

    @Override
    public int getTotalOriginalPosts() {
        int count = 0;
        for(Post post: posts){
            if(post.isComment() || post.isEndorsement()){
                continue;
            }
            count++;
        }
        return count;
    }

    @Override
    public int getTotalEndorsmentPosts() {
        int count = 0;
        for(Post post: posts){
            if(!post.isEndorsement()){
                continue;
            }
            count++;
        }
        return count;
    }

    @Override
    public int getTotalCommentPosts() {
        int count = 0;
        for(Post post: posts){
            if(!post.isComment()){
                continue;
            }
            count++;
        }
        return count;
    }

    @Override
    public int getMostEndorsedPost() {
        Post MostEndorsed = new Post();
        for(Post post : posts){
            if(post.isEndorsement()){
                continue;
            }
            if(MostEndorsed.getAccount() == null){
                MostEndorsed = post;
            }
            else if(MostEndorsed.getEndorsementCount() < post.
  getEndorsementCount()){
                MostEndorsed = post;
            }

        }
        return MostEndorsed.getId();

    }

    @Override
    public int getMostEndorsedAccount() {
        Account MostEndorsed = new Account();
        for(Account user : accounts){
            if(MostEndorsed.getHandle() == null){
                MostEndorsed = user;
            }
            else if(MostEndorsed.getUserEndorsements() < user.getUserEndorsements()){
                MostEndorsed = user;
            }
        }
        return MostEndorsed.getId();

    }

    @Override
    public void erasePlatform() {
        for(Account account: accounts){
            account.getPosts().clear();
        }
        accounts.clear();
        for(Post posts: posts){
            posts.getPostChildrenList().clear();
            posts.getEndorsements().clear();
            posts.clearAccount();
        }
        posts.clear();
        DELETED_USER.getPosts().clear();

    }

    @Override
    public void savePlatform(String filename) throws IOException {
        try{
            FileOutputStream fileStore = new FileOutputStream(filename + ".ser");
            ObjectOutputStream objectStore = new ObjectOutputStream(fileStore);
            objectStore.writeObject(accounts);
            objectStore.writeObject(posts);
            objectStore.writeObject(DELETED_USER);
            objectStore.close();
            fileStore.close(); }
        catch(IOException e){
            System.out.println("Unfortunately, not able to find the file. Please try again.");
        }
    }

    @Override
    public void loadPlatform(String filename) throws IOException, ClassNotFoundException {
        try{
            ObjectInputStream ois = new ObjectInputStream(new FileInputStream(filename + ".ser"));
            accounts = (ArrayList<Account>) ois.readObject();
            posts = (ArrayList<Post>) ois.readObject();
            DELETED_USER = (Account) ois.readObject();
            ois.close();}
        catch(IOException | ClassNotFoundException e) {
            System.out.println("Unfortunately, the class or the file were not found. Please try again.");
        }

    }

    public List<Account> getAllAccounts(){
        return accounts;
    }

    public List<Post> getPosts() {return posts; }
}

```

```java
package socialmedia;

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

/**
 * A class representing user's post.
 */
public class Post {
    private Account account;
    private int numberOfEndorsements = 0;
    private int numberOfComments = 0;
    private String postContent;
    private int commentCount = 0;
    private int endorsementCount = 0;
    private boolean isDeleted = false;
    private boolean isComment = false;
    private ArrayList<Post> postChildrenList = new ArrayList<>();
    private boolean isEndorsement = false;
    private int depth = 0;
    protected int id;
    // will increment with each new post. Will make sure that they all have an unique numerical id
    private static int counter = 0;
    protected String message;
    Account author;
    List<Endorsement> endorsements = new ArrayList<>();
    List<Comment> comments = new ArrayList<>();
    Post(){
    }
    Post(Account account, int id){
        this.account = account;
        this.id = id;
    }
    Post(Account account, String postContent, int postID) {
        this.account = account;
        this.postContent = postContent;
        this.id = id;
    }

    public Post(String message, Account author) throws InvalidPostException {
        setId();
        setMessage(message);
        this.author = author;
    }

    public void clearAccount(){
        this.account = null;
    }

    /**
     * Constructor used by child classes (does not throw exception)
     * It also does not set the message.
     *
     * @param author author Account
     */
    public Post(Account author) {
        setId();
        this.author = author;
    }

    protected void setId() {
        this.id = counter;
        counter += 1;
    }

    /**
     * Sets post's message
     *
     * @param message message to be set
     * @throws InvalidPostException if message is null, just white characters, empty or longer than 100 characters
     */
    public void setMessage(String message) throws InvalidPostException {
        if (message != null && message.trim() != ""
                && message.length() <= 100) {
            this.message = message;
        } else
            throw new InvalidPostException("Message '" + message + "' is either null, just whitespaces " +
                    "or longer than 100 chars. Post cannot be created.");
    }

    public int getId() {
        return id;
    }

    public String getMessage() {
        return message;
    }

    public Account getAuthor() {
        return author;
    }

    public List<Endorsement> getEndorsements() {
        return endorsements;
    }

    public List<Comment> getComments() {
        return comments;
    }

    /**
     * increments number of endorsements for the post and adds the endorsement.
     */
    public void endorse(Endorsement endorsement) {
        endorsements.add(endorsement);
        this.numberOfEndorsements += 1;
    }

    /**
     * increments comments count and adds the comment.
     */
    public void comment(Comment comment) {
        comments.add(comment);
        this.numberOfComments += 1;
    }

    public void clearAll() {
        account = null;
        postContent = "<The original content was deleted from the system and is not available anymore.>";
        commentCount = -1;
        endorsementCount = -1;
    }

    @Override
    public String toString() {
        return "Post{" +
                "id=" + id +
                ", message='" + message + '\'' +
                ", author=" + author +
                ", endorsements=" + endorsements +
                ", comments=" + comments +
                '}';
    }


    public void setPostContent(String postContent) {
        this.postContent = postContent;
    }
    public boolean isDeleted() {
        return isDeleted;
    }
    public void setDeleted(boolean deleted) {
        isDeleted = deleted;
    }
    public Account getAccount() {
        return account;
    }
    public int getEndorsementCount() {
        return endorsementCount;
    }
    public int getCommentCount() {
        return commentCount;
    }
    public String getPostContent() {
        return postContent;
    }
    public void setComment(boolean comment) {
        isComment = comment;
    }
    public boolean isComment() {
        return isComment;
    }
    public void addDepth(){
        depth++;
    }
    public int getDepth() {
        return depth;
    }
    public void setDepth(int depth){
        this.depth = depth;
    }
    public ArrayList<Post> getPostChildrenList() {
        return postChildrenList;
    }
    public void setEndorsement(boolean endorsement) {
        isEndorsement = endorsement;
    }

    public boolean isEndorsement() {
        return isEndorsement;
    }
}
```

```java
package socialmedia;

public class Comment extends Post{
    private Post commentedPost;

    public Comment(Account author, Post commentedPost, String commentText) throws InvalidPostException {
        super(author);
        super.setMessage(message);
        this.commentedPost = commentedPost;
        commentedPost.comment(this);
        commentedPost.author.comment();
    }

    @Override
    public String toString() {
        return "Comment{" +
                "commentedPost=" + commentedPost +
                ", id=" + id +
                ", message='" + message + '\'' +
                ", author=" + author +
                '}';
    }
}

```

```java
package socialmedia;

/**
 * A class representing other post's endorsement.
 */
public class Endorsement extends Post{
    private Post endorsedPost;

    public Endorsement(Post endorsedPost, Account account){
        super(account);
        this.message = endorsedPost.message;
        this.endorsedPost = endorsedPost;
        endorsedPost.endorse(this);
        endorsedPost.author.endorse();
    }

    @Override
    public String toString() {
        return "Endorsement{" +
                "id=" + id +
                ", message='" + message + '\'' +
                ", author=" + author +
                '}';
    }
}

```

```java
package socialmedia;

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

/**
 * A class representing user's account
 */
public class Account {
    private int id;
    private int numberOfEndorsements = 0;
    private int numberOfComments = 0;
    // will increment with each new account. Will make sure that they all have an unique numerical id
    private static int counter = 0;
    private String descriptionField;
    private String handle;
    private ArrayList<Post> posts = new ArrayList<>();
    private static List<String> accountHandles = new ArrayList<>();
    Account(){

    }

    /**
     * Creates an instance of Account class
     * @param descriptionField String containing description field
     * @param handle handle to identify the account
     * @throws IllegalHandleException if the handle already exists in the platform.
     * @throws InvalidHandleException if the new handle is empty, has more than 30 characters, or has white spaces.
     */
    public Account( String handle, String descriptionField) throws IllegalHandleException, InvalidHandleException {
        setId();
        setHandle(handle);
        setDescriptionField(descriptionField);
    }

    /**
     * Creates an instance of Account class with empty description field
     * @param handle handle to identify the account
     * @throws IllegalHandleException if the handle already exists in the platform.
     * @throws InvalidHandleException if the new handle is empty, has more than 30 characters, or has white spaces.
     */
    public Account(String handle) throws IllegalHandleException, InvalidHandleException {
        setId();
        setHandle(handle);
        setDescriptionField("");
    }


    public int getId() {
        return id;
    }

    public void setId() {
        this.id = counter;
        counter += 1;
    }

    public String getDescriptionField() {
        return descriptionField;
    }

    public void setDescriptionField(String descriptionField) {
        this.descriptionField = descriptionField;
    }

    public String getHandle() {
        return handle;
    }

    /**
     * Sets a handle for account instance.
     * @param handle handle to identify the account.
     * @throws IllegalHandleException if the handle already exists in the platform.
     * @throws InvalidHandleException if the new handle is empty, has more than 30 characters, or has white spaces.
     */
    private void setHandle(String handle) throws IllegalHandleException, InvalidHandleException{
        if (handle != null && handle != ""
                && handle.length() <= 30 &&
                !containsWhitespace(handle)) {
            if (!accountHandles.contains(handle)){
                this.handle = handle;
                accountHandles.add(handle);
            }else
                throw new IllegalHandleException("Handle '" + handle + "' already exists in the system.");
        }else
            throw new InvalidHandleException("Handle '"+ handle +"' is empty, has more than 30 characters," +
                    " or has white spaces");
    }

    public void changeAccountHandle(String newHandle)throws IllegalHandleException, InvalidHandleException{
        accountHandles.remove(this.handle);
        setHandle(newHandle);
    }

    /**
     * Checks a string for whitespaces
     * @param str string checked for whitespaces
     * @return true if given string contains any whitespace, else false
     */
    private boolean containsWhitespace(String str){
        // true if contains onw or more instances of any whitespace character
        return str.matches(".*\\s.*");
    }

    public void removeAccount(){
        accountHandles.remove(this.handle);
    }

    /**
     * increments number of endorsements for the account.
     */
    public void endorse(){
        this.numberOfEndorsements += 1;
    }
    public void comment(){
        this.numberOfComments +=1;
    }

    @Override
    public String toString() {
        return "Account{" +
                "id=" + id +
                ", descriptionField='" + descriptionField + '\'' +
                ", handle='" + handle + '\'' +
                '}';
    }
    public int getUserEndorsements() {
        return numberOfEndorsements;
    }

    public ArrayList<Post> getPosts() {
        return posts;
    }
}

```