```java
package socialmedia;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.ArrayList;

import java.lang.StringBuilder;

/**
 * BadSocialMedia is a minimally compiling, but non-functioning implementor of
 * the SocialMediaPlatform interface.
 *
 * @author Diogo Pacheco
 * @version 1.0
 */
public class SocialMedia implements SocialMediaPlatform {

    ArrayList<Account> accounts;
    ArrayList<Post> posts;

    /**
     * constructor
     */
    public SocialMedia(){
        accounts = new ArrayList<Account>();
        posts = new ArrayList<Post>();
    }

    @Override
    public int createAccount(String handle) throws IllegalHandleException,
    InvalidHandleException {
        // TODO Auto-generated method stub
        if (handleExists(handle)){
            throw new IllegalHandleException();
        }
        if (!isValidHandle(handle)){
            throw new InvalidHandleException();
        }
        Account newAccount = new Account(handle);
        accounts.add(newAccount);
        return newAccount.getID();
    }

    @Override
    public int createAccount(String handle, String description) throws
    IllegalHandleException, InvalidHandleException {
        // TODO Auto-generated method stub
        Account newAccount = new Account(handle, description);
        accounts.add(newAccount);
        return newAccount.getID();
    }

    @Override
    public void removeAccount(int id) throws AccountIDNotRecognisedException {
        // TODO Auto-generated method stub
        for (Account a : accounts){
            if (a.getID() == id){
                accounts.remove(a);
                return;
            }
        }
        throw new AccountIDNotRecognisedException();
    }

    @Override
    public void removeAccount(String handle) throws HandleNotRecognisedException {
        // TODO Auto-generated method stub
        accounts.remove(getAccountByHandle(handle));
        throw new HandleNotRecognisedException();
    }

    @Override
    public void changeAccountHandle(String oldHandle, String newHandle)
            throws HandleNotRecognisedException, IllegalHandleException,
            InvalidHandleException {
        // TODO Auto-generated method stub
        if (handleExists(newHandle)){
            throw new IllegalHandleException();
        }
        if (!isValidHandle(newHandle)){
            throw new InvalidHandleException();
        }
        Account accountToChange = getAccountByHandle(oldHandle);
        if (accountToChange == null){
            throw new HandleNotRecognisedException();
        }

        accountToChange.updateHandle(newHandle);

    }

    @Override
    public void updateAccountDescription(String handle, String description) throws
    HandleNotRecognisedException {
        // TODO Auto-generated method stub
        Account accountToChange = getAccountByHandle(handle);
        if (accountToChange == null){
            throw new HandleNotRecognisedException();
        }

        accountToChange.updateDescription(handle);

    }

    @Override
    public String showAccount(String handle) throws HandleNotRecognisedException {
        // TODO Auto-generated method stub
        Account account = getAccountByHandle(handle);
        if (account == null){
            throw new HandleNotRecognisedException();
        }
        return account.showAccount();
    }

    @Override
    public int createPost(String handle, String message) throws
    HandleNotRecognisedException, InvalidPostException {
        // TODO Auto-generated method stub
        Account poster = getAccountByHandle(handle);
        if (poster == null){
            throw new HandleNotRecognisedException();
        }
        if(!isValidPost(message)){
            throw new InvalidPostException();
        }
        Post newPost = new Original(poster, message);
        posts.add(newPost);
        poster.addPost(newPost);
        return newPost.getID();
    }

    @Override
    public int endorsePost(String handle, int id)
            throws HandleNotRecognisedException, PostIDNotRecognisedException,
            NotActionablePostException {
```

```java
        // TODO Auto-generated method stub
        // EP@exampleuser:Endorsed message
        Account endorser = getAccountByHandle(handle);
        if (endorser == null){
            throw new HandleNotRecognisedException();
        }
        Post postToEndorse = getPostByID(id);
        if(postToEndorse == null){
            throw new PostIDNotRecognisedException();
        }
        if (postToEndorse instanceof Endorsement || postToEndorse instanceof EmptyPost){
            throw new NotActionablePostException();
        }
        Endorsement newEndorsement = new Endorsement(endorser, postToEndorse);
        postToEndorse.addEndorsements(newEndorsement);
        posts.add(newEndorsement);
        endorser.addPost(newEndorsement);
        getAccountByHandle(postToEndorse.getAccountHandle()).increaseEndorsements();
        return newEndorsement.getID();
    }

    @Override
    public int commentPost(String handle, int id, String message) throws HandleNotRecognisedException,
            PostIDNotRecognisedException, NotActionablePostException, InvalidPostException {
        // TODO Auto-generated method stub
        Account commenter = getAccountByHandle(handle);
        if(commenter == null){
            throw new HandleNotRecognisedException();
        }
        Post postToComment = getPostByID(id);
        if(postToComment == null){
            throw new PostIDNotRecognisedException();
        }
        if (postToComment instanceof Endorsement || postToComment instanceof EmptyPost){
            throw new NotActionablePostException();
        }
        if(!isValidPost(message)){
            throw new InvalidPostException();
        }
        Comment newComment = new Comment(commenter, message, postToComment);
        posts.add(newComment);
        postToComment.addComment(newComment);
        commenter.addPost(newComment);
        return newComment.getID();
    }

    @Override
    public void deletePost(int id) throws PostIDNotRecognisedException {
        // TODO Auto-generated method stub
        Post postToRemove = getPostByID(id);
        if (postToRemove == null){
            throw new PostIDNotRecognisedException();
        }
        posts.removeAll(postToRemove.getEndorsements());
        postToRemove.deleteEndorsements();
        int index = posts.indexOf(postToRemove);
        EmptyPost newEmptyPost = new EmptyPost(postToRemove);
        posts.set(index, newEmptyPost);
        getAccountByHandle(postToRemove.getAccountHandle()).removePost(postToRemove);
        ArrayList<Comment> oldComments = postToRemove.getComments();
        for (Comment c : oldComments){
            c.setOriginalPost(newEmptyPost);
        }
    }

    @Override
    public String showIndividualPost(int id) throws PostIDNotRecognisedException {
        // TODO Auto-generated method stub
        Post postToShow = getPostByID(id);
        if (postToShow == null){
            throw new PostIDNotRecognisedException();
        }
        return postToShow.show(0);
    }

    @Override
    public StringBuilder showPostChildrenDetails(int id)
            throws PostIDNotRecognisedException, NotActionablePostException {
        // TODO Auto-generated method stub
        Post postToShow = getPostByID(id);
        if (postToShow == null){
            throw new PostIDNotRecognisedException();
        }
        if (postToShow instanceof Endorsement){
            throw new NotActionablePostException();
        }
        StringBuilder sb = new StringBuilder(postToShow.showWithChildren(0));
        return sb;
    }

    @Override
    public int getNumberOfAccounts() {
        // TODO Auto-generated method stub
        return accounts.size();
    }

    @Override
    public int getTotalOriginalPosts() {
        // TODO Auto-generated method stub
        int numOriginalPosts = 0;
        for (Post p : posts){
            if (p instanceof Original){
                numOriginalPosts++;
            }
        }
        return numOriginalPosts;
    }

    @Override
    public int getTotalEndorsmentPosts() {
        // TODO Auto-generated method stub
        int numEndorsementPosts = 0;
        for (Post p : posts){
            if (p instanceof Endorsement){
                numEndorsementPosts++;
            }
        }
        return numEndorsementPosts;
    }

    @Override
    public int getTotalCommentPosts() {
        // TODO Auto-generated method stub
        int numCommentsPosts = 0;
        for (Post p : posts){
            if (p instanceof Comment){
                numCommentsPosts++;
            }
        }
        return numCommentsPosts;
    }

    @Override
    public int getMostEndorsedPost() {
        // TODO Auto-generated method stub
        Post mostEndorsed = null;
        boolean firstFound = false;
        for (Post p : posts){
            if (p instanceof Original || p instanceof Comment){
                if (firstFound){
                    if (p.getNumEndorsements() > mostEndorsed.getNumEndorsements()){
                        mostEndorsed = p;
                    }
                }
                else{
                    mostEndorsed = p;
                }
            }
        }
        return mostEndorsed.getID();
    }

    @Override
    public int getMostEndorsedAccount() {
        // TODO Auto-generated method stub
        Account mostEndorsed = null;
        boolean firstFound = false;
        for (Account a : accounts){
            if (firstFound){
                if (a.getEndorseCount() > mostEndorsed.getEndorseCount()){
                    mostEndorsed = a;
                }
            }
            else{
                mostEndorsed = a;
            }
        }
        return mostEndorsed.getID();
    }

    @Override
    public void erasePlatform() {
        // TODO Auto-generated method stub
        posts.clear();
        accounts.clear();

    }

    @Override
    public void savePlatform(String filename) throws IOException {
        // TODO Auto-generated method stub
        try{
            Object arr[] = new Object[2];
            arr[0] = accounts;
            arr[1] = posts;
            FileOutputStream fileOut = new FileOutputStream("platforms/" +  filename +
            ".obj");
            ObjectOutputStream objOut = new ObjectOutputStream(fileOut);
            objOut.writeObject(arr);
            objOut.close();
        }
        catch(IOException e){
            throw e;
        }
        catch (Exception e){
            System.out.println("something went wrong, man");
        }

    }

    @Override
    public void loadPlatform(String filename) throws IOException,
    ClassNotFoundException {
        // TODO Auto-generated method stub
        try{
            FileInputStream inStream = new FileInputStream("platforms/" + filename +
            ".obj");
            ObjectInputStream objIn = new ObjectInputStream(inStream);
            Object arr[] = (Object[])objIn.readObject();
            accounts = (ArrayList<Account>)arr[0];
            posts = (ArrayList<Post>)arr[1];
            objIn.close();
        }
        catch(IOException e){
            throw e;
        }
        catch(ClassNotFoundException e){
            throw e;
        }
        catch(Exception e){
            System.out.println("something went wrong, man");
        }

    }

    private boolean handleExists(String handle){
        for (Account a : accounts){
            if (a.getHandle().equals(handle)){
                return true;
            }
        }
        return false;
    }

    private boolean isValidHandle(String handle){
        //no whitespace, no empty, no more than 30 characters
        if (handle.isEmpty()){
            return false;
        }
        if (handle.contains(" ")){
            return false;
        }
        if (handle.length() > 30){
            return false;
        }
        return true;
    }

    private boolean isValidPost(String content){
        if(content.isEmpty()){
            return false;
        }
        if (content.length() > 100){
            return false;
        }
        return true;
    }

    private Account getAccountByHandle(String handle){
        for (Account a : accounts){
            if (a.getHandle().equals(handle)){
                return a;
            }
        }
        return null;
    }

    private Post getPostByID(int id){
        for (Post p : posts){
            if (p.getID() == id){
                return p;
            }
        }
        return null;
        }
    }

```