```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.io.File;

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
```

```
68      public void removeAccount(String handle) throws HandleNotRecognisedException {
69          // TODO Auto-generated method stub
70          accounts.remove(getAccountByHandle(handle));
71          throw new HandleNotRecognisedException();
72      }
73
74      @Override
75      public void changeAccountHandle(String oldHandle, String newHandle)
76              throws HandleNotRecognisedException, IllegalHandleException,
                InvalidHandleException {
77          // TODO Auto-generated method stub
78          if (handleExists(newHandle)){
79              throw new IllegalHandleException();
80          }
81          if (!isValidHandle(newHandle)){
82              throw new InvalidHandleException();
83          }
84          Account accountToChange = getAccountByHandle(oldHandle);
85          if (accountToChange == null){
86              throw new HandleNotRecognisedException();
87          }
88
89          accountToChange.updateHandle(newHandle);
90
91      }
92
93      @Override
94      public void updateAccountDescription(String handle, String description) throws
        HandleNotRecognisedException {
95          // TODO Auto-generated method stub
96          Account accountToChange = getAccountByHandle(handle);
97          if (accountToChange == null){
98              throw new HandleNotRecognisedException();
99          }
100
101         accountToChange.updateDescription(handle);
102
103     }
104
105     @Override
106     public String showAccount(String handle) throws HandleNotRecognisedException {
107         // TODO Auto-generated method stub
108         Account account = getAccountByHandle(handle);
109         if (account == null){
110             throw new HandleNotRecognisedException();
111         }
112         return account.showAccount();
113     }
114
115     @Override
116     public int createPost(String handle, String message) throws
        HandleNotRecognisedException, InvalidPostException {
117         // TODO Auto-generated method stub
118         Account poster = getAccountByHandle(handle);
119         if (poster == null){
120             throw new HandleNotRecognisedException();
121         }
122         if(!isValidPost(message)){
123             throw new InvalidPostException();
124         }
125         Post newPost = new Original(poster, message);
126         posts.add(newPost);
127         poster.addPost(newPost);
128         return newPost.getID();
129     }
130
131     @Override
132     public int endorsePost(String handle, int id)
133             throws HandleNotRecognisedException, PostIDNotRecognisedException,
```

```
            NotActionablePostException {
        // 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
```

```
200         public String showIndividualPost(int id) throws PostIDNotRecognisedException {
201             // TODO Auto-generated method stub
202             Post postToShow = getPostByID(id);
203             if (postToShow == null){
204                 throw new PostIDNotRecognisedException();
205             }
206             return postToShow.show(0);
207         }
208 
209         @Override
210         public StringBuilder showPostChildrenDetails(int id)
211                 throws PostIDNotRecognisedException, NotActionablePostException {
212             // TODO Auto-generated method stub
213             Post postToShow = getPostByID(id);
214             if (postToShow == null){
215                 throw new PostIDNotRecognisedException();
216             }
217             if (postToShow instanceof Endorsement){
218                 throw new NotActionablePostException();
219             }
220             StringBuilder sb = new StringBuilder(postToShow.showWithChildren(0));
221             return sb;
222         }
223 
224         @Override
225         public int getNumberOfAccounts() {
226             // TODO Auto-generated method stub
227             return accounts.size();
228         }
229 
230         @Override
231         public int getTotalOriginalPosts() {
232             // TODO Auto-generated method stub
233             int numOriginalPosts = 0;
234             for (Post p : posts){
235                 if (p instanceof Original){
236                     numOriginalPosts++;
237                 }
238             }
239             return numOriginalPosts;
240         }
241 
242         @Override
243         public int getTotalEndorsmentPosts() {
244             // TODO Auto-generated method stub
245             int numEndorsementPosts = 0;
246             for (Post p : posts){
247                 if (p instanceof Endorsement){
248                     numEndorsementPosts++;
249                 }
250             }
251             return numEndorsementPosts;
252         }
253 
254         @Override
255         public int getTotalCommentPosts() {
256             // TODO Auto-generated method stub
257             int numCommentsPosts = 0;
258             for (Post p : posts){
259                 if (p instanceof Comment){
260                     numCommentsPosts++;
261                 }
262             }
263             return numCommentsPosts;
264         }
265 
266         @Override
267         public int getMostEndorsedPost() {
268             // TODO Auto-generated method stub
```

```
269             Post mostEndorsed = null;
270             boolean firstFound = false;
271             for (Post p : posts){
272                 if (p instanceof Original || p instanceof Comment){
273                     if (firstFound){
274                         if (p.getNumEndorsements() > mostEndorsed.getNumEndorsements()){
275                             mostEndorsed = p;
276                         }
277                     }
278                     else{
279                         mostEndorsed = p;
280                     }
281                 }
282             }
283             return mostEndorsed.getID();
284         }
285 
286         @Override
287         public int getMostEndorsedAccount() {
288             // TODO Auto-generated method stub
289             Account mostEndorsed = null;
290             boolean firstFound = false;
291             for (Account a : accounts){
292                 if (firstFound){
293                     if (a.getEndorseCount() > mostEndorsed.getEndorseCount()){
294                         mostEndorsed = a;
295                     }
296                 }
297                 else{
298                     mostEndorsed = a;
299                 }
300             }
301             return mostEndorsed.getID();
302         }
303 
304         @Override
305         public void erasePlatform() {
306             // TODO Auto-generated method stub
307             posts.clear();
308             accounts.clear();
309 
310         }
311 
312         @Override
313         public void savePlatform(String filename) throws IOException {
314             // TODO Auto-generated method stub
315             try{
316                 Object arr[] = new Object[2];
317                 arr[0] = accounts;
318                 arr[1] = posts;
319                 FileOutputStream fileOut = new FileOutputStream( filename + ".obj");
320                 ObjectOutputStream objOut = new ObjectOutputStream(fileOut);
321                 objOut.writeObject(arr);
322                 objOut.close();
323             }
324             catch(IOException e){
325                 throw e;
326             }
327             catch (Exception e){
328                 System.out.println("something went wrong, man");
329             }
330 
331         }
332 
333         @Override
334         public void loadPlatform(String filename) throws IOException,
        ClassNotFoundException {
335             // TODO Auto-generated method stub
336             try{
```

```
            FileInputStream inStream = new FileInputStream(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;
    }
```

```
406     }
```