

```

1  package socialmedia;
2
3  import java.io.FileInputStream;
4  import java.io.FileOutputStream;
5  import java.io.IOException;
6  import java.io.ObjectInputStream;
7  import java.io.ObjectOutputStream;
8  import java.util.ArrayList;
9
10 import java.lang.StringBuilder;
11
12 /**
13  * BadSocialMedia is a minimally compiling, but non-functioning implementor of
14  * the SocialMediaPlatform interface.
15  *
16  * @author Diogo Pacheco
17  * @version 1.0
18  */
19 public class SocialMedia implements SocialMediaPlatform {
20
21     ArrayList<Account> accounts;
22     ArrayList<Post> posts;
23
24     /**
25      * constructor
26      */
27     public SocialMedia() {
28         accounts = new ArrayList<Account>();
29         posts = new ArrayList<Post>();
30     }
31
32     @Override
33     public int createAccount(String handle) throws IllegalHandleException,
34     InvalidHandleException {
35         // TODO Auto-generated method stub
36         if (handleExists(handle)) {
37             throw new IllegalHandleException();
38         }
39         if (!isValidHandle(handle)) {
40             throw new InvalidHandleException();
41         }
42         Account newAccount = new Account(handle);
43         accounts.add(newAccount);
44         return newAccount.getID();
45     }
46
47     @Override
48     public int createAccount(String handle, String description) throws
49     IllegalHandleException, InvalidHandleException {
50         // TODO Auto-generated method stub
51         Account newAccount = new Account(handle, description);
52         accounts.add(newAccount);
53         return newAccount.getID();
54     }
55
56     @Override
57     public void removeAccount(int id) throws AccountIDNotRecognisedException {
58         // TODO Auto-generated method stub
59         for (Account a : accounts) {
60             if (a.getID() == id) {
61                 accounts.remove(a);
62                 return;
63             }
64         }
65         throw new AccountIDNotRecognisedException();
66     }
67
68     @Override
69     public void removeAccount(String handle) throws HandleNotRecognisedException {

```

```

68         // TODO Auto-generated method stub
69         accounts.remove(getAccountByHandle(handle));
70         throw new HandleNotRecognisedException();
71     }
72
73     @Override
74     public void changeAccountHandle(String oldHandle, String newHandle)
75         throws HandleNotRecognisedException, IllegalHandleException,
76         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         accountToChange.updateHandle(newHandle);
89
90     }
91
92     @Override
93     public void updateAccountDescription(String handle, String description) throws
94     HandleNotRecognisedException {
95         // TODO Auto-generated method stub
96         Account accountToChange = getAccountByHandle(handle);
97         if (accountToChange == null){
98             throw new HandleNotRecognisedException();
99         }
100         accountToChange.updateDescription(handle);
101
102     }
103
104     @Override
105     public String showAccount(String handle) throws HandleNotRecognisedException {
106         // TODO Auto-generated method stub
107         Account account = getAccountByHandle(handle);
108         if (account == null){
109             throw new HandleNotRecognisedException();
110         }
111         return account.showAccount();
112     }
113
114     @Override
115     public int createPost(String handle, String message) throws
116     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,
134         NotActionablePostException {

```

```

133         // TODO Auto-generated method stub
134         // EP@exampleuser:Endorsed message
135         Account endorser = getAccountByHandle(handle);
136         if (endorser == null){
137             throw new HandleNotRecognisedException();
138         }
139         Post postToEndorse = getPostByID(id);
140         if(postToEndorse == null){
141             throw new PostIDNotRecognisedException();
142         }
143         if (postToEndorse instanceof Endorsement || postToEndorse instanceof EmptyPost){
144             throw new NotActionablePostException();
145         }
146         Endorsement newEndorsement = new Endorsement(endorser, postToEndorse);
147         postToEndorse.addEndorsements(newEndorsement);
148         posts.add(newEndorsement);
149         endorser.addPost(newEndorsement);
150         getAccountByHandle(postToEndorse.getAccountHandle()).increaseEndorsements();
151         return newEndorsement.getID();
152     }
153
154     @Override
155     public int commentPost(String handle, int id, String message) throws
156         HandleNotRecognisedException,
157         PostIDNotRecognisedException, NotActionablePostException,
158         InvalidPostException {
159         // TODO Auto-generated method stub
160         Account commenter = getAccountByHandle(handle);
161         if(commenter == null){
162             throw new HandleNotRecognisedException();
163         }
164         Post postToComment = getPostByID(id);
165         if(postToComment == null){
166             throw new PostIDNotRecognisedException();
167         }
168         if (postToComment instanceof Endorsement || postToComment instanceof EmptyPost){
169             throw new NotActionablePostException();
170         }
171         if(!isValidPost(message)){
172             throw new InvalidPostException();
173         }
174         Comment newComment = new Comment(commenter, message, postToComment);
175         posts.add(newComment);
176         postToComment.addComment(newComment);
177         commenter.addPost(newComment);
178         return newComment.getID();
179     }
180
181     @Override
182     public void deletePost(int id) throws PostIDNotRecognisedException {
183         // TODO Auto-generated method stub
184         Post postToRemove = getPostByID(id);
185         if (postToRemove == null){
186             throw new PostIDNotRecognisedException();
187         }
188         posts.removeAll(postToRemove.getEndorsements());
189         postToRemove.deleteEndorsements();
190         int index = posts.indexOf(postToRemove);
191         EmptyPost newEmptyPost = new EmptyPost(postToRemove);
192         posts.set(index, newEmptyPost);
193         getAccountByHandle(postToRemove.getAccountHandle()).removePost(postToRemove);
194         ArrayList<Comment> oldComments = postToRemove.getComments();
195         for (Comment c : oldComments){
196             c.setOriginalPost(newEmptyPost);
197         }
198     }
199
200     @Override
201     public String showIndividualPost(int id) throws PostIDNotRecognisedException {

```

```

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

```

```

269         boolean firstFound = false;
270         for (Post p : posts){
271             if (p instanceof Original || p instanceof Comment){
272                 if (firstFound){
273                     if (p.getNumEndorsements() > mostEndorsed.getNumEndorsements()){
274                         mostEndorsed = p;
275                     }
276                 }
277                 else{
278                     mostEndorsed = p;
279                 }
280             }
281         }
282         return mostEndorsed.getID();
283     }
284
285     @Override
286     public int getMostEndorsedAccount() {
287         // TODO Auto-generated method stub
288         Account mostEndorsed = null;
289         boolean firstFound = false;
290         for (Account a : accounts){
291             if (firstFound){
292                 if (a.getEndorseCount() > mostEndorsed.getEndorseCount()){
293                     mostEndorsed = a;
294                 }
295             }
296             else{
297                 mostEndorsed = a;
298             }
299         }
300         return mostEndorsed.getID();
301     }
302
303     @Override
304     public void erasePlatform() {
305         // TODO Auto-generated method stub
306         posts.clear();
307         accounts.clear();
308     }
309
310
311     @Override
312     public void savePlatform(String filename) throws IOException {
313         // TODO Auto-generated method stub
314         try{
315             Object arr[] = new Object[2];
316             arr[0] = accounts;
317             arr[1] = posts;
318             FileOutputStream fileOut = new FileOutputStream("platforms/" + filename +
319                 ".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     @Override
333     public void loadPlatform(String filename) throws IOException,
334         ClassNotFoundException {
335         // TODO Auto-generated method stub
336         try{

```

```

336         FileInputStream inStream = new FileInputStream("platforms/" + filename +
337             ".obj");
338         ObjectInputStream objIn = new ObjectInputStream(inStream);
339         Object arr[] = (Object[])objIn.readObject();
340         accounts = (ArrayList<Account>)arr[0];
341         posts = (ArrayList<Post>)arr[1];
342         objIn.close();
343     }
344     catch(IOException e){
345         throw e;
346     }
347     catch(ClassNotFoundException e){
348         throw e;
349     }
350     catch(Exception e){
351         System.out.println("something went wrong, man");
352     }
353 }
354
355 private boolean handleExists(String handle){
356     for (Account a : accounts){
357         if (a.getHandle().equals(handle)){
358             return true;
359         }
360     }
361     return false;
362 }
363
364 private boolean isValidHandle(String handle){
365     //no whitespace, no empty, no more than 30 characters
366     if (handle.isEmpty()){
367         return false;
368     }
369     if (handle.contains(" ")){
370         return false;
371     }
372     if (handle.length() > 30){
373         return false;
374     }
375     return true;
376 }
377
378 private boolean isValidPost(String content){
379     if(content.isEmpty()){
380         return false;
381     }
382     if (content.length() > 100){
383         return false;
384     }
385     return true;
386 }
387
388 private Account getAccountByHandle(String handle){
389     for (Account a : accounts){
390         if (a.getHandle().equals(handle)){
391             return a;
392         }
393     }
394     return null;
395 }
396
397 private Post getPostByID(int id){
398     for (Post p : posts){
399         if (p.getID() == id){
400             return p;
401         }
402     }
403     return null;

```

404 }
405 }
406 }