

1 SocialMedia.java

```
1 package socialmedia;
2
3 import java.io.IOException;
4 import java.util.ArrayList;
5 import java.io.FileInputStream;
6 import java.io.FileOutputStream;
7 import java.io.ObjectInputStream;
8 import java.io.ObjectOutputStream;
9 import java.util.Dictionary;
10 import java.util.Hashtable;
11
12 public class SocialMedia implements SocialMediaPlatform {
13     private ArrayList<Account> accountsList = new ArrayList<Account>();
14     private ArrayList<DeletedPost> deletedPostsList = new ArrayList<DeletedPost>();
15     private ArrayList<DeletedComment> deletedCommentsList = new ArrayList<DeletedComment>();
16
17     @Override
18     public int createAccount(String handle) throws IllegalHandleException, InvalidHandleException {
19         // First checks if the newHandle exists in the system, if so, throw IllegalHandleException
20         for (Account acc : accountsList) {
21             if (acc.getHandle() == handle) {
22                 throw new IllegalHandleException();
23             }
24         }
25         // Check if the new handle is legal within the system spec
26         if (handle == "" || handle == null || handle.length() > 30 || handle.contains(" ")) {
27             throw new InvalidHandleException();
28         }
29         // Creates new account object, with only a handle, which is added to accountsList
30         accountsList.add(new Account(handle));
31         return accountsList.get(accountsList.size() - 1).getAccountID();
32     }
33
34
35     @Override
36     public int createAccount(String handle, String description) throws IllegalHandleException, InvalidHandleException {
37         // First checks if the newHandle exists in the system, if so, throw IllegalHandleException
38         for (Account acc : accountsList) {
39             if (acc.getHandle() == handle) {
40                 throw new IllegalHandleException();
41             }
42         }
43         // Check if the new handle is legal within the system spec
44         if (handle == "" || handle == null || handle.length() > 30 || handle.contains(" ")) {
45             throw new InvalidHandleException();
46         }
47         // Creates new account object, with a handle and a description, which is added to accountsList
48         accountsList.add(new Account(handle, description));
49         return accountsList.get(accountsList.size() - 1).getAccountID();
50     }
51
52     @Override
53     public void removeAccount(int id) throws AccountIDNotRecognisedException {
54         for (int accPos = 0; accPos < accountsList.size(); accPos++) {
55             // Finds the account with the id
56             if (accountsList.get(accPos).getAccountID() == id) {
57                 // Deletes all posts, comments and endorsements of the account
58                 try {
59                     ArrayList<Post> tempPosts = new ArrayList<Post>(accountsList.get(accPos).getOriginalPosts());
60                     for (Post post : tempPosts) {
61                         deletePost(post.getPostID());
62                     }
63                     ArrayList<Comment> tempComments = new ArrayList<Comment>(accountsList.get(accPos).getComments());
64                     for (Comment comment : tempComments) {
65                         deletePost(comment.getPostID());
66                     }
67                     ArrayList<Endorsement> tempEndorsements = new ArrayList<Endorsement>(
68                         accountsList.get(accPos).getEndorsements());
69                     for (Endorsement endorse : tempEndorsements) {
70                         deletePost(endorse.getPostID());
71                     }
72                 } catch (PostIDNotRecognisedException e) {
73                     continue;
74                 }
75                 // Deletes the account with the id
76                 accountsList.remove(accountsList.get(accPos));
77                 return;
78             }
79         }
80         // If it has looped through here and not returned, then the id must not be in the system, so throws id not recognised.
81         throw new AccountIDNotRecognisedException();
82     }
83
84     @Override
85     public void removeAccount(String handle) throws HandleNotRecognisedException {
86         for (int accPos = 0; accPos < accountsList.size(); accPos++) {
87             // Finds the account with the handle
```

```

88         if (accountsList.get(accPos).getHandle() == handle) {
89             // Deletes all posts, comments and endorsements of the account
90             try {
91                 ArrayList<Post> tempPosts = new ArrayList<Post>(accountsList.get(accPos).getOriginalPosts());
92                 for (Post post : tempPosts) {
93                     deletePost(post.getPostID());
94                 }
95                 ArrayList<Comment> tempComments = new ArrayList<Comment>(accountsList.get(accPos).getComments());
96                 for (Comment comment : tempComments) {
97                     deletePost(comment.getPostID());
98                 }
99                 ArrayList<Endorsement> tempEndorsements = new ArrayList<Endorsement>({
100                     accountsList.get(accPos).getEndorsements());
101                 for (Endorsement endorse : tempEndorsements) {
102                     deletePost(endorse.getPostID());
103                 }
104             } catch (PostIDNotRecognisedException e) {
105                 continue;
106             }
107             // Deletes the account with the handle
108             accountsList.remove(accountsList.get(accPos));
109             return;
110         }
111     }
112     // If it has looped through here and not returned, then the handle must not be in the system, so throws handle not recognised.
113     throw new HandleNotRecognisedException();
114 }
115
116 @Override
117 public void changeAccountHandle(String oldHandle, String newHandle)
118     throws HandleNotRecognisedException, IllegalHandleException, InvalidHandleException {
119     // First checks if the newHandle exists in the system, if so, throw IllegalHandleException
120     for (Account acc : accountsList) {
121         if (acc.getHandle() == newHandle) {
122             throw new IllegalHandleException();
123         }
124     }
125     // Check if the new handle is legal within the system spec
126     if (newHandle == "" || newHandle == null || newHandle.length() > 30 || newHandle.contains(" ")) {
127         throw new InvalidHandleException();
128     }
129     for (int pos = 0; pos < accountsList.size(); pos++) {
130         // Finds the account with the oldHandle
131         if (accountsList.get(pos).getHandle() == oldHandle) {
132             // Changes the handle
133             accountsList.get(pos).setHandle(newHandle);
134             return;
135         }
136     }
137     // If it has looped through here and not returned, then the oldHandle must not be in the system, so throws handle not recognised.
138     throw new HandleNotRecognisedException();
139 }
140
141 @Override
142 public void updateAccountDescription(String handle, String description) throws HandleNotRecognisedException {
143     for (int pos = 0; pos < accountsList.size(); pos++) {
144         // Finds the account with the handle
145         if (accountsList.get(pos).getHandle() == handle) {
146             // Updates the description
147             accountsList.get(pos).setDescription(description);
148             return;
149         }
150     }
151     // If it has looped through here and not returned, then the handle must not be in the system, so throws handle not recognised.
152     throw new HandleNotRecognisedException();
153 }
154
155 @Override
156 public String showAccount(String handle) throws HandleNotRecognisedException {
157     int totalEndorsements = 0;
158     for (Account acc : accountsList) {
159         // Finds accounts with the handle
160         if (acc.getHandle() == handle) {
161             ArrayList<Post> postsList = acc.getOriginalPosts();
162             ArrayList<Comment> commentsList = acc.getComments();
163             // Finds sum of endorsements received
164             for (int x = 0; x < postsList.size(); x++) {
165                 totalEndorsements += postsList.get(x).getEndorsementsList().size();
166             }
167             for (int x = 0; x < commentsList.size(); x++) {
168                 totalEndorsements += commentsList.get(x).getEndorsementsList().size();
169             }
170             return String.format(
171                 "<pre>\nID: %d\nHandle: %s\nDescription: %s\nPost count: %d\nEndorse count: %s\n</pre>",
172                 acc.getAccountID(), acc.getHandle(), acc.getDescription(),
173                 (acc.getComments().size() + acc.getEndorsements().size() + acc.getOriginalPosts().size()),
174                 totalEndorsements);
175         }
176     }
177     // If it has looped through here and not returned, then the handle must not be in the system, so throws handle not recognised.
178     throw new HandleNotRecognisedException();

```

```

179 }
180
181 @Override
182 public int createPost(String handle, String message) throws HandleNotRecognisedException, InvalidPostException {
183     // Checks the message is inside the character limit
184     if (message.length() <= 100 && message != "" && message != null) {
185         for (int pos = 0; pos < accountsList.size(); pos++) {
186             // Finds account with the handle
187             if (accountsList.get(pos).getHandle() == handle) {
188                 return accountsList.get(pos).createOriginalPost(message);
189             }
190         }
191         throw new HandleNotRecognisedException();
192     } else {
193         throw new InvalidPostException();
194     }
195 }
196
197 @Override
198 public int endorsePost(String handle, int id)
199     throws HandleNotRecognisedException, PostIDNotRecognisedException, NotActionablePostException {
200     boolean handleExists = false;
201     boolean notActionable = false;
202     for (int pos = 0; pos < accountsList.size(); pos++) {
203         // Finds account with the handle
204         if (accountsList.get(pos).getHandle() == handle) {
205             handleExists = true;
206             int endorseID = -1;
207             for (Account acc : accountsList) {
208                 // Adds endorsement ID to post when the ID of the post being endorsed is found in an account
209                 int foundPostIDPos = acc.searchPost(id);
210                 if (foundPostIDPos != -1) {
211                     endorseID = accountsList.get(pos).createEndorsement("EP@" + acc.getHandle() + ": "
212                         + acc.getOriginalPosts().get(foundPostIDPos).getMessage(), id);
213                     acc.addEndorseIDtoPostAt(foundPostIDPos, endorseID);
214                     break;
215                 }
216                 int foundCommentIDPos = acc.searchComment(id);
217                 if (foundCommentIDPos != -1) {
218                     endorseID = accountsList.get(pos).createEndorsement(
219                         "EP@" + acc.getHandle() + ": " + acc.getComments().get(foundCommentIDPos).getMessage(),
220                         id);
221                     acc.addEndorseIDtoCommentAt(foundCommentIDPos, endorseID);
222                     break;
223                 }
224                 int foundEndorseIDPos = acc.searchEndorsement(id);
225                 if (foundEndorseIDPos != -1) {
226                     notActionable = true;
227                     break;
228                 }
229             }
230             if (endorseID != -1) {
231                 return endorseID;
232             }
233         }
234     }
235     if (!handleExists) {
236         throw new HandleNotRecognisedException();
237     } else if (notActionable) {
238         throw new NotActionablePostException();
239     }
240     throw new PostIDNotRecognisedException();
241 }
242
243 @Override
244 public int commentPost(String handle, int id, String message) throws HandleNotRecognisedException,
245     PostIDNotRecognisedException, NotActionablePostException, InvalidPostException {
246     // Checks the message is inside the character limit
247     if (message.length() <= 100 && message != "" && message != null) {
248         boolean handleExists = false;
249         boolean postIDExists = false;
250         boolean notActionable = false;
251         for (int pos = 0; pos < accountsList.size(); pos++) {
252             if (accountsList.get(pos).getHandle() == handle) {
253                 handleExists = true;
254                 int commentID = -1;
255                 for (Account acc : accountsList) {
256                     // Adds comment ID to post when the ID of the post being commented is found in an account
257                     int foundPostIDPos = acc.searchPost(id);
258                     if (foundPostIDPos != -1) {
259                         postIDExists = true;
260                         commentID = accountsList.get(pos).createComment(message, id);
261                         acc.addCommentIDtoPostAt(foundPostIDPos, commentID);
262                         break;
263                     }
264                     int foundCommentIDPos = acc.searchComment(id);
265                     if (foundCommentIDPos != -1) {
266                         postIDExists = true;
267                         commentID = accountsList.get(pos).createComment(message, id);
268                         acc.addCommentIDtoCommentAt(foundCommentIDPos, commentID);
269                         break;

```

```

270     }
271         int foundEndorseIDPos = acc.searchEndorsement(id);
272         if (foundEndorseIDPos != -1) {
273             postIDExists = true;
274             notActionable = true;
275             break;
276         }
277     }
278     if (commentID != -1) {
279         return commentID;
280     }
281 }
282 }
283 if (!handleExists) {
284     throw new HandleNotRecognisedException();
285 } else if (notActionable) {
286     throw new NotActionablePostException();
287 } else if (!postIDExists) {
288     throw new PostIDNotRecognisedException();
289 }
290 }
291 throw new InvalidPostException();
292 }
293
294 @Override
295 public void deletePost(int id) throws PostIDNotRecognisedException {
296     int postPos = -1;
297     int endorsePos = -1;
298     int commentPos = -1;
299     int accPos = 0;
300     while (accPos < accountsList.size()) {
301         // Looks for post, endorsement or comment with the ID
302         postPos = accountsList.get(accPos).searchPost(id);
303         endorsePos = accountsList.get(accPos).searchEndorsement(id);
304         commentPos = accountsList.get(accPos).searchComment(id);
305         // Stops searching when object has been found
306         if (postPos != -1 || endorsePos != -1 || commentPos != -1) {
307             break;
308         }
309         accPos++;
310     }
311     // Throws exception if the object doesn't exist
312     if (postPos == -1 && endorsePos == -1 && commentPos == -1) {
313         throw new PostIDNotRecognisedException();
314     }
315     // ID found in original posts
316     if (postPos != -1) {
317         Post oldPost = accountsList.get(accPos).getOriginalPosts().get(postPos);
318         // Placeholder creation, doesn't create if there are no comments
319         if (oldPost.getCommentsList().size() != 0) {
320             deletedPostsList.add(new DeletedPost(oldPost.getPostID(), oldPost.getCommentsList()));
321         }
322         ArrayList<Integer> oldEndorsements = oldPost.getEndorsementsList();
323         // Looks through accounts again to find endorsements made on the post
324         for (Account acc : accountsList) {
325             ArrayList<Endorsement> endorsements = new ArrayList<Endorsement>{acc.getEndorsements()};
326             for (Endorsement endorse : endorsements) {
327                 // Checks if the ID is in the list of endorsement IDs to be deleted
328                 if (oldEndorsements.contains(endorse.getPostID())) {
329                     // Deletes endorsement object
330                     acc.getEndorsements().remove(endorse);
331                     // Removes from endorsements to be deleted list since it has been found
332                     oldEndorsements.remove((Integer) endorse.getPostID());
333                 }
334             }
335             // Stops early when all endorsements have been found and deleted
336             if (oldEndorsements.size() == 0) {
337                 break;
338             }
339         }
340         // Removes the post object
341         accountsList.get(accPos).removePostAt(postPos);
342     } else if (endorsePos != -1) { // ID found in endorsements
343         Endorsement oldEndorsement = accountsList.get(accPos).getEndorsements().get(endorsePos);
344         for (Account acc : accountsList) {
345             // Looks for post or comment with the post reference ID
346             int postPos2 = acc.searchPost(oldEndorsement.getPostReferenceID());
347             int commentPos2 = acc.searchComment(oldEndorsement.getPostReferenceID());
348             // Deletes endorsement ID from post
349             if (postPos2 != -1) {
350                 acc.removeEndorseIDinPostAt(postPos2, oldEndorsement.getPostID());
351                 break;
352             } else if (commentPos2 != -1) {
353                 acc.removeEndorseIDinCommentAt(commentPos2, oldEndorsement.getPostID());
354                 break;
355             }
356         }
357         // Removes the endorsement object
358         accountsList.get(accPos).removeEndorsementAt(endorsePos);
359     } else if (commentPos != -1) { // ID found in comments
360         Comment oldComment = accountsList.get(accPos).getComments().get(commentPos);

```

```

// If true, deletes the id from reference post
boolean emptyComments = true;
boolean commentFound = false;
// Placeholder creation, doesn't create if there are no comments
if (oldComment.getCommentsList().size() != 0) {
    emptyComments = false;
    deletedCommentsList.add(new DeletedComment(oldComment.getPostID(), oldComment.getCommentsList(),
        oldComment.getPostReferenceID())); // placeholder post
}
ArrayList<Integer> oldEndorsements = oldComment.getEndorsementsList();
// Checks the placeholder posts for comment ID
if (emptyComments) {
    for (int deletedPostPos = 0; deletedPostPos < deletedPostsList.size(); deletedPostPos++) {
        DeletedPost deletedPost = deletedPostsList.get(deletedPostPos);
        if (deletedPost.searchComment(id) != -1) {
            // Removes ID from the reference post
            deletedPost.removeCommentIDAt(deletedPost.searchComment(id));
            // Deletes the placeholder if it doesn't contain any comments
            if (deletedPost.getCommentsList().size() == 0) {
                deletedPostsList.remove(deletedPostPos);
            }
            commentFound = true;
            break;
        }
    }
}
if (!commentFound) {
    for (int deletedCommentPos = 0; deletedCommentPos < deletedCommentsList
        .size(); deletedCommentPos++) {
        // Placeholder storing the id for the comment to be deleted
        DeletedComment deletedComment = deletedCommentsList.get(deletedCommentPos);
        if (deletedComment.searchComment(id) != -1) {
            deletedComment.removeCommentIDAt(deletedComment.searchComment(id));
            // Deletes newly empty DeletedComment objects and their ID
            int change = 0;
            while (deletedCommentsList.size() != 0) {
                ArrayList<DeletedComment> tempDeletedComments = new ArrayList<DeletedComment> (
                    deletedCommentsList);
                for (DeletedComment del : tempDeletedComments) {
                    if (del.getCommentsList().size() == 0) {
                        boolean deletedCommentFound = false;
                        // Looks in accounts
                        for (Account acc : accountsList) {
                            boolean found = false;
                            for (Post post : acc.getOriginalPosts()) {
                                if (post.getCommentsList().contains(del.getPostID())) {
                                    post.getCommentsList().remove((Integer) del.getPostID());
                                    found = true;
                                    deletedCommentFound = true;
                                    break;
                                }
                            }
                        }
                        for (Comment comment : acc.getComments()) {
                            if (comment.getCommentsList().contains(del.getPostID())) {
                                comment.getCommentsList().remove((Integer) del.getPostID());
                                found = true;
                                deletedCommentFound = true;
                                break;
                            }
                        }
                        if (found) {
                            break;
                        }
                    }
                }
            }
            // Looks in deletedposts
            if (!deletedCommentFound) {
                for (int deletedPostPos = 0; deletedPostPos < deletedPostsList
                    .size(); deletedPostPos++) {
                    DeletedPost deletedPost = deletedPostsList.get(deletedPostPos);
                    if (deletedPost.searchComment(del.getPostID()) != -1) {
                        // Removes ID from the reference post
                        deletedPost.removeCommentIDAt(
                            deletedPost.searchComment(del.getPostID()));
                        // Deletes the placeholder if it doesn't contain any comments
                        if (deletedPost.getCommentsList().size() == 0) {
                            deletedPostsList.remove(deletedPostPos);
                        }
                        deletedCommentFound = true;
                        break;
                    }
                }
            }
        }
    }
    // Looks in deletedcomments
    if (!deletedCommentFound) {
        for (int deletedCommentPos2 = 0; deletedCommentPos2 < deletedCommentsList
            .size(); deletedCommentPos2++) {
            DeletedComment deletedComment2 = deletedCommentsList
                .get(deletedCommentPos2);
            if (deletedComment2.searchComment(del.getPostID()) != -1) {
                // Removes ID from the reference post
                deletedComment2.removeCommentIDAt(

```

```
451 deletedComment2.searchComment(del.getPostID()));
452
453 change++;
454 break;
455 }
456 }
457 deletedCommentsList.remove(del);
458 }
459 }
460 if (change == 0) {
461     break;
462 }
463 change = 0;
464 }
465 commentFound = true;
466 break;
467 }
468 }
469 }
470 }
471 for (Account acc : accountsList) {
472     if (oldEndorsements.size() != 0) {
473         ArrayList<Endorsement> endorsements = new ArrayList<Endorsement>(acc.getEndorsements());
474         // Looks through accounts again to find endorsements made on the post
475         for (Endorsement endorse : endorsements) {
476             // Checks if the ID is in the list of endorsement IDs to be deleted
477             if (oldEndorsements.contains(endorse.getPostID())) {
478                 // Deletes endorsement object
479                 acc.getEndorsements().remove(endorse);
480                 // Removes from endorsements to be deleted list since it has been found
481                 oldEndorsements.remove((Integer) endorse.getPostID());
482             }
483         }
484     }
485     if (!commentFound && emptyComments) {
486         // Looks for post or comment with the post reference ID
487         int postPos2 = acc.searchPost(oldComment.getPostReferenceID());
488         int commentPos2 = acc.searchComment(oldComment.getPostReferenceID());
489         // Deletes comment ID from post
490         if (postPos2 != -1) {
491             acc.removeCommentIDinPostAt(postPos2, oldComment.getPostID());
492             commentFound = true;
493         } else if (commentPos2 != -1) {
494             acc.removeCommentIDinCommentAt(commentPos2, oldComment.getPostID());
495             commentFound = true;
496         }
497     }
498     // Stops early when all endorsements are deleted and the ID has been removed from reference post
499     if (oldEndorsements.size() == 0 && commentFound) {
500         break;
501     }
502 }
503 // Removes the comment object
504 accountsList.get(accPos).removeCommentAt(commentPos);
505 }
506 }
507
508 @Override
509 public String showIndividualPost(int id) throws PostIDNotRecognisedException {
510     for (int pos = 0; pos < accountsList.size(); pos++) {
511         // Formatted string for original post
512         ArrayList<Post> postsList = accountsList.get(pos).getOriginalPosts();
513         for (int x = 0; x < postsList.size(); x++) {
514             if (postsList.get(x).getPostID() == id) {
515                 return String.format(
516                     "<pre>\nID: %d\nAccount: %s\nNo. endorsements: %s | No. comments: %s\n%s\n</pre>", id,
517                     accountsList.get(pos).getHandle(), accountsList.get(pos).getEndorsements().size(),
518                     accountsList.get(pos).getComments().size(), postsList.get(x).getMessage());
519             }
520         }
521         // Formatted string for comment
522         ArrayList<Comment> commentsList = accountsList.get(pos).getComments();
523         for (int x = 0; x < commentsList.size(); x++) { // for every comment in comments list
524             if (commentsList.get(x).getPostID() == id) { // if the comment has the ID we are looking for
525                 return String.format(
526                     "<pre>\nID: %d\nAccount: %s\nNo. endorsements: %s | No. comments: %s\n%s\n</pre>", id,
527                     accountsList.get(pos).getHandle(), commentsList.get(x).getEndorsementsList().size(),
528                     commentsList.get(x).getCommentsList().size(), commentsList.get(x).getMessage());
529             }
530         }
531         // Formatted string for endorsement
532         ArrayList<Endorsement> endorsementsList = accountsList.get(pos).getEndorsements();
533         for (int x = 0; x < endorsementsList.size(); x++) {
534             if (endorsementsList.get(x).getPostID() == id) {
535                 return String.format(
536                     "<pre>\nID: %d\nAccount: %s\nNo. endorsements: N/A | No. comments: N/A\n%s\n</pre>", id,
537                     accountsList.get(pos).getHandle(), endorsementsList.get(x).getMessage());
538             }
539         }
540     }
```

```

541 // Formatted string for deleted comments
542 for (int x = 0; x < deletedCommentsList.size(); x++) {
543     if (deletedCommentsList.get(x).getPostID() == id) {
544         return String.format(
545             "<pre>\nID: %d\nAccount: N/A\nNo. endorsements: N/A | No. comments: %s\n%s\n</pre>", id,
546             deletedCommentsList.get(x).getCommentsList().size(), deletedCommentsList.get(x).getMessage());
547     }
548 }
549 // Formatted string for deleted posts
550 for (int x = 0; x < deletedPostsList.size(); x++) {
551     if (deletedPostsList.get(x).getPostID() == id) {
552         return String.format(
553             "<pre>\nID: %d\nAccount: N/A\nNo. endorsements: N/A | No. comments: %s\n%s\n</pre>", id,
554             deletedPostsList.get(x).getCommentsList().size(), deletedPostsList.get(x).getMessage());
555     }
556 }
557 // Now it has looped through all the lists of each account and deleted lists, if
558 // not found, we now the post does not exist, so throw post ID not recognised
559 // exception
560 throw new PostIDNotRecognisedException();
561 }
562
563 private int indentationNumber = -1;
564 // Stores IDs which have been checked
565 private ArrayList<Integer> checkedElements = new ArrayList<Integer>();
566
567 @Override
568 public StringBuilder showPostChildrenDetails(int id)
569     throws PostIDNotRecognisedException, NotActionablePostException {
570     // The following 3 for loops check each list to see if the post is non
571     // actionable, creates and sets a flag to false if actionable, false if not
572     // actionable. They also set a flag postIDRecognised to true if the post ID is
573     // found in a list.
574     Boolean notActionable = false;
575     Boolean postIDRecognised = false;
576     for (int pos = 0; pos < accountsList.size(); pos++) {
577         ArrayList<Post> postsList = accountsList.get(pos).getOriginalPosts();
578         for (int x = 0; x < postsList.size(); x++) {
579             if (postsList.get(x).getPostID() == id) {
580                 notActionable = false;
581                 postIDRecognised = true;
582             }
583         }
584         ArrayList<Comment> commentsList = accountsList.get(pos).getComments();
585         for (int x = 0; x < commentsList.size(); x++) {
586             if (commentsList.get(x).getPostID() == id) {
587                 notActionable = false;
588                 postIDRecognised = true;
589             }
590         }
591         ArrayList<Endorsement> endorsementsList = accountsList.get(pos).getEndorsements();
592         for (int x = 0; x < endorsementsList.size(); x++) {
593             if (endorsementsList.get(x).getPostID() == id) {
594                 notActionable = true;
595                 postIDRecognised = true;
596             }
597         }
598     }
599     for (int x = 0; x < deletedPostsList.size(); x++) {
600         if (deletedPostsList.get(x).getPostID() == id) {
601             notActionable = false;
602             postIDRecognised = true;
603         }
604     }
605     for (int x = 0; x < deletedCommentsList.size(); x++) {
606         if (deletedCommentsList.get(x).getPostID() == id) {
607             notActionable = false;
608             postIDRecognised = true;
609         }
610     }
611     // I can then do 2 simple if statements to throw the correct errors should the
612     // booleans be fulfilled
613     if (!postIDRecognised) {
614         throw new PostIDNotRecognisedException();
615     }
616     if (notActionable) {
617         throw new NotActionablePostException();
618     }
619     StringBuilder string = new StringBuilder();
620     string.append("<pre>\n");
621     string.append(recursiveApproach(id));
622     string.append("\n</pre>");
623     indentationNumber = -1;
624     checkedElements.clear();
625     return string;
626 }
627
628 public StringBuilder recursiveApproach(int id) {
629     indentationNumber += 1;
630     // Looks through all accounts
631     for (int pos = 0; pos < accountsList.size(); pos++) {

```

```

632 ArrayList<Post> postsList = accountsList.get(pos).getOriginalPosts();
633 // Looks through all original posts of the account
634 for (int x = 0; x < postsList.size(); x++) {
635     if (postsList.get(x).getPostID() == id && !checkedElements.contains(id)) {
636         // Adds ID since it is being checked
637         checkedElements.add(id);
638         StringBuilder string = new StringBuilder();
639         String tab = "\t";
640         int changeableIndentationNumber = indentationNumber;
641         changeableIndentationNumber = indentationNumber;
642         if (changeableIndentationNumber != 0) {
643             string.append("\n");
644         }
645         changeableIndentationNumber = indentationNumber;
646         while (changeableIndentationNumber > 1) {
647             string.append(tab);
648             changeableIndentationNumber -= 1;
649         }
650         string.append("ID:");
651         string.append(Integer.toString(id) + "\n");
652         changeableIndentationNumber = indentationNumber;
653         while (changeableIndentationNumber > 0) {
654             string.append(tab);
655             changeableIndentationNumber -= 1;
656         }
657         for (int a = 0; a < accountsList.size(); a++) {
658             if (accountsList.get(a).getAccountID() == postsList.get(x).getAccountID()) {
659                 changeableIndentationNumber = indentationNumber;
660                 while (changeableIndentationNumber > 1) {
661                     string.append(tab);
662                     changeableIndentationNumber -= 1;
663                 }
664                 string.append("Account: " + accountsList.get(a).getHandle() + "\n");
665             }
666         }
667         changeableIndentationNumber = indentationNumber;
668         while (changeableIndentationNumber > 0) {
669             string.append(tab);
670             changeableIndentationNumber -= 1;
671         }
672         string.append("No. endorsements: " + Integer.toString(postsList.get(x).getEndorsementsList().size())
673             + " | No. comments: " + Integer.toString(postsList.get(x).getCommentsList().size()) + "\n");
674         changeableIndentationNumber = indentationNumber;
675         while (changeableIndentationNumber > 0) {
676             string.append(tab);
677             changeableIndentationNumber -= 1;
678         }
679         string.append(postsList.get(x).getMessage());
680
681         for (int y = 0; y < postsList.get(x).getCommentsList().size(); y++) {
682             ArrayList<Integer> childElementsList = postsList.get(x).getCommentsList();
683             for (int z = 0; z < childElementsList.size(); z++) {
684                 changeableIndentationNumber = indentationNumber;
685                 if (changeableIndentationNumber != 0) {
686                     string.append("\n");
687                 }
688                 changeableIndentationNumber = indentationNumber + 1;
689                 while (changeableIndentationNumber > 1) {
690                     string.append(tab);
691                     changeableIndentationNumber -= 1;
692                 }
693                 changeableIndentationNumber = indentationNumber;
694                 if (changeableIndentationNumber != 0) {
695                     string.append(" | ");
696                 }
697                 string.append(recursiveApproach(childElementsList.get(z)));
698             }
699         }
700         indentationNumber -= 1;
701         return string;
702     }
703 }
704 ArrayList<Comment> commentsList = accountsList.get(pos).getComments();
705 for (int x = 0; x < commentsList.size(); x++) {
706     if (commentsList.get(x).getPostID() == id && !checkedElements.contains(id)) {
707         checkedElements.add(id);
708         StringBuilder string = new StringBuilder();
709         String tab = "\t";
710         int changeableIndentationNumber = indentationNumber;
711         if (indentationNumber != 0) {
712             changeableIndentationNumber = indentationNumber;
713             if (changeableIndentationNumber != 0) {
714                 string.append("\n");
715             }
716             changeableIndentationNumber = indentationNumber;
717             while (changeableIndentationNumber > 1) {
718                 string.append(tab);
719                 changeableIndentationNumber -= 1;
720             }
721             string.append(" | " + tab + "> ID:");
722             string.append(Integer.toString(id) + "\n");

```



```

723 changeableIndentationNumber = indentationNumber;
724 while (changeableIndentationNumber > 1) {
725     string.append(tab);
726     changeableIndentationNumber -= 1;
727 }
728 for (int a = 0; a < accountsList.size(); a++) {
729     if (accountsList.get(a).getAccountID() == commentsList.get(x).getAccountID()) {
730         changeableIndentationNumber = indentationNumber;
731         while (changeableIndentationNumber > 1) {
732             string.append(tab);
733             changeableIndentationNumber -= 1;
734         }
735         changeableIndentationNumber = indentationNumber;
736         if (changeableIndentationNumber == 1) {
737             string.append(tab);
738         }
739         string.append("Account: " + accountsList.get(a).getHandle() + "\n");
740     }
741 }
742 changeableIndentationNumber = indentationNumber;
743 while (changeableIndentationNumber > 1) {
744     string.append(tab);
745     changeableIndentationNumber -= 1;
746 }
747 string.append(tab + "No. endorsements: "
748             + Integer.toString(commentsList.get(x).getEndorsementsList().size())
749             + " | No. comments: " + Integer.toString(commentsList.get(x).getCommentsList().size())
750             + "\n");
751 changeableIndentationNumber = indentationNumber;
752 while (changeableIndentationNumber > 1) {
753     string.append(tab);
754     changeableIndentationNumber -= 1;
755 }
756 string.append(tab + commentsList.get(x).getMessage());
757 }
758 if (indentationNumber == 0) {
759     changeableIndentationNumber = indentationNumber;
760     if (changeableIndentationNumber != 0) {
761         string.append("\n");
762     }
763     changeableIndentationNumber = indentationNumber;
764     while (changeableIndentationNumber > 1) {
765         string.append(tab);
766         changeableIndentationNumber -= 1;
767     }
768     string.append("ID:");
769     string.append(Integer.toString(id) + "\n");
770     changeableIndentationNumber = indentationNumber;
771     while (changeableIndentationNumber > 1) {
772         string.append(tab);
773         changeableIndentationNumber -= 1;
774     }
775     string.append("Account: " + Integer.toString(commentsList.get(x).getAccountID()) + "\n");
776     changeableIndentationNumber = indentationNumber;
777     while (changeableIndentationNumber > 1) {
778         string.append(tab);
779         changeableIndentationNumber -= 1;
780     }
781     string.append("No. endorsements: "
782             + Integer.toString(commentsList.get(x).getEndorsementsList().size())
783             + " | No. comments: " + Integer.toString(commentsList.get(x).getCommentsList().size())
784             + "\n");
785     changeableIndentationNumber = indentationNumber;
786     while (changeableIndentationNumber > 1) {
787         string.append(tab);
788         changeableIndentationNumber -= 1;
789     }
790     string.append(commentsList.get(x).getMessage());
791 }
792
793 for (int y = 0; y < commentsList.get(x).getCommentsList().size(); y++) {
794     ArrayList<Integer> childElementsList = commentsList.get(x).getCommentsList();
795     for (int z = 0; z < childElementsList.size(); z++) {
796         changeableIndentationNumber = indentationNumber;
797         if (changeableIndentationNumber != 0) {
798             string.append("\n");
799         }
800         changeableIndentationNumber = indentationNumber + 1;
801         while (changeableIndentationNumber > 1) {
802             string.append(tab);
803             changeableIndentationNumber -= 1;
804         }
805         changeableIndentationNumber = indentationNumber;
806         if (changeableIndentationNumber != 0) {
807             string.append("|");
808         }
809         string.append(recursiveApproach(childElementsList.get(z)));
810     }
811 }
812 indentationNumber -= 1;
813 return string;

```

```

814     }
815 }
816 }
817 ArrayList<DeletedComment> deleteCommentsListObj = deletedCommentsList;
818 for (int x = 0; x < deleteCommentsListObj.size(); x++) {
819     if (deleteCommentsListObj.get(x).getPostID() == id && !checkedElements.contains(id)) {
820         checkedElements.add(id);
821         StringBuilder string = new StringBuilder();
822         String tab = "\t";
823         int changeableIndentationNumber = indentationNumber;
824         if (indentationNumber != 0) {
825             changeableIndentationNumber = indentationNumber;
826             if (changeableIndentationNumber != 0) {
827                 string.append("\n");
828             }
829             changeableIndentationNumber = indentationNumber;
830             while (changeableIndentationNumber > 1) {
831                 string.append(tab);
832                 changeableIndentationNumber -= 1;
833             }
834             string.append("| " + tab + "> ID:");
835             string.append(Integer.toString(id) + "\n");
836             changeableIndentationNumber = indentationNumber;
837             while (changeableIndentationNumber > 1) {
838                 string.append(tab);
839                 changeableIndentationNumber -= 1;
840             }
841             changeableIndentationNumber = indentationNumber;
842             while (changeableIndentationNumber > 1) {
843                 string.append(tab);
844                 changeableIndentationNumber -= 1;
845             }
846             changeableIndentationNumber = indentationNumber;
847             if (changeableIndentationNumber == 1) {
848                 string.append(tab);
849             }
850             string.append("Account: N/A \n");
851             changeableIndentationNumber = indentationNumber;
852             while (changeableIndentationNumber > 1) {
853                 string.append(tab);
854                 changeableIndentationNumber -= 1;
855             }
856             string.append(tab + "No. endorsements: N/A | No. comments: "
857                 + Integer.toString(deleteCommentsListObj.get(x).getCommentsList().size()) + "\n");
858             changeableIndentationNumber = indentationNumber;
859             while (changeableIndentationNumber > 1) {
860                 string.append(tab);
861                 changeableIndentationNumber -= 1;
862             }
863             string.append(tab + deleteCommentsListObj.get(x).getMessage());
864         }
865         if (indentationNumber == 0) {
866             changeableIndentationNumber = indentationNumber;
867             if (changeableIndentationNumber != 0) {
868                 string.append("\n");
869             }
870             changeableIndentationNumber = indentationNumber;
871             while (changeableIndentationNumber > 1) {
872                 string.append(tab);
873                 changeableIndentationNumber -= 1;
874             }
875             string.append("ID:");
876             string.append(Integer.toString(id) + "\n");
877             changeableIndentationNumber = indentationNumber;
878             while (changeableIndentationNumber > 1) {
879                 string.append(tab);
880                 changeableIndentationNumber -= 1;
881             }
882             string.append("Account: N/A \n");
883             changeableIndentationNumber = indentationNumber;
884             while (changeableIndentationNumber > 1) {
885                 string.append(tab);
886                 changeableIndentationNumber -= 1;
887             }
888             string.append("No. endorsements: N/A | No. comments: "
889                 + Integer.toString(deleteCommentsListObj.get(x).getCommentsList().size()) + "\n");
890             changeableIndentationNumber = indentationNumber;
891             while (changeableIndentationNumber > 1) {
892                 string.append(tab);
893                 changeableIndentationNumber -= 1;
894             }
895             string.append(deleteCommentsListObj.get(x).getMessage());
896         }
897     }
898     for (int y = 0; y < deleteCommentsListObj.get(x).getCommentsList().size(); y++) {
899         ArrayList<Integer> childElementsList = deleteCommentsListObj.get(x).getCommentsList();
900         for (int z = 0; z < childElementsList.size(); z++) {
901             changeableIndentationNumber = indentationNumber;
902             if (changeableIndentationNumber != 0) {
903                 string.append("\n");
904             }

```

```

905         changeableIndentationNumber = indentationNumber + 1;
906         while (changeableIndentationNumber > 1) {
907             string.append(tab);
908             changeableIndentationNumber -= 1;
909         }
910         changeableIndentationNumber = indentationNumber;
911         if (changeableIndentationNumber != 0) {
912             string.append("|");
913         }
914         string.append(recursiveApproach(childElementsList.get(z)));
915     }
916 }
917 indentationNumber -= 1;
918 return string;
919 }
920 }
921 ArrayList<DeletedPost> deletePostsListObj = deletedPostsList;
922 for (int x = 0; x < deletePostsListObj.size(); x++) {
923     if (deletePostsListObj.get(x).getPostID() == id && !checkedElements.contains(id)) {
924         checkedElements.add(id);
925         StringBuilder string = new StringBuilder();
926         String tab = "\t";
927         int changeableIndentationNumber = indentationNumber;
928         if (indentationNumber != 0) {
929             changeableIndentationNumber = indentationNumber;
930             if (changeableIndentationNumber != 0) {
931                 string.append("\n");
932             }
933             changeableIndentationNumber = indentationNumber;
934             while (changeableIndentationNumber > 1) {
935                 string.append(tab);
936                 changeableIndentationNumber -= 1;
937             }
938             string.append("|" + tab + "> ID:");
939             string.append(Integer.toString(id) + "\n");
940             changeableIndentationNumber = indentationNumber;
941             while (changeableIndentationNumber > 1) {
942                 string.append(tab);
943                 changeableIndentationNumber -= 1;
944             }
945             changeableIndentationNumber = indentationNumber;
946             while (changeableIndentationNumber > 1) {
947                 string.append(tab);
948                 changeableIndentationNumber -= 1;
949             }
950             changeableIndentationNumber = indentationNumber;
951             if (changeableIndentationNumber == 1) {
952                 string.append(tab);
953             }
954             string.append("Account: N/A \n");
955             changeableIndentationNumber = indentationNumber;
956             while (changeableIndentationNumber > 1) {
957                 string.append(tab);
958                 changeableIndentationNumber -= 1;
959             }
960             string.append(tab + "No. endorsements: N/A | No. comments: "
961                 + Integer.toString(deletePostsListObj.get(x).getCommentsList().size()) + "\n");
962             changeableIndentationNumber = indentationNumber;
963             while (changeableIndentationNumber > 1) {
964                 string.append(tab);
965                 changeableIndentationNumber -= 1;
966             }
967             string.append(tab + deletePostsListObj.get(x).getMessage());
968         }
969         if (indentationNumber == 0) {
970             changeableIndentationNumber = indentationNumber;
971             if (changeableIndentationNumber != 0) {
972                 string.append("\n");
973             }
974             changeableIndentationNumber = indentationNumber;
975             while (changeableIndentationNumber > 1) {
976                 string.append(tab);
977                 changeableIndentationNumber -= 1;
978             }
979             string.append("ID:");
980             string.append(Integer.toString(id) + "\n");
981             changeableIndentationNumber = indentationNumber;
982             while (changeableIndentationNumber > 1) {
983                 string.append(tab);
984                 changeableIndentationNumber -= 1;
985             }
986             string.append("Account: N/A \n");
987             changeableIndentationNumber = indentationNumber;
988             while (changeableIndentationNumber > 1) {
989                 string.append(tab);
990                 changeableIndentationNumber -= 1;
991             }
992             string.append("No. endorsements: N/A | No. comments: "
993                 + Integer.toString(deletePostsListObj.get(x).getCommentsList().size()) + "\n");
994             changeableIndentationNumber = indentationNumber;
995             while (changeableIndentationNumber > 1) {

```

```

996         string.append(tab);
997         changeableIndentationNumber -= 1;
998     }
999     string.append(deletePostsListObj.get(x).getMessage());
1000 }
1001
1002     for (int y = 0; y < deletePostsListObj.get(x).getCommentsList().size(); y++) {
1003         ArrayList<Integer> childElementsList = deletePostsListObj.get(x).getCommentsList();
1004         for (int z = 0; z < childElementsList.size(); z++) {
1005             changeableIndentationNumber = indentationNumber;
1006             if (changeableIndentationNumber != 0) {
1007                 string.append("\n");
1008             }
1009             changeableIndentationNumber = indentationNumber + 1;
1010             while (changeableIndentationNumber > 1) {
1011                 string.append(tab);
1012                 changeableIndentationNumber -= 1;
1013             }
1014             changeableIndentationNumber = indentationNumber;
1015             if (changeableIndentationNumber != 0) {
1016                 string.append("|");
1017             }
1018             string.append(recursiveApproach(childElementsList.get(z)));
1019         }
1020     }
1021     indentationNumber -= 1;
1022     return string;
1023 }
1024 }
1025
1026     StringBuilder string = new StringBuilder();
1027     indentationNumber -= 1;
1028     return string;
1029 }
1030
1031 @Override
1032 public int getNumberOfAccounts() {
1033     return accountsList.size();
1034 }
1035
1036 @Override
1037 public int getTotalOriginalPosts() {
1038     int total = 0;
1039     for (int pos = 0; pos < accountsList.size(); pos++) {
1040         // Increments the size of original post list of the account
1041         total += accountsList.get(pos).getOriginalPosts().size();
1042     }
1043     return total;
1044 }
1045
1046 @Override
1047 public int getTotalEndorsmentPosts() {
1048     int total = 0;
1049     for (int pos = 0; pos < accountsList.size(); pos++) {
1050         // Increments the size of endorsement list of the account
1051         total += accountsList.get(pos).getEndorsements().size();
1052     }
1053     return total;
1054 }
1055
1056 @Override
1057 public int getTotalCommentPosts() {
1058     int total = 0;
1059     for (int pos = 0; pos < accountsList.size(); pos++) {
1060         // Increments the size of comment list of the account
1061         total += accountsList.get(pos).getComments().size();
1062     }
1063     return total;
1064 }
1065
1066 @Override
1067 public int getMostEndorsedPost() {
1068     // Stores the number of endorsements of the currently most endorsed post
1069     int largestNumber = -1;
1070     // Stores the ID of the most endorsed post
1071     int mostEndorsedPostID = -1;
1072     for (int pos = 0; pos < accountsList.size(); pos++) {
1073         // Looks through all original posts
1074         ArrayList<Post> postsList = accountsList.get(pos).getOriginalPosts();
1075         for (int x = 0; x < postsList.size(); x++) {
1076             // Compares the number endorsements with currently largest number
1077             if (postsList.get(x).getEndorsementsList().size() > largestNumber) {
1078                 largestNumber = postsList.get(x).getEndorsementsList().size();
1079                 mostEndorsedPostID = postsList.get(x).getPostID();
1080             }
1081         }
1082         // Looks through all comments
1083         ArrayList<Comment> commentsList = accountsList.get(pos).getComments();
1084         for (int x = 0; x < commentsList.size(); x++) {
1085             if (commentsList.get(x).getEndorsementsList().size() > largestNumber) {
1086                 largestNumber = commentsList.get(x).getEndorsementsList().size();
1087                 mostEndorsedPostID = commentsList.get(x).getPostID();
1088             }
1089         }
1090     }
1091     return mostEndorsedPostID;
1092 }

```

```

1087         }
1088     }
1089 }
1090     return mostEndorsedPostID;
1091 }
1092
1093 @Override
1094 public int getMostEndorsedAccount() {
1095     int maxNumberOfEndorsements = -1;
1096     int mostEndorsedID = -1;
1097     for (int pos = 0; pos < accountsList.size(); pos++) {
1098         // Stores the total number of endorsements for an account
1099         int numberOfEndorsements = 0;
1100         ArrayList<Post> postsList = accountsList.get(pos).getOriginalPosts();
1101         // Adds the number of endorsements for each post and comment
1102         for (int x = 0; x < postsList.size(); x++) {
1103             numberOfEndorsements += postsList.get(x).getEndorsementsList().size();
1104         }
1105         ArrayList<Comment> commentsList = accountsList.get(pos).getComments();
1106         for (int x = 0; x < commentsList.size(); x++) {
1107             numberOfEndorsements += commentsList.get(x).getEndorsementsList().size();
1108         }
1109         // Compares the number endorsements with currently largest number
1110         if (numberOfEndorsements > maxNumberOfEndorsements) {
1111             maxNumberOfEndorsements = numberOfEndorsements;
1112             mostEndorsedID = accountsList.get(pos).getAccountID();
1113         }
1114     }
1115     return mostEndorsedID;
1116 }
1117
1118 @Override
1119 public void erasePlatform() {
1120     // Resets post counter
1121     for (Account acc : accountsList) {
1122         if (acc.getOriginalPosts().size() != 0) {
1123             // Not called if there are no existing posts, so the counter was not changed
1124             acc.getOriginalPosts().get(0).resetNumberOfPosts();
1125             break;
1126         }
1127     }
1128     // Clears all the lists
1129     accountsList.clear();
1130     deletedCommentsList.clear();
1131     deletedPostsList.clear();
1132     // Also resets/clears other counters
1133     checkedElements.clear();
1134     indentationNumber = -1;
1135 }
1136
1137 @Override
1138 public void savePlatform(String filename) throws IOException {
1139     // Creates a dictionary to store the arraylists
1140     Dictionary<String, Object> all = new Hashtable<>();
1141     all.put("Accounts", accountsList);
1142     all.put("DeletedPosts", deletedPostsList);
1143     all.put("DeletedComments", deletedCommentsList);
1144     try {
1145         // Stores the dictionary into a file
1146         FileOutputStream file = new FileOutputStream(filename + ".ser");
1147         ObjectOutputStream out = new ObjectOutputStream(file);
1148         out.writeObject(all);
1149         out.close();
1150         file.close();
1151     } catch (IOException e) {
1152         e.printStackTrace();
1153     }
1154 }
1155
1156 @Override
1157 public void loadPlatform(String filename) throws IOException, ClassNotFoundException {
1158     Dictionary<String, ArrayList<Object>> all = null;
1159     try {
1160         // Retrieves a dictionary from a file
1161         FileInputStream file = new FileInputStream(filename + ".ser");
1162         ObjectInputStream in = new ObjectInputStream(file);
1163         all = (Dictionary<String, ArrayList<Object>>) in.readObject();
1164         in.close();
1165         file.close();
1166         // Replaces the arraylists with the ones in the file
1167         erasePlatform();
1168         for (Object obj : all.get("Accounts")) {
1169             accountsList.add((Account) obj);
1170         }
1171         for (Object obj : all.get("DeletedPosts")) {
1172             deletedPostsList.add((DeletedPost) obj);
1173         }
1174         for (Object obj : all.get("DeletedComments")) {
1175             deletedCommentsList.add((DeletedComment) obj);
1176         }
1177     } catch (IOException e) {

```

```
1178         e.printStackTrace();
1179     } catch (ClassNotFoundException e) {
1180         e.printStackTrace();
1181     }
1182
1183 }
1184
1185 }
```

2 Account.java

```
1 package socialmedia;
2
3 import java.io.Serializable;
4 import java.util.ArrayList;
5
6 public class Account implements Serializable {
7     private static final long serialVersionUID = 1L;
8
9     private static int numberOfAccounts = 0;
10    private int accountID;
11    private String handle;
12    private String description;
13    private ArrayList<Post> originalPostList = new ArrayList<Post>(); // account's posts
14    private ArrayList<Endorsement> endorsementsList = new ArrayList<Endorsement>(); // account's reposts of other's
15    private ArrayList<Comment> commentsList = new ArrayList<Comment>(); // account's comments
16
17    public Account(String handle, String description){
18        this.handle = handle;
19        this.description = description;
20        accountID = numberOfAccounts;
21        numberOfAccounts += 1;
22    }
23
24    public Account(String handle){
25        this.handle = handle;
26        this.description = "";
27        accountID = numberOfAccounts;
28        numberOfAccounts += 1;
29    }
30
31    public void setDescription(String description){
32        this.description = description;
33    }
34
35    public void setHandle(String handle){
36        this.handle = handle;
37    }
38
39    public String getDescription(){
40        return description;
41    }
42
43    public String getHandle(){
44        return handle;
45    }
46
47    public int getAccountID(){
48        return accountID;
49    }
50
51    public ArrayList<Post> getOriginalPosts() {
52        return originalPostList;
53    }
54
55    public ArrayList<Endorsement> getEndorsements() {
56        return endorsementsList;
57    }
58
59    public ArrayList<Comment> getComments() {
60        return commentsList;
61    }
62
63    public int createOriginalPost(String message) {
64        originalPostList.add(new Post(accountID, message));
65        return originalPostList.get(originalPostList.size()-1).getPostID();
66    }
67
68    public int createEndorsement(String message, int postReferenceID) {
69        endorsementsList.add(new Endorsement(accountID, message, postReferenceID));
70        return endorsementsList.get(endorsementsList.size()-1).getPostID();
71    }
72
73    public int createComment(String message, int postReferenceID) {
74        commentsList.add(new Comment(accountID, message, postReferenceID));
75        return commentsList.get(commentsList.size()-1).getPostID();
76    }
77
78    public void removePostAt(int pos){
79        originalPostList.remove(pos);
```

```

80 }
81
82 public void removeEndorsementAt(int pos){
83     endorsementsList.remove(pos);
84 }
85
86 public void removeCommentAt(int pos){
87     commentsList.remove(pos);
88 }
89
90 public int searchPost(int id) {
91     for (int pos = 0; pos < originalPostList.size(); pos++) {
92         if (originalPostList.get(pos).getPostID() == id) {
93             return pos;
94         }
95     }
96     return -1;
97 }
98
99 public int searchEndorsement(int id) {
100     for (int pos = 0; pos < endorsementsList.size(); pos++) {
101         if (endorsementsList.get(pos).getPostID() == id) {
102             return pos;
103         }
104     }
105     return -1;
106 }
107
108 public int searchComment(int id) {
109     for (int pos = 0; pos < commentsList.size(); pos++) {
110         if (commentsList.get(pos).getPostID() == id) {
111             return pos;
112         }
113     }
114     return -1;
115 }
116
117 public void addCommentIDtoPostAt(int pos, int commentID) {
118     originalPostList.get(pos).addCommentID(commentID);
119 }
120
121 public void addEndorseIDtoPostAt(int pos, int endorseID) {
122     originalPostList.get(pos).addEndorsementID(endorseID);
123 }
124
125 public void addCommentIDtoCommentAt(int pos, int commentID) {
126     commentsList.get(pos).addCommentID(commentID);
127 }
128
129 public void addEndorseIDtoCommentAt(int pos, int endorseID) {
130     commentsList.get(pos).addEndorsementID(endorseID);
131 }
132
133 public void removeCommentIDinPostAt(int pos, int commentID) {
134     originalPostList.get(pos).removeCommentID(commentID);;
135 }
136
137 public void removeEndorseIDinPostAt(int pos, int endorseID) {
138     originalPostList.get(pos).removeEndorsementID(endorseID);
139 }
140
141 public void removeCommentIDinCommentAt(int pos, int commentID) {
142     commentsList.get(pos).removeCommentID(commentID);
143 }
144
145 public void removeEndorseIDinCommentAt(int pos, int endorseID) {
146     commentsList.get(pos).removeEndorsementID(endorseID);;
147 }
148 }

```

3 Post.java

```

1 package socialmedia;
2
3 import java.io.Serializable;
4 import java.util.ArrayList;
5
6 public class Post implements Serializable {
7     private static final long serialVersionUID = 2L;
8
9     private int postID;
10    protected static int numberOfPosts = 0;
11    private int accountID;
12    private String message;
13    private ArrayList<Integer> endorsementsList = new ArrayList<Integer>();
14    private ArrayList<Integer> commentsList = new ArrayList<Integer>();
15
16    public Post(int accountID, String message){
17        this.accountID = accountID;
18        this.message = message;
19        postID = numberOfPosts;

```

```
20     numberOfPosts += 1;
21 }
22
23 public String getMessage(){
24     return message;
25 }
26
27 public int getPostID(){
28     return postID;
29 }
30
31 public int getAccountID(){
32     return accountID;
33 }
34
35 public void addEndorsementID(int endorsementID){
36     endorsementsList.add(Integer.valueOf(endorsementID));
37 }
38
39 public void addCommentID(int commentID){
40     commentsList.add(Integer.valueOf(commentID));
41 }
42
43 public ArrayList<Integer> getEndorsementsList(){
44     return endorsementsList;
45 }
46
47 public ArrayList<Integer> getCommentsList(){
48     return commentsList;
49 }
50
51 public void removeEndorsementID(int endorsementID){
52     for (int pos = 0; pos < endorsementsList.size(); pos++) {
53         if (endorsementsList.get(pos) == endorsementID) {
54             endorsementsList.remove(pos);
55             break;
56         }
57     }
58 }
59
60 public void removeCommentID(int commentID){
61     for (int pos = 0; pos < commentsList.size(); pos++) {
62         if (commentsList.get(pos) == commentID) {
63             commentsList.remove(pos);
64             break;
65         }
66     }
67 }
68
69 public void resetNumberOfPosts() {
70     numberOfPosts = 0;
71 }
72 }
```

4 Comment.java

```
1 package socialmedia;
2
3 public class Comment extends Post{
4     private static final long serialVersionUID = 3L;
5
6     private int postReferenceID;
7
8     public Comment(int accountID, String message, int postReferenceID) {
9         super(accountID, message);
10         this.postReferenceID = postReferenceID;
11     }
12
13     public int getPostReferenceID(){
14         return postReferenceID;
15     }
16 }
```

5 Endorsement.java

```
1 package socialmedia;
2
3 public class Endorsement extends Post {
4     private static final long serialVersionUID = 4L;
5
6     private int postReferenceID;
7
8     public Endorsement(int accountID, String message, int postReferenceID) {
9         super(accountID, message); //pass in the message of the post we are endorsing
10         this.postReferenceID = postReferenceID;
11     }
12
13     public int getPostReferenceID(){
```



```
14     return postReferenceID;
15 }
16 }
```

6 DeletedPost.java

```
1 package socialmedia;
2
3 import java.io.Serializable;
4 import java.util.ArrayList;
5
6 public class DeletedPost implements Serializable {
7     private static final long serialVersionUID = 5L;
8
9     private int postID; //postID of old post
10    private ArrayList<Integer> commentsList; //comments of the old post
11    protected String message = "The original content was removed from the system and is no longer available.";
12
13    public DeletedPost(int postID, ArrayList<Integer> commentsList){
14        this.postID = postID;
15        this.commentsList = commentsList;
16    }
17
18    public String getMessage(){
19        return message;
20    }
21
22    public int getPostID(){
23        return postID;
24    }
25
26    public ArrayList<Integer> getCommentsList(){
27        return commentsList;
28    }
29
30    public int searchComment(int id) {
31        for (int pos = 0; pos < commentsList.size(); pos++) {
32            if (commentsList.get(pos) == id) {
33                return pos;
34            }
35        }
36        return -1;
37    }
38
39    public void removeCommentIDAt(int pos){
40        commentsList.remove(pos);
41    }
42 }
```

7 DeletedComment.java

```
1 package socialmedia;
2
3 import java.util.ArrayList;
4
5 public class DeletedComment extends DeletedPost{
6     private static final long serialVersionUID = 6L;
7
8     private int postReferenceID;
9
10    public DeletedComment(int postID, ArrayList<Integer> commentsList, int postReferenceID) {
11        super(postID, commentsList);
12        this.postReferenceID = postReferenceID;
13    }
14
15    public int getPostReferenceID(){
16        return postReferenceID;
17    }
18 }
```