

# 1 Account.java

```
1 package socialmedia;
2 import java.util.*;
3 import java.io.Serializable;
4
5 /**
6  * Represents an individual account.
7  * @author Ashley Card
8  * @author Maryia Fralova
9  * @version 1.0
10 */
11 public class Account implements Serializable{
12
13     // Attributes
14     private String handle;
15     private int id;
16     private String description;
17     private HashMap<String, ArrayList<Integer>> storage;
18
19     // Getter methods
20
21     /** Gets account's handle.
22      * @return handle of account
23      */
24     public String getHandle(){
25         return handle;
26     }
27
28     /** Gets account's ID.
29      * @return ID of account
30      */
31     public int getID(){
32         return id;
33     }
34
35     /** Gets account's description.
36      * @return description of account
37      */
38     public String getDescription(){
39         return description;
40     }
41
42     /** Gets account's storage of posts.
43      * @return storage of account
44      */
45     public HashMap<String, ArrayList<Integer>> getAccountStorage(){
46         return storage;
47     }
48
49     // Setter methods
50
51     /** Sets account's handle.
52      * @param handle handle of account
```

```

53     */
54     public void setHandle(String handle){
55         this.handle = handle;
56     }
57
58     /** Sets account's description.
59      * @param description description of account
60      */
61     public void setDescription(String description){
62         this.description = description;
63     }
64
65     /** Adds original post ID to the account's storage of posts.
66      * @param idPost ID of post to be added to account original storage
67      */
68     public void addToAccountStorageOriginal(int idPost){
69         ArrayList<Integer> value = new ArrayList<>();
70         value = storage.get("original");
71         value.add(idPost);
72         storage.put("original", value);
73     }
74
75     /** Adds comment post ID to the account's storage of posts.
76      * @param idPost ID of post to be added to account comment storage
77      */
78     public void addToAccountStorageComment(int idPost){
79         ArrayList<Integer> value = new ArrayList<>();
80         value = storage.get("comments");
81         value.add(idPost);
82         storage.put("comments", value);
83     }
84
85     /** Adds endorsement post ID to the account's storage of posts.
86      * @param idPost ID of post to be added to account endorsement storage
87      */
88     public void addToAccountStorageEndors(int idPost){
89         ArrayList<Integer> value = new ArrayList<>();
90         value = storage.get("endorsements");
91         value.add(idPost);
92         storage.put("endorsements", value);
93     }
94
95     // Constructors
96
97     /** Creates account with a specified handle.
98      * @param handle handle of account
99      * @param id ID of account
100     */
101     public Account(String handle, int id){
102         this.handle = handle;
103         this.id = id;
104         description="";
105         storage = new HashMap<String, ArrayList<Integer>>();
106         /*
107          * original - posts that account made

```

```

108     * comments - comments that account made
109     * endorsements - account made
110     * ArrayList contains post IDs
111     */
112     storage.put("original", new ArrayList<Integer>());
113     storage.put("comments", new ArrayList<Integer>());
114     storage.put("endorsements", new ArrayList<Integer>());
115 }
116
117 /** Creates account with a specified handle and description.
118  * @param handle handle of account
119  * @param id ID of account
120  * @param description description of account
121  */
122 public Account(String handle,int id, String description){
123     this.handle = handle;
124     this.id = id;
125     this.description = description;
126     storage = new HashMap<String, ArrayList<Integer>>();
127     /*
128     * original - posts that account made
129     * comments - comments that account made
130     * endorsements - account made
131     * ArrayList contains post IDs
132     */
133     storage.put("original", new ArrayList<Integer>());
134     storage.put("comments", new ArrayList<Integer>());
135     storage.put("endorsements", new ArrayList<Integer>());
136 }
137 }

```

## 2 Post.java

```

1 package socialmedia;
2 import java.util.*;
3 import java.io.Serializable;
4
5 /**
6  * Represents a post.
7  * @author Ashley Card
8  * @author Maryia Fralova
9  * @version 1.0
10 */
11 public class Post implements Serializable{
12
13     // Attributes
14     private int numIdentifier;
15     private String message;
16     private int pointerToOriginal;
17     private HashMap<String, ArrayList<Integer>> storage ;
18
19     // Getter methods
20
21     /** Gets post's ID.

```

```

22     * @return numIdentifier ID of post
23     */
24     public int getNumIdentifier(){
25         return numIdentifier;
26     }
27
28     /** Gets post's message.
29     * @return message of post
30     */
31     public String getMessage(){
32         return message;
33     }
34
35     /** Gets post's parent post.
36     * @return pointerToOriginal ID of post parent
37     */
38     public int getPointerToOriginal(){
39         return pointerToOriginal;
40     }
41
42     /** Gets post's storage with comments and endorsements it received.
43     * @return storage of post comment and endorsement IDs
44     */
45     public HashMap<String, ArrayList<Integer>> getPostStorage(){
46         return storage;
47     }
48
49     // Setter methods
50
51     /** Adds comment post to the storage of the post being endorsed.
52     * @param idPost id of post being added to comment storage
53     */
54     public void addToPostStorageComment(int idPost){
55         ArrayList<Integer> value = new ArrayList<>();
56         value = storage.get("comments");
57         value.add(idPost);
58         storage.put("comments", value);
59     }
60
61     /** Adds endorsement post to the storage of the post being endorsed.
62     * @param idPost id of post being added to endorsement storage
63     */
64     public void addToPostStorageEndors(int idPost){
65         ArrayList<Integer> value = new ArrayList<>();
66         value = storage.get("endorsements");
67         value.add(idPost);
68         storage.put("endorsements", value);
69     }
70
71     /** Sets post's message.
72     * @param message post message
73     */
74     public void setMessage(String message){
75         this.message = message;
76     }

```

```

77
78  /** Creates an original post.
79  * @param numIdentifier ID of post
80  * @param message post message
81  */
82  public Post(int numIdentifier,String message){
83      this.numIdentifier = numIdentifier;
84      this.message = message;
85      storage = new HashMap<String, ArrayList<Integer>>();
86      pointerToOriginal = -1;
87      // endorsements - the no. endorsements the post has
88      // comments - the no. comments the post has
89      // ArrayList contains the postIDs of these
90      storage.put("comments", new ArrayList<Integer>());
91      storage.put("endorsements", new ArrayList<Integer>());
92  }
93
94  /** Creates a comment post.
95  * @param numIdentifier ID of post
96  * @param message post message
97  * @param pointerToOriginal ID of parent post
98  */
99  public Post(int numIdentifier,String message,int pointerToOriginal){
100      this.numIdentifier = numIdentifier;
101      this.message = message;
102      this.pointerToOriginal = pointerToOriginal;
103      storage = new HashMap<String, ArrayList<Integer>>();
104      /* endorsements - the no. endorsements the post has
105      comments - the no. comments the post has
106      */
107      storage.put("comments", new ArrayList<Integer>());
108      storage.put("endorsements", new ArrayList<Integer>());
109  }
110
111  /** Creates an endorsement post.
112  * @param numIdentifier ID of post
113  * @param pointerToOriginal ID of parent post
114  */
115  public Post(int numIdentifier,int pointerToOriginal){
116      this.numIdentifier = numIdentifier;
117      this.pointerToOriginal = pointerToOriginal;
118      message = new String();
119  }
120
121  /** Creates an empty post.
122  * @param numIdentifier ID of post
123  */
124  public Post(int numIdentifier){
125      this.numIdentifier = numIdentifier;
126      message = "The original content was removed from the system and is no longer available.";
127  }
128  }

```

### 3 SocialMedia.java

```
1 package socialmedia;
2
3 import java.util.*;
4 import java.io.*;
5
6 /**
7  * SocialMedia is a compiling, fully-functioning implementor of
8  * the SocialMediaPlatform interface.
9  * @author Ashley Card
10  * @author Maryia Fralova
11  * @version 1.0
12  */
13 public class SocialMedia implements SocialMediaPlatform {
14
15     // Attributes
16     private ArrayList<Account> listOfAccounts = new ArrayList<>();
17     private ArrayList<Post> listOfPosts = new ArrayList<>();
18     private ArrayList<Post> listOfEmptyPosts = new ArrayList<>();
19     private int idAccount = 0;
20     private int idPost = 0;
21
22     // Getter methods
23
24     /** Gets the array of existing accounts.
25      * @return listOfAccounts ArrayList of all account IDs
26      */
27     public ArrayList<Account> getAccounts() {
28         return listOfAccounts;
29     }
30
31     /** Gets the array of existing posts.
32      * @return listOfPosts ArrayList of all post IDs
33      */
34     public ArrayList<Post> getPosts() {
35         return listOfPosts;
36     }
37
38     /** Gets the array of existing empty posts.
39      * @return listOfEmptyPosts ArrayList of empty post IDs
40      */
41     public ArrayList<Post> getEmptyPosts() {
42         return listOfEmptyPosts;
43     }
44
45     /** Gets account's ID.
46      * @return idAccount ID of account
47      */
48     public int getIDAccount() {
49         return idAccount;
50     }
51
52     /** Gets posts's ID.
```

```

53     * @return idPost ID of post
54     */
55     public int getIDPost() {
56         return idPost;
57     }
58
59     // Methods
60
61     @Override
62     public int createAccount(String handle) throws IllegalHandleException, InvalidHandleException {
63
64         int numAccountsInitially = getNumberOfAccounts();
65         // Check if the handle already exists
66         for (Account i : listOfAccounts) {
67             if (i.getHandle().equals(handle)) {
68                 throw new IllegalHandleException();
69             }
70         }
71
72         // Check if the handle is valid
73         if (handle.isEmpty() || handle.length() > 30 || handle.matches(".*\\s.*")){
74             throw new InvalidHandleException();
75         }
76
77         // Create a new account object
78         Account acc = new Account(handle, idAccount);
79         listOfAccounts.add(acc);
80
81         // Asserting if account was added
82         assert ((numAccountsInitially + 1) == getNumberOfAccounts()): "The account was added incorrectly";
83
84         return idAccount++;
85     }
86
87     @Override
88     public int createAccount(String handle, String description) throws IllegalHandleException,
89         InvalidHandleException {
90
91         int numAccountsInitially = getNumberOfAccounts();
92         // Check if the handle already exists
93         for (Account i : listOfAccounts) {
94             if (i.getHandle().equals(handle)) {
95                 throw new IllegalHandleException();
96             }
97         }
98
99         // Check if the handle is valid
100        if (handle.isEmpty() || handle.length() > 30 || handle.matches(".*\\s.*")){
101            throw new InvalidHandleException();
102        }
103
104        // Create a new account object
105        Account acc = new Account(handle, idAccount, description);
106        listOfAccounts.add(acc);

```

```

107     // Asserting if account was added
108     assert ((numAccountsInitially + 1) == getNumberOfAccounts()): "The account was added incorrectly";
109
110     return idAccount++;
111 }
112
113 @Override
114 public void removeAccount(int id) throws AccountIDNotRecognisedException {
115
116     // Finds the account with the given id
117     int counter = 0;
118     int numAccountsInitially = getNumberOfAccounts();
119     boolean isThere = false;
120     for (Account acc : listOfAccounts) {
121         if (acc.getID() == id) {
122
123             // Gets the accounts post storage
124             HashMap<String, ArrayList<Integer>> storage = new HashMap<>();
125             storage = acc.getAccountStorage();
126             ArrayList<Integer> origposts = new ArrayList<Integer>();
127             ArrayList<Integer> commposts = new ArrayList<Integer>();
128             ArrayList<Integer> endorposts = new ArrayList<Integer>();
129             origposts = storage.get("original");
130             commposts = storage.get("comments");
131             endorposts = storage.get("endorsements");
132             ArrayList<Integer> listOfPostsToDelete = new ArrayList<>();
133
134             //Iterates over all of its posts
135             for (int post : origposts) {
136                 listOfPostsToDelete.add(post);
137             }
138
139             for (int post : commposts) {
140                 listOfPostsToDelete.add(post);
141             }
142
143             for (int post : endorposts) {
144                 listOfPostsToDelete.add(post);
145             }
146
147             // Deletes each post from everywhere
148             for (int i : listOfPostsToDelete) {
149                 try {
150                     deletePost(i);
151                 }
152                 catch (PostIDNotRecognisedException e){}
153             }
154
155             listOfAccounts.remove(counter);
156             isThere = true;
157             break;
158         }
159         counter++;
160     }
161 }

```



```

162 // Asserting account was removed
163 assert ((numAccountsInitially - 1) == getNumberOfAccounts()): "The account was removed incorrectly";
164
165 // If the id doesn't exist
166 if (!isThere){
167     throw new AccountIDNotRecognisedException();
168 }
169 }
170
171 @Override
172 public void removeAccount(String handle) throws HandleNotRecognisedException {
173
174     // Finds the account with the given handle
175     int counter = 0;
176     int numAccountsInitially = getNumberOfAccounts();
177     boolean isThere = false;
178     for (Account acc : listOfAccounts) {
179         if (acc.getHandle() == handle) {
180
181             // Gets the accounts post storage
182             HashMap<String, ArrayList<Integer>> storage = new HashMap<>();
183             storage = acc.getAccountStorage();
184             ArrayList<Integer> origposts = new ArrayList<Integer>();
185             ArrayList<Integer> commposts = new ArrayList<Integer>();
186             ArrayList<Integer> endorposts = new ArrayList<Integer>();
187             origposts = storage.get("original");
188             commposts = storage.get("comments");
189             endorposts = storage.get("endorsements");
190             ArrayList<Integer> listOfPostsToDelete = new ArrayList<>();
191
192             //Iterates over all of its posts
193             for (int post : origposts) {
194                 listOfPostsToDelete.add(post);
195             }
196
197             for (int post : commposts) {
198                 listOfPostsToDelete.add(post);
199             }
200
201             for (int post : endorposts) {
202                 listOfPostsToDelete.add(post);
203             }
204
205             // Deletes each post from everywhere
206             for (int i : listOfPostsToDelete) {
207                 try {
208                     deletePost(i);
209                 }
210                 catch (PostIDNotRecognisedException e){}
211             }
212
213             listOfAccounts.remove(counter);
214             isThere = true;
215             break;
216         }

```

```

217         counter++;
218     }
219
220     // Asserting account was removed
221     assert ((numAccountsInitially - 1) == getNumberOfAccounts()): "The account was removed incorrectly";
222
223     // If the handle doesn't exist
224     if (!isThere){
225         throw new HandleNotRecognisedException();
226     }
227 }
228
229 @Override
230 public void changeAccountHandle(String oldHandle, String newHandle)
231     throws HandleNotRecognisedException, IllegalHandleException, InvalidHandleException {
232
233     // Check if newHandle already exists
234     for (Account i : listOfAccounts) {
235         if (i.getHandle().equals(newHandle)) {
236             throw new IllegalHandleException();
237         }
238     }
239
240     // Check if new handle is valid
241     if (newHandle.isEmpty() || newHandle.length() > 30 || newHandle.matches(".*\\s.*")){
242         throw new InvalidHandleException();
243     }
244
245     // Find the account that matches oldHandle
246     boolean isThere = false;
247     for (Account i : listOfAccounts) {
248         if (i.getHandle().equals(oldHandle)){
249             i.setHandle(newHandle);
250             isThere = true;
251
252             // Asserting the handle was changed
253             assert (newHandle == i.getHandle()): "The handle was changed incorrectly";
254             break;
255         }
256     }
257
258     // If the oldHandle doesn't exist
259     if (!isThere){
260         throw new HandleNotRecognisedException();
261     }
262 }
263
264 @Override
265 public void updateAccountDescription(String handle, String description) throws
266     HandleNotRecognisedException {
267
268     // Find the account with the given handle
269     boolean isThere = false;
270     for (Account i : listOfAccounts) {
271         if (i.getHandle().equals(handle)){

```

```

271         i.setDescription(description);
272         isThere = true;
273
274         // Asserting the description was changed
275         assert (description == i.getDescription()): "The description was changed incorrectly";
276         break;
277     }
278 }
279
280 // If the handle doesn't exist
281 if (!isThere){
282     throw new HandleNotRecognisedException();
283 }
284 }
285
286 @Override
287 public String showAccount(String handle) throws HandleNotRecognisedException {
288
289     // Find the account with the given handle
290     boolean isThere = false;
291     int index=0;
292     for (Account i : listOfAccounts) {
293         if (i.getHandle().equals(handle)){
294             isThere = true;
295             break;
296         }
297         index++;
298     }
299
300     // If the handle doesn't exist
301     if (!isThere){
302         throw new HandleNotRecognisedException();
303     }
304
305     // Getting the posts made by the account
306     int sumOfEndors = 0;
307     HashMap<String, ArrayList<Integer>> storageOfPostsFromAccount =
308         listOfAccounts.get(index).getAccountStorage();
309
310     // Iterate over original
311     for (int eachID : storageOfPostsFromAccount.get("original")){
312         for (Post i : listOfPosts) {
313             if (i.getNumIdentifier() == eachID) {
314                 HashMap<String, ArrayList<Integer>> storageOfPostsFromPost = i.getPostStorage();
315                 sumOfEndors+=storageOfPostsFromPost.get("endorsements").size();
316             }
317         }
318     }
319
320     // Iterate over comments
321     for (int eachID : storageOfPostsFromAccount.get("comments")){
322         for (Post i : listOfPosts) {
323             if (i.getNumIdentifier() == eachID) {
324                 HashMap<String, ArrayList<Integer>> storageOfPostsFromPost = i.getPostStorage();
325                 sumOfEndors+=storageOfPostsFromPost.get("endorsements").size();

```

```

325     }
326 }
327 }
328
329 // Find each post that the account's made and count its endorsements
330 int sumOfPosts =
    storageOfPostsFromAccount.get("original").size()+storageOfPostsFromAccount.get("comments").size()+storageOfP
331 String toReturn = String.format("ID: %d\n"+
    "Handle: %s\n"+
    "Description: %s\n"+
    "Post count: %d\n"+
    "Endorse count: %d", listOfAccounts.get(index).getID(),
    listOfAccounts.get(index).getHandle(),
    listOfAccounts.get(index).getDescription(), sumOfPosts, sumOfEndors);
336 return toReturn;
337 }
338
339 @Override
340 public int createPost(String handle, String message) throws HandleNotRecognisedException,
    InvalidPostException {
341
342     int numPostsInitially = listOfPosts.size();
343
344     // Check if the message is valid
345     if (message.isEmpty() || message.length() > 100){
346         throw new InvalidPostException();
347     }
348
349     // Find the account with the given handle and create the post
350     boolean isThere = false;
351     int counter = 0;
352     for (Account i : listOfAccounts) {
353         if (i.getHandle() == handle) {
354             listOfAccounts.get(counter).addToAccountStorageOriginal(idPost);
355             Post post = new Post(idPost, message);
356             listOfPosts.add(post);
357             isThere = true;
358             break;
359         }
360         counter++;
361     }
362
363     // Asserting post was added
364     assert ((numPostsInitially + 1) == listOfPosts.size()): "The post was added incorrectly";
365
366     // If the handle doesn't exist
367     if (!isThere){
368         throw new HandleNotRecognisedException();
369     }
370     return idPost++;
371 }
372
373 @Override
374 public int endorsePost(String handle, int id) throws HandleNotRecognisedException,
    PostIDNotRecognisedException, NotActionablePostException {

```

```

375
376     int numPostsInitially = listOfPosts.size();
377     // Find the post with the given id
378     String message="";
379     String newMessage="";
380     boolean postIsThere = false;
381     int postCounter = 0;
382
383     // Finds the post that is endorsed
384     for (Post i : listOfPosts) {
385         if (i.getNumIdentifier() == id) {
386             postIsThere = true;
387
388             // Checks if it is an endorsement post itself
389             if (listOfPosts.get(postCounter).getPostStorage() == null){
390                 throw new NotActionablePostException();
391             }
392
393             // Adds the id of created post to the endorsed post storage
394             listOfPosts.get(postCounter).addToPostStorageEndors(idPost);
395             message = i.getMessage();
396             break;
397         }
398         postCounter++;
399     }
400
401     // If the post doesn't exist
402     if (!postIsThere){
403         throw new PostIDNotRecognisedException();
404     }
405
406     // Find account that was endorsed and edit endorsed message
407     for(Account acc : listOfAccounts){
408         HashMap<String, ArrayList<Integer>> storage = new HashMap<>();
409         storage = acc.getAccountStorage();
410
411         // Iterate over original
412         boolean found = false;
413         int counter = 0;
414         for (int orig : storage.get("original")){
415             if (orig == id){
416                 found = true;
417                 break;
418             }
419             counter++;
420         }
421
422         // Iterate over comments
423         if (!found){
424             counter = 0;
425             for (int comm : storage.get("comments")){
426                 if (comm == id){
427                     found = true;
428                     break;
429                 }

```

```

430         counter++;
431     }
432 }
433
434 if (found){
435     String endorsedHandle = acc.getHandle();
436     //Edit message
437     newMessage = "EP@" + endorsedHandle + ": " + message;
438     break;
439 }
440 }
441
442 // Find the account with the given handle
443 boolean isThere = false;
444 int counter = 0;
445 for (Account i : listOfAccounts) {
446     if (i.getHandle() == handle) {
447         listOfAccounts.get(counter).addToAccountStorageEndors(idPost);
448         Post post = new Post(idPost, id);
449         post.setMessage(newMessage);
450         listOfPosts.add(post);
451         isThere = true;
452         break;
453     }
454     counter++;
455 }
456
457 // Asserting post was added
458 assert ((numPostsInitially + 1) == listOfPosts.size()): "The post was added incorrectly";
459
460 // If the handle doesn't exist
461 if (!isThere){
462     throw new HandleNotRecognisedException();
463 }
464 return idPost++;
465 }
466
467 @Override
468 public int commentPost(String handle, int id, String message) throws HandleNotRecognisedException,
469     PostIDNotRecognisedException, NotActionablePostException, InvalidPostException {
470
471     int numPostsInitially = listOfPosts.size();
472
473     // Check if the message is valid
474     if (message.isEmpty() || message.length() > 100){
475         throw new InvalidPostException();
476     }
477
478     // Find the post with the given id
479     boolean postIsThere = false;
480     int postCounter = 0;
481     for (Post i : listOfPosts) {
482         if (i.getNumIdentifier() == id) {
483             postIsThere = true;
484             if (listOfPosts.get(postCounter).getPostStorage() == null){

```

```

485         throw new NotActionablePostException();
486     }
487     listOfPosts.get(postCounter).addToPostStorageComment(idPost);
488     break;
489 }
490 postCounter++;
491 }
492
493 // If the post doesn't exist
494 if (!postIsThere){
495     throw new PostIDNotRecognisedException();
496 }
497
498 // Find the account with the given handle and create the comment
499 boolean isThere = false;
500 int counter = 0;
501 for (Account i : listOfAccounts) {
502     if (i.getHandle() == handle) {
503         listOfAccounts.get(counter).addToAccountStorageComment(idPost);
504         Post post = new Post(idPost, message, id);
505         listOfPosts.add(post);
506         isThere = true;
507         break;
508     }
509     counter++;
510 }
511
512 // Asserting post was added
513 assert ((numPostsInitially + 1) == listOfPosts.size()): "The post was added incorrectly";
514
515 // If the handle doesn't exist
516 if (!isThere){
517     throw new HandleNotRecognisedException();
518 }
519 return idPost++;
520 }
521
522 @Override
523 public void deletePost(int id) throws PostIDNotRecognisedException {
524
525     // Find the post with the given id
526     boolean postIsThere = false;
527     int postCounter = 0;
528     int numPostsInitially = listOfPosts.size();
529     for (Post i : listOfPosts) {
530         List<Integer> counters = new ArrayList<>();
531         if (i.getNumIdentifier() == id) {
532             postIsThere = true;
533             Post post = new Post(id);
534             listOfEmptyPosts.add(post);
535
536             // Checks type of post
537             boolean isOriginal = false;
538             if (i.getPointerToOriginal() == -1) {
539                 isOriginal = true;

```

```

540     }
541     // If it is a comment or endorsement, removes from parent post
542     if (!isOriginal) {
543         for (Post postmain : listOfPosts) {
544             if (postmain.getNumIdentifier() == i.getPointerToOriginal()) {
545                 HashMap<String, ArrayList<Integer>> postMainStorage = postmain.getPostStorage();
546                 ArrayList<Integer> idList = new ArrayList<>();
547                 idList.add(id);
548                 postMainStorage.get("comments").removeAll(idList);
549                 postMainStorage.get("endorsements").removeAll(idList);
550             }
551         }
552     }
553
554     // Checking that it isn't an endorsement
555     if (listOfPosts.get(postCounter).getPostStorage() != null){
556
557         // Remove all its endorsements
558         HashMap<String, ArrayList<Integer>> storagePosts =
559             listOfPosts.get(postCounter).getPostStorage();
560         ArrayList<Integer> value = new ArrayList<Integer>();
561         value = storagePosts.get("endorsements");
562
563         // Copy of the storage with postIDs of endorsements to delete from Account storage
564         ArrayList<Integer> storageOfEndors = new ArrayList<Integer>();
565         for (int val : value){
566             storageOfEndors.add(val);
567         }
568
569         // Deleting all the endorsements from listOfPosts
570         value.clear();
571     }
572
573     // Remove from Account's storage
574     for(Account acc : listOfAccounts){
575         HashMap<String, ArrayList<Integer>> storage = new HashMap<>();
576         storage = acc.getAccountStorage();
577
578         // Iterate over original
579         boolean deleted = false;
580         int counter = 0;
581         for (int orig : storage.get("original")){
582             if (orig == id){
583                 storage.get("original").remove(counter);
584                 deleted = true;
585                 break;
586             }
587             counter++;
588         }
589
590         // Iterate over comments
591         if (!deleted){
592             counter = 0;
593             for (int comm : storage.get("comments")){

```



```

594         if (comm == id){
595             storage.get("comments").remove(counter);
596             deleted = true;
597             break;
598         }
599         counter++;
600     }
601 }
602
603 // Iterate over comments
604 if (!deleted){
605     counter = 0;
606     for (int endors : storage.get("endorsements")){
607         if (endors == id){
608             storage.get("endorsements").remove(counter);
609             break;
610         }
611         counter++;
612     }
613 }
614 // Checking that it isn't an endorsement
615 if (listOfPosts.get(postCounter).getPostStorage() != null){
616
617     // Removes all its endorsements
618     HashMap<String, ArrayList<Integer>> storagePosts =
619         listOfPosts.get(postCounter).getPostStorage();
620     ArrayList<Integer> value = new ArrayList<Integer>();
621     value = storagePosts.get("endorsements");
622     ArrayList<Integer> storageOfEndors = new ArrayList<Integer>();
623
624     // Iterating over endorsements
625     for (int endor : storage.get("endorsements")){
626         for (int idEndors : storageOfEndors){
627             if (endor == idEndors){
628                 counters.add(endor);
629                 break;
630             }
631         }
632     }
633     storage.get("endorsements").removeAll(counters);
634 }
635 }
636
637 // Removes post from listOfPosts
638 listOfPosts.remove(listOfPosts.get(postCounter));
639
640 // also need to remove endorsemetns from ListOfPosts
641 int counter = 0;
642 ArrayList<Post> indexes = new ArrayList<>();
643
644 // Finds the indexes of the endorsements to be removed
645 for (Post a : listOfPosts){
646     for (int idInCounters : counters){
647         if (a.getNumIdentifier() == idInCounters){

```

```

648         indexes.add(a);
649         break;
650     }
651 }
652     counter++;
653 }
654     listOfPosts.removeAll(indexes);
655     break;
656 }
657     postCounter++;
658 }
659
660 // Asserting post was removed
661 assert ((numPostsInitially - 1) == listOfPosts.size()): "The post was removed incorrectly";
662
663 // If the post doesn't exist
664 if (!postIsThere){
665     throw new PostIDNotRecognisedException();
666 }
667 }
668
669 @Override
670 public String showIndividualPost(int id) throws PostIDNotRecognisedException {
671
672     // Finding the account with the post
673     boolean postIsThere = false;
674     String handle = "";
675     for (Account acc : listOfAccounts){
676         HashMap<String, ArrayList<Integer>> storage = new HashMap<>();
677         storage = acc.getAccountStorage();
678
679         // Search the original
680         for (int orig : storage.get("original")){
681             if (orig == id){
682                 handle = acc.getHandle();
683                 postIsThere = true;
684                 break;
685             }
686         }
687
688         // Search the comments
689         for (int comm : storage.get("comments")){
690             if (comm == id){
691                 handle = acc.getHandle();
692                 postIsThere = true;
693                 break;
694             }
695         }
696
697         // Search the endorsements
698         for (int endor : storage.get("endorsements")){
699             if (endor == id){
700                 handle = acc.getHandle();
701                 postIsThere = true;
702                 break;

```

```

703     }
704 }
705 }
706
707 // If the post doesn't exist
708 if (!postIsThere){
709     throw new PostIDNotRecognisedException();
710 }
711
712 // Finds the index of the post in listOfPosts
713 int indexNeededPost=0;
714 for (Post post : listOfPosts){
715     if (post.getNumIdentifier() == id){
716         indexNeededPost = listOfPosts.indexOf(post);
717         break;
718     }
719 }
720
721 // Checking if it is an endorsement post
722 if (listOfPosts.get(indexNeededPost).getPostStorage() == null){
723     String message = listOfPosts.get(indexNeededPost).getMessage();
724     return String.format("ID: %d\n Account: %s\n No.endorsements: %d\nNo.comments: %d\n %s", id, handle,
725         0, 0, message);
726 }
727
728 // Gets the number of endorsements the post has received
729 ArrayList<Integer> postStorageEnd = new ArrayList<>();
730 postStorageEnd = listOfPosts.get(indexNeededPost).getPostStorage().get("endorsements");
731 int noEndor = postStorageEnd.size();
732
733 // Gets the number of comments the post has received
734 ArrayList<Integer> postStorageComm = new ArrayList<>();
735 postStorageComm = listOfPosts.get(indexNeededPost).getPostStorage().get("comments");
736 int noComm = postStorageComm.size();
737
738 // Gets the message of the post
739 String message = listOfPosts.get(indexNeededPost).getMessage();
740
741 return String.format("ID: %d\n Account: %s\n No.endorsements: %d\nNo.comments: %d\n %s", id, handle,
742     noEndor, noComm, message);
743 }
744
745 @Override
746 public StringBuilder showPostChildrenDetails(int id)
747     throws PostIDNotRecognisedException, NotActionablePostException {
748
749     boolean postIsThere = false;
750     StringBuilder str = new StringBuilder();
751
752     // Checking for post
753     for (Post initialPost : listOfPosts){
754         if (initialPost.getNumIdentifier() == id){
755
756             // Check if the post is an endorsement
757             if (initialPost.getPostStorage() == null){

```

```

756         throw new NotActionablePostException();
757     }
758     postIsThere = true;
759 }
760 }
761
762 // If the post doesn't exist
763 if (!postIsThere){
764     throw new PostIDNotRecognisedException();
765 }
766
767 // Calling helper function
768 str = showChildPostsHelper(id, 0);
769 str.delete(0,5);
770 return str;
771 }
772
773 /*
774  * Helper function for showPostChildrenDetails.
775  * Recursively displays children comments of a parent post.
776  */
777 private StringBuilder showChildPostsHelper(int id, int indentLevel) throws PostIDNotRecognisedException{
778     StringBuilder str2 = new StringBuilder();
779     boolean postIsThere2 = false;
780     String stickthing = "|"; // stick :)
781
782     for (Post initialPost : listOfPosts){
783         if (initialPost.getNumIdentifier() == id){
784             postIsThere2 = true;
785
786             str2.append((stickthing.indent(indentLevel)));
787             str2.append(("| > "+showIndividualPost(id).indent(indentLevel)));
788
789             // Gets children of initial post
790             ArrayList<Integer> initialPostStorageComm = new ArrayList<>();
791             initialPostStorageComm = initialPost.getPostStorage().get("comments");
792
793             // Loops through every comment in the posts storage
794             for (Integer childPostID : initialPostStorageComm){
795                 ArrayList<Integer> storage = new ArrayList<>();
796                 storage = initialPost.getPostStorage().get("comments");
797
798                 // If the child post has it's own children, calls this function on itself
799                 if (!storage.isEmpty()){
800                     str2.append(showChildPostsHelper(childPostID, indentLevel + 4));
801                     // Otherwise, display the post
802                 }else{
803                     str2.append((stickthing.indent(indentLevel)));
804                     str2.append(("| > "+showIndividualPost(childPostID).indent(indentLevel)));
805                 }
806             }
807         }
808     }
809
810     // Checks if the post exists

```

```

811     if (!postIsThere2){
812         System.out.println(str2);
813         throw new PostIDNotRecognisedException();
814     }
815     return str2;
816 }
817
818 @Override
819 public int getNumberOfAccounts() {
820     return listOfAccounts.size();
821 }
822
823 @Override
824 public int getTotalOriginalPosts() {
825
826     int sumPosts = 0;
827
828     // Increments a counter if post is an original
829     for (Post post : listOfPosts){
830         if (post.getPointerToOriginal() == -1){
831             sumPosts++;
832         }
833     }
834     return sumPosts;
835 }
836
837 @Override
838 public int getTotalEndorsmentPosts() {
839
840     int sumPosts = 0;
841
842     // Increments a counter if post is an endorsement
843     for (Post post : listOfPosts){
844         if (post.getPostStorage() == null){
845             sumPosts++;
846         }
847     }
848     return sumPosts;
849 }
850
851 @Override
852 public int getTotalCommentPosts() {
853
854     int sumPosts = 0;
855
856     // Increments a counter if post is a comment
857     for (Post post : listOfPosts){
858         if (post.getPostStorage() != null && post.getPointerToOriginal() != -1){
859             sumPosts++;
860         }
861     }
862     return sumPosts;
863 }
864
865 @Override

```

```

866 public int getMostEndorsedPost() {
867
868     int max = 0;
869     int idPost = 0;
870
871     // Iterating over every post
872     for (Post post : listOfPosts){
873         HashMap<String, ArrayList<Integer>> storagePosts = post.getPostStorage();
874         ArrayList<Integer> storageEndors = new ArrayList<Integer>();
875
876         // Checks for endorsement storage
877         if (post.getPostStorage() != null){
878             storageEndors = storagePosts.get("endorsements");
879
880             // Changes max value to current value if greater
881             if (storageEndors.size() > max){
882                 max = storageEndors.size();
883                 idPost = post.getNumIdentifier();
884             }
885         }
886     }
887     return idPost;
888 }
889
890 @Override
891 public int getMostEndorsedAccount() {
892
893     int max = 0;
894     int idOfAccount = 0;
895     for (Account acc : listOfAccounts){
896
897         // Getting the posts made by the account
898         int sumOfEndors = 0;
899         HashMap<String, ArrayList<Integer>> storageOfPostsFromAccount = acc.getAccountStorage();
900
901         // Iterate over original
902         for (int eachID : storageOfPostsFromAccount.get("original")){
903             for (Post i : listOfPosts) {
904                 if (i.getNumIdentifier() == eachID) {
905                     HashMap<String, ArrayList<Integer>> storageOfPostsFromPost = i.getPostStorage();
906                     sumOfEndors+=storageOfPostsFromPost.get("endorsements").size();
907                 }
908             }
909         }
910
911         // Iterate over comments
912         for (int eachID : storageOfPostsFromAccount.get("comments")){
913             for (Post i : listOfPosts) {
914                 if (i.getNumIdentifier() == eachID) {
915                     HashMap<String, ArrayList<Integer>> storageOfPostsFromPost = i.getPostStorage();
916                     sumOfEndors+=storageOfPostsFromPost.get("endorsements").size();
917                 }
918             }
919         }
920     }

```

```

921         // Changes max value to current value if greater
922         if (sumOfEndors > max){
923             max = sumOfEndors;
924             idOfAccount = acc.getID();
925         }
926     }
927     return idOfAccount;
928 }
929
930 @Override
931 public void erasePlatform() {
932     listOfAccounts.clear();
933     listOfPosts.clear();
934     listOfEmptyPosts.clear();
935     idAccount = 0;
936     idPost = 0;
937 }
938
939 @Override
940 public void savePlatform(String filename) throws IOException {
941
942     // Saves the platform global variables to a file
943     try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(filename))) {
944         out.writeObject(listOfAccounts);
945         out.writeObject(listOfPosts);
946         out.writeObject(listOfEmptyPosts);
947         out.writeObject(idAccount);
948         out.writeObject(idPost);
949     } catch (IOException e){
950         throw new IOException();
951     }
952 }
953
954 @Override
955 public void loadPlatform(String filename) throws IOException, ClassNotFoundException {
956
957     // Gets each global variable from the saved file
958     try (ObjectInputStream in = new ObjectInputStream(new FileInputStream(filename))) {
959         Object obj = in.readObject();
960         if (obj instanceof ArrayList)
961             listOfAccounts = (ArrayList<Account>) obj;
962         obj = in.readObject();
963         if (obj instanceof ArrayList)
964             listOfPosts = (ArrayList<Post>) obj;
965         obj = in.readObject();
966         if (obj instanceof ArrayList)
967             listOfEmptyPosts = (ArrayList<Post>) obj;
968         obj = in.readObject();
969         if (obj instanceof ArrayList)
970             idAccount = (Integer) obj;
971         obj = in.readObject();
972         if (obj instanceof ArrayList)
973             idPost = (Integer) obj;
974     } catch (ClassNotFoundException e){
975         throw new ClassNotFoundException();

```

```
976     } catch (IOException e){  
977         throw new IOException();  
978     }  
979 }  
980 }
```