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();
```

```
	}

	@Override
	public int createPost(String handle, String message) throws HandleNotRecognisedException, InvalidPostException {
		// Checks the message is inside the character limit
		if (message.length() <= 100 && message != "" && message != null) {
			for (int pos = 0; pos < accountsList.size(); pos++) {
				// Finds account with the handle
				if (accountsList.get(pos).getHandle() == handle) {
					return accountsList.get(pos).createOriginalPost(message);
				}
			}
			throw new HandleNotRecognisedException();
		} else {
			throw new InvalidPostException();
		}
	}

	@Override
	public int endorsePost(String handle, int id)
			throws HandleNotRecognisedException, PostIDNotRecognisedException, NotActionablePostException {
		boolean handleExists = false;
		boolean notActionable = false;
		for (int pos = 0; pos < accountsList.size(); pos++) {
			// Finds account with the handle
			if (accountsList.get(pos).getHandle() == handle) {
				handleExists = true;
				int endorseID = -1;
				for (Account acc : accountsList) {
					// Adds endorsement ID to post when the ID of the post being endorsed is found in an account
					int foundPostIDPos = acc.searchPost(id);
					if (foundPostIDPos != -1) {
						endorseID = accountsList.get(pos).createEndorsement("EP@" + acc.getHandle() + ": "
								+ acc.getOriginalPosts().get(foundPostIDPos).getMessage(), id);
						acc.addEndorseIDtoPostAt(foundPostIDPos, endorseID);
						break;
					}
					int foundCommentIDPos = acc.searchComment(id);
					if (foundCommentIDPos != -1) {
						endorseID = accountsList.get(pos).createEndorsement(
								"EP@" + acc.getHandle() + ": " + acc.getComments().get(foundCommentIDPos).getMessage(),
								id);
						acc.addEndorseIDtoCommentAt(foundCommentIDPos, endorseID);
						break;
					}
					int foundEndorseIDPos = acc.searchEndorsement(id);
					if (foundEndorseIDPos != -1) {
						notActionable = true;
						break;
					}
				}
				if (endorseID != -1) {
					return endorseID;
				}
			}
		}
		if (!handleExists) {
			throw new HandleNotRecognisedException();
		} else if (notActionable) {
			throw new NotActionablePostException();
		}
		throw new PostIDNotRecognisedException();
	}

	@Override
	public int commentPost(String handle, int id, String message) throws HandleNotRecognisedException,
			PostIDNotRecognisedException, NotActionablePostException, InvalidPostException {
		// Checks the message is inside the character limit
		if (message.length() <= 100 && message != "" && message != null) {
			boolean handleExists = false;
			boolean postIDExists = false;
			boolean notActionable = false;
			for (int pos = 0; pos < accountsList.size(); pos++) {
				if (accountsList.get(pos).getHandle() == handle) {
					handleExists = true;
					int commentID = -1;
					for (Account acc : accountsList) {
						// Adds comment ID to post when the ID of the post being commented is found in an account
						int foundPostIDPos = acc.searchPost(id);
						if (foundPostIDPos != -1) {
							postIDExists = true;
							commentID = accountsList.get(pos).createComment(message, id);
							acc.addCommentIDtoPostAt(foundPostIDPos, commentID);
							break;
						}
						int foundCommentIDPos = acc.searchComment(id);
						if (foundCommentIDPos != -1) {
							postIDExists = true;
							commentID = accountsList.get(pos).createComment(message, id);
							acc.addCommentIDtoCommentAt(foundCommentIDPos, commentID);
							break;
```

```
                                        }
                                        int foundEndorseIDPos = acc.searchEndorsement(id);
                                        if (foundEndorseIDPos != -1) {
                                                postIDExists = true;
                                                notActionable = true;
                                                break;
                                        }
                                }
                                if (commentID != -1) {
                                        return commentID;
                                }
                        }
                }
                if (!handleExists) {
                        throw new HandleNotRecognisedException();
                } else if (notActionable) {
                        throw new NotActionablePostException();
                } else if (!postIDExists) {
                        throw new PostIDNotRecognisedException();
                }
        }
        throw new InvalidPostException();
    }

    @Override
    public void deletePost(int id) throws PostIDNotRecognisedException {
        int postPos = -1;
        int endorsePos = -1;
        int commentPos = -1;
        int accPos = 0;
        while (accPos < accountsList.size()) {
                // Looks for post, endorsement or comment with the ID
                postPos = accountsList.get(accPos).searchPost(id);
                endorsePos = accountsList.get(accPos).searchEndorsement(id);
                commentPos = accountsList.get(accPos).searchComment(id);
                // Stops searching when object has been found
                if (postPos != -1 || endorsePos != -1 || commentPos != -1) {
                        break;
                }
                accPos++;
        }
        // Throws exception if the object doesn't exist
        if (postPos == -1 && endorsePos == -1 && commentPos == -1) {
                throw new PostIDNotRecognisedException();
        }
        // ID found in original posts
        if (postPos != -1) {
                Post oldPost = accountsList.get(accPos).getOriginalPosts().get(postPos);
                // Placeholder creation, doesn't create if there are no comments
                if (oldPost.getCommentsList().size() != 0) {
                        deletedPostsList.add(new DeletedPost(oldPost.getPostID(), oldPost.getCommentsList()));
                }
                ArrayList<Integer> oldEndorsements = oldPost.getEndorsementsList();
                // Looks through accounts again to find endorsements made on the post
                for (Account acc : accountsList) {
                        ArrayList<Endorsement> endorsements = new ArrayList<Endorsement>(acc.getEndorsements());
                        for (Endorsement endorse : endorsements) {
                                // Checks if the ID is in the list of endorsement IDs to be deleted
                                if (oldEndorsements.contains(endorse.getPostID())) {
                                        // Deletes endorsement object
                                        acc.getEndorsements().remove(endorse);
                                        // Removes from endorsements to be deleted list since it has been found
                                        oldEndorsements.remove((Integer) endorse.getPostID());
                                }
                        }
                        // Stops early when all endorsements have been found and deleted
                        if (oldEndorsements.size() == 0) {
                                break;
                        }
                }
                // Removes the post object
                accountsList.get(accPos).removePostAt(postPos);
        } else if (endorsePos != -1) { // ID found in endorsements
                Endorsement oldEndorsement = accountsList.get(accPos).getEndorsements().get(endorsePos);
                for (Account acc : accountsList) {
                        // Looks for post or comment with the post reference ID
                        int postPos2 = acc.searchPost(oldEndorsement.getPostReferenceID());
                        int commentPos2 = acc.searchComment(oldEndorsement.getPostReferenceID());
                        // Deletes endorsement ID from post
                        if (postPos2 != -1) {
                                acc.removeEndorseIDinPostAt(postPos2, oldEndorsement.getPostID());
                                break;
                        } else if (commentPos2 != -1) {
                                acc.removeEndorseIDinCommentAt(commentPos2, oldEndorsement.getPostID());
                                break;
                        }
                }
                // Removes the endorsement object
                accountsList.get(accPos).removeEndorsementAt(endorsePos);
        } else if (commentPos != -1) { // ID found in comments
                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                                                                                                     change++;
453                                                                                                     break;
454                                                                                             }
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, sop 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++) {
```

```
                ArrayList<Post> postsList = accountsList.get(pos).getOriginalPosts();
                // Looks through all original posts of the account
                for (int x = 0; x < postsList.size(); x++) {
                        if (postsList.get(x).getPostID() == id && !checkedElements.contains(id)) {
                                // Adds ID since it is being checked
                                checkedElements.add(id);
                                StringBuilder string = new StringBuilder();
                                String tab = "\t";
                                int changeableIndentationNumber = indentationNumber;
                                changeableIndentationNumber = indentationNumber;
                                if (changeableIndentationNumber != 0) {
                                        string.append("\n");
                                }
                                changeableIndentationNumber = indentationNumber;
                                while (changeableIndentationNumber > 1) {
                                        string.append(tab);
                                        changeableIndentationNumber -= 1;
                                }
                                string.append("ID:");
                                string.append(Integer.toString(id) + "\n");
                                changeableIndentationNumber = indentationNumber;
                                while (changeableIndentationNumber > 0) {
                                        string.append(tab);
                                        changeableIndentationNumber -= 1;
                                }
                                for (int a = 0; a < accountsList.size(); a++) {
                                        if (accountsList.get(a).getAccountID() == postsList.get(x).getAccountID()) {
                                                changeableIndentationNumber = indentationNumber;
                                                while (changeableIndentationNumber > 1) {
                                                        string.append(tab);
                                                        changeableIndentationNumber -= 1;
                                                }
                                                string.append("Account: " + accountsList.get(a).getHandle() + "\n");
                                        }
                                }
                                changeableIndentationNumber = indentationNumber;
                                while (changeableIndentationNumber > 0) {
                                        string.append(tab);
                                        changeableIndentationNumber -= 1;
                                }
                                string.append("No. endorsements: " + Integer.toString(postsList.get(x).getEndorsementsList().size())
                                                + " | No. comments: " + Integer.toString(postsList.get(x).getCommentsList().size()) + "\n");
                                changeableIndentationNumber = indentationNumber;
                                while (changeableIndentationNumber > 0) {
                                        string.append(tab);
                                        changeableIndentationNumber -= 1;
                                }
                                string.append(postsList.get(x).getMessage());

                                for (int y = 0; y < postsList.get(x).getCommentsList().size(); y++) {
                                        ArrayList<Integer> childElementsList = postsList.get(x).getCommentsList();
                                        for (int z = 0; z < childElementsList.size(); z++) {
                                                changeableIndentationNumber = indentationNumber;
                                                if (changeableIndentationNumber != 0) {
                                                        string.append("\n");
                                                }
                                                changeableIndentationNumber = indentationNumber + 1;
                                                while (changeableIndentationNumber > 1) {
                                                        string.append(tab);
                                                        changeableIndentationNumber -= 1;
                                                }
                                                changeableIndentationNumber = indentationNumber;
                                                if (changeableIndentationNumber != 0) {
                                                        string.append("|");
                                                }
                                                string.append(recursiveApproach(childElementsList.get(z)));
                                        }
                                }
                                indentationNumber -= 1;
                                return string;
                        }
                }
                ArrayList<Comment> commentsList = accountsList.get(pos).getComments();
                for (int x = 0; x < commentsList.size(); x++) {
                        if (commentsList.get(x).getPostID() == id && !checkedElements.contains(id)) {
                                checkedElements.add(id);
                                StringBuilder string = new StringBuilder();
                                String tab = "\t";
                                int changeableIndentationNumber = indentationNumber;
                                if (indentationNumber != 0) {
                                        changeableIndentationNumber = indentationNumber;
                                        if (changeableIndentationNumber != 0) {
                                                string.append("\n");
                                        }
                                        changeableIndentationNumber = indentationNumber;
                                        while (changeableIndentationNumber > 1) {
                                                string.append(tab);
                                                changeableIndentationNumber -= 1;
                                        }
                                        string.append("|" + tab + "> ID:");
                                        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;
```

```
                        }
                    }
                }
                ArrayList<DeletedComment> deleteCommentsListObj = deletedCommentsList;
                for (int x = 0; x < deleteCommentsListObj.size(); x++) {
                    if (deleteCommentsListObj.get(x).getPostID() == id && !checkedElements.contains(id)) {
                        checkedElements.add(id);
                        StringBuilder string = new StringBuilder();
                        String tab = "\t";
                        int changeableIndentationNumber = indentationNumber;
                        if (indentationNumber != 0) {
                            changeableIndentationNumber = indentationNumber;
                            if (changeableIndentationNumber != 0) {
                                string.append("\n");
                            }
                            changeableIndentationNumber = indentationNumber;
                            while (changeableIndentationNumber > 1) {
                                string.append(tab);
                                changeableIndentationNumber -= 1;
                            }
                            string.append("|" + tab + "> ID:");
                            string.append(Integer.toString(id) + "\n");
                            changeableIndentationNumber = indentationNumber;
                            while (changeableIndentationNumber > 1) {
                                string.append(tab);
                                changeableIndentationNumber -= 1;
                            }
                            changeableIndentationNumber = indentationNumber;
                            while (changeableIndentationNumber > 1) {
                                string.append(tab);
                                changeableIndentationNumber -= 1;
                            }
                            changeableIndentationNumber = indentationNumber;
                            if (changeableIndentationNumber == 1) {
                                string.append(tab);
                            }
                            string.append("Account: N/A \n");
                            changeableIndentationNumber = indentationNumber;
                            while (changeableIndentationNumber > 1) {
                                string.append(tab);
                                changeableIndentationNumber -= 1;
                            }
                            string.append(tab + "No. endorsements: N/A | No. comments: "
                                    + Integer.toString(deleteCommentsListObj.get(x).getCommentsList().size()) + "\n");
                            changeableIndentationNumber = indentationNumber;
                            while (changeableIndentationNumber > 1) {
                                string.append(tab);
                                changeableIndentationNumber -= 1;
                            }
                            string.append(tab + deleteCommentsListObj.get(x).getMessage());
                        }
                        if (indentationNumber == 0) {
                            changeableIndentationNumber = indentationNumber;
                            if (changeableIndentationNumber != 0) {
                                string.append("\n");
                            }
                            changeableIndentationNumber = indentationNumber;
                            while (changeableIndentationNumber > 1) {
                                string.append(tab);
                                changeableIndentationNumber -= 1;
                            }
                            string.append("ID:");
                            string.append(Integer.toString(id) + "\n");
                            changeableIndentationNumber = indentationNumber;
                            while (changeableIndentationNumber > 1) {
                                string.append(tab);
                                changeableIndentationNumber -= 1;
                            }
                            string.append("Account: N/A \n");
                            changeableIndentationNumber = indentationNumber;
                            while (changeableIndentationNumber > 1) {
                                string.append(tab);
                                changeableIndentationNumber -= 1;
                            }
                            string.append("No. endorsements: N/A | No. comments: "
                                    + Integer.toString(deleteCommentsListObj.get(x).getCommentsList().size()) + "\n");
                            changeableIndentationNumber = indentationNumber;
                            while (changeableIndentationNumber > 1) {
                                string.append(tab);
                                changeableIndentationNumber -= 1;
                            }
                            string.append(deleteCommentsListObj.get(x).getMessage());
                        }

                        for (int y = 0; y < deleteCommentsListObj.get(x).getCommentsList().size(); y++) {
                            ArrayList<Integer> childElementsList = deleteCommentsListObj.get(x).getCommentsList();
                            for (int z = 0; z < childElementsList.size(); z++) {
                                changeableIndentationNumber = indentationNumber;
                                if (changeableIndentationNumber != 0) {
                                    string.append("\n");
                                }
```

```
                                        changeableIndentationNumber = indentationNumber + 1;
                                        while (changeableIndentationNumber > 1) {
                                                string.append(tab);
                                                changeableIndentationNumber -= 1;
                                        }
                                        changeableIndentationNumber = indentationNumber;
                                        if (changeableIndentationNumber != 0) {
                                                string.append("|");
                                        }
                                        string.append(recursiveApproach(childElementsList.get(z)));
                                }
                        }
                        indentationNumber -= 1;
                        return string;
                }
        }
        ArrayList<DeletedPost> deletePostsListObj = deletedPostsList;
        for (int x = 0; x < deletePostsListObj.size(); x++) {
                if (deletePostsListObj.get(x).getPostID() == id && !checkedElements.contains(id)) {
                        checkedElements.add(id);
                        StringBuilder string = new StringBuilder();
                        String tab = "\t";
                        int changeableIndentationNumber = indentationNumber;
                        if (indentationNumber != 0) {
                                changeableIndentationNumber = indentationNumber;
                                if (changeableIndentationNumber != 0) {
                                        string.append("\n");
                                }
                                changeableIndentationNumber = indentationNumber;
                                while (changeableIndentationNumber > 1) {
                                        string.append(tab);
                                        changeableIndentationNumber -= 1;
                                }
                                string.append("|" + tab + "> ID:");
                                string.append(Integer.toString(id) + "\n");
                                changeableIndentationNumber = indentationNumber;
                                while (changeableIndentationNumber > 1) {
                                        string.append(tab);
                                        changeableIndentationNumber -= 1;
                                }
                                changeableIndentationNumber = indentationNumber;
                                while (changeableIndentationNumber > 1) {
                                        string.append(tab);
                                        changeableIndentationNumber -= 1;
                                }
                                changeableIndentationNumber = indentationNumber;
                                if (changeableIndentationNumber == 1) {
                                        string.append(tab);
                                }
                                string.append("Account: N/A \n");
                                changeableIndentationNumber = indentationNumber;
                                while (changeableIndentationNumber > 1) {
                                        string.append(tab);
                                        changeableIndentationNumber -= 1;
                                }
                                string.append(tab + "No. endorsements: N/A | No. comments: "
                                                + Integer.toString(deletePostsListObj.get(x).getCommentsList().size()) + "\n");
                                changeableIndentationNumber = indentationNumber;
                                while (changeableIndentationNumber > 1) {
                                        string.append(tab);
                                        changeableIndentationNumber -= 1;
                                }
                                string.append(tab + deletePostsListObj.get(x).getMessage());
                        }
                        if (indentationNumber == 0) {
                                changeableIndentationNumber = indentationNumber;
                                if (changeableIndentationNumber != 0) {
                                        string.append("\n");
                                }
                                changeableIndentationNumber = indentationNumber;
                                while (changeableIndentationNumber > 1) {
                                        string.append(tab);
                                        changeableIndentationNumber -= 1;
                                }
                                string.append("ID:");
                                string.append(Integer.toString(id) + "\n");
                                changeableIndentationNumber = indentationNumber;
                                while (changeableIndentationNumber > 1) {
                                        string.append(tab);
                                        changeableIndentationNumber -= 1;
                                }
                                string.append("Account: N/A \n");
                                changeableIndentationNumber = indentationNumber;
                                while (changeableIndentationNumber > 1) {
                                        string.append(tab);
                                        changeableIndentationNumber -= 1;
                                }
                                string.append("No. endorsements: N/A | No. comments: "
                                                + Integer.toString(deletePostsListObj.get(x).getCommentsList().size()) + "\n");
                                changeableIndentationNumber = indentationNumber;
                                while (changeableIndentationNumber > 1) {
```

```
                                        string.append(tab);
                                        changeableIndentationNumber -= 1;
                                }
                                string.append(deletePostsListObj.get(x).getMessage());
                        }

                        for (int y = 0; y < deletePostsListObj.get(x).getCommentsList().size(); y++) {
                                ArrayList<Integer> childElementsList = deletePostsListObj.get(x).getCommentsList();
                                for (int z = 0; z < childElementsList.size(); z++) {
                                        changeableIndentationNumber = indentationNumber;
                                        if (changeableIndentationNumber != 0) {
                                                string.append("\n");
                                        }
                                        changeableIndentationNumber = indentationNumber + 1;
                                        while (changeableIndentationNumber > 1) {
                                                string.append(tab);
                                                changeableIndentationNumber -= 1;
                                        }
                                        changeableIndentationNumber = indentationNumber;
                                        if (changeableIndentationNumber != 0) {
                                                string.append("|");
                                        }
                                        string.append(recursiveApproach(childElementsList.get(z)));
                                }
                        }
                        indentationNumber -= 1;
                        return string;
                }
        }
        StringBuilder string = new StringBuilder();
        indentationNumber -= 1;
        return string;
    }

    @Override
    public int getNumberOfAccounts() {
            return accountsList.size();
    }

    @Override
    public int getTotalOriginalPosts() {
            int total = 0;
            for (int pos = 0; pos < accountsList.size(); pos++) {
                    // Increments the size of original post list of the account
                    total += accountsList.get(pos).getOriginalPosts().size();
            }
            return total;
    }

    @Override
    public int getTotalEndorsmentPosts() {
            int total = 0;
            for (int pos = 0; pos < accountsList.size(); pos++) {
                    // Increments the size of endorsement list of the account
                    total += accountsList.get(pos).getEndorsements().size();
            }
            return total;
    }

    @Override
    public int getTotalCommentPosts() {
            int total = 0;
            for (int pos = 0; pos < accountsList.size(); pos++) {
                    // Increments the size of comment list of the account
                    total += accountsList.get(pos).getComments().size();
            }
            return total;
    }

    @Override
    public int getMostEndorsedPost() {
            // Stores the number of endorsements of the currently most endorsed post
            int largestNumber = -1;
            // Stores the ID of the most endorsed post
            int mostEndorsedPostID = -1;
            for (int pos = 0; pos < accountsList.size(); pos++) {
                    // Looks through all original posts
                    ArrayList<Post> postsList = accountsList.get(pos).getOriginalPosts();
                    for (int x = 0; x < postsList.size(); x++) {
                            // Compares the number endorsements with currently largest number
                            if (postsList.get(x).getEndorsementsList().size() > largestNumber) {
                                    largestNumber = postsList.get(x).getEndorsementsList().size();
                                    mostEndorsedPostID = postsList.get(x).getPostID();
                            }
                    }
                    // Looks through all comments
                    ArrayList<Comment> commentsList = accountsList.get(pos).getComments();
                    for (int x = 0; x < commentsList.size(); x++) {
                            if (commentsList.get(x).getEndorsementsList().size() > largestNumber) {
                                    largestNumber = commentsList.get(x).getEndorsementsList().size();
                                    mostEndorsedPostID = commentsList.get(x).getPostID();
```

```
                                }
                        }
                }
                return mostEndorsedPostID;
        }

        @Override
        public int getMostEndorsedAccount() {
                int maxNumberOfEndorsements = -1;
                int mostEndorsedID = -1;
                for (int pos = 0; pos < accountsList.size(); pos++) {
                        // Stores the total number of endorsements for an account
                        int numberOfEndorsements = 0;
                        ArrayList<Post> postsList = accountsList.get(pos).getOriginalPosts();
                        // Adds the number of endorsements for each post and comment
                        for (int x = 0; x < postsList.size(); x++) {
                                numberOfEndorsements += postsList.get(x).getEndorsementsList().size();
                        }
                        ArrayList<Comment> commentsList = accountsList.get(pos).getComments();
                        for (int x = 0; x < commentsList.size(); x++) {
                                numberOfEndorsements += commentsList.get(x).getEndorsementsList().size();
                        }
                        // Compares the number endorsements with currently largest number
                        if (numberOfEndorsements > maxNumberOfEndorsements) {
                                maxNumberOfEndorsements = numberOfEndorsements;
                                mostEndorsedID = accountsList.get(pos).getAccountID();
                        }
                }
                return mostEndorsedID;
        }

        @Override
        public void erasePlatform() {
                // Resets post counter
                for (Account acc : accountsList) {
                        if (acc.getOriginalPosts().size() != 0) {
                                // Not called if there are no existing posts, so the counter was not changed
                                acc.getOriginalPosts().get(0).resetNumberOfPosts();
                                break;
                        }
                }
                // Clears all the lists
                accountsList.clear();
                deletedCommentsList.clear();
                deletedPostsList.clear();
                // Also resets/clears other counters
                checkedElements.clear();
                indentationNumber = -1;
        }

        @Override
        public void savePlatform(String filename) throws IOException {
                // Creates a dictionary to store the arraylists
                Dictionary<String, Object> all = new Hashtable<>();
                all.put("Accounts", accountsList);
                all.put("DeletedPosts", deletedPostsList);
                all.put("DeletedComments", deletedCommentsList);
                try {
                        // Stores the dictionary into a file
                        FileOutputStream file = new FileOutputStream(filename + ".ser");
                        ObjectOutputStream out = new ObjectOutputStream(file);
                        out.writeObject(all);
                        out.close();
                        file.close();
                } catch (IOException e) {
                        e.printStackTrace();
                }
        }

        @Override
        public void loadPlatform(String filename) throws IOException, ClassNotFoundException {
                Dictionary<String, ArrayList<Object>> all = null;
                try {
                        // Retrieves a dictionary from a file
                        FileInputStream file = new FileInputStream(filename + ".ser");
                        ObjectInputStream in = new ObjectInputStream(file);
                        all = (Dictionary<String, ArrayList<Object>>) in.readObject();
                        in.close();
                        file.close();
                        // Replaces the arraylists with the ones in the file
                        erasePlatform();
                        for (Object obj : all.get("Accounts")) {
                                accountsList.add((Account) obj);
                        }
                        for (Object obj : all.get("DeletedPosts")) {
                                deletedPostsList.add((DeletedPost) obj);
                        }
                        for (Object obj : all.get("DeletedComments")) {
                                deletedCommentsList.add((DeletedComment) obj);
                        }
                } 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 }
```