src/socialmedia/SocialMedia.java

```java
package socialmedia;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.ArrayList;

/**
 * SocialMedia is a functioning implementor of
 * the SocialMediaPlatform interface.
 */
public class SocialMedia implements SocialMediaPlatform {
    private ArrayList<Post> arrOfPosts  = new ArrayList<Post>(0);
    private ArrayList<EndorsedPost> arrOfEndorsedPosts = new ArrayList<EndorsedPost>(0);
    private ArrayList<Comment> arrOfComments = new ArrayList<Comment>(0);
    private ArrayList<Post> arrOfEmptyPosts = new ArrayList<Post>(0);

    private ArrayList<Account> arrOfActiveAccounts = new ArrayList<Account>(0);
    private ArrayList<Account> arrOfDeactivatedAccounts = new ArrayList<Account>(0);

    @Override
    public int createAccount(String handle) throws IllegalHandleException, InvalidHandleException
{

        for (Account account : arrOfActiveAccounts) {
            if (account.getHandle().equals(handle)) // iterates through all accounts to check none
of them have already used the handle
            {
                throw new IllegalHandleException("Illegal handle: " + handle); //an illegal handle
is a handle already in use
            }
        }

        if (handle.length() == 0 || handle.length() > 30 || handle.contains(" ")) //if the handle
empty, too long or contains whitespace
        {
            throw new InvalidHandleException();
        }

        int accountID = arrOfActiveAccounts.size() + arrOfDeactivatedAccounts.size(); // generates
unique accountID

        Account account = new Account(accountID, handle, "");
        this.arrOfActiveAccounts.add(account); //adds reference to account object to list of
accounts

        return account.getAccountID();
    }

    @Override
    public int createAccount(String handle, String description) throws IllegalHandleException,
InvalidHandleException {

        for (Account account : arrOfActiveAccounts) {
            if (account.getHandle().equals(handle)) // iterates through all accounts to check none
of them have already used the handle
            {
                throw new IllegalHandleException("Illegal handle: " + handle); //an illegal handle
is a handle already in use
            }
        }
```

```
55
56          if (handle.length() == 0 || handle.length() > 30 || handle.contains(" ")) //if the handle
    empty, too long or contains whitespace
57          {
58              throw new InvalidHandleException();
59          }
60
61          int accountID = arrOfActiveAccounts.size() + arrOfDeactivatedAccounts.size(); // generates
    unique accountID
62
63          Account account = new Account(accountID, handle, description);
64          this.arrOfActiveAccounts.add(account); //adds reference to account object to list of
    accounts
65
66          return account.getAccountID();
67      }
68
69      @Override
70      public void removeAccount(int id) throws AccountIDNotRecognisedException {
71          for (Account account : arrOfActiveAccounts) {
72              if (account.getAccountID() == id) // iterates through all accounts until the desired
    account is found
73              {
74                  //Delete all posts for a given account
75                  for (Post post : account.getPosts()) {
76                      arrOfPosts.remove(post);
77                      post.setEmptyPost();
78                  }
79
80                  //Delete all endorsed posts for a given account
81                  for (EndorsedPost endorsedPost : account.getEndorsedPosts()) {
82                      arrOfEndorsedPosts.remove(endorsedPost);
83                      endorsedPost.setEmptyPost();
84                  }
85
86                  //Deletes all comments for a given account
87                  for (Comment comment : account.getComments()) {
88                      arrOfComments.remove(comment);
89                      comment.setEmptyPost();
90                  }
91
92                  arrOfActiveAccounts.remove(account);
93                  arrOfDeactivatedAccounts.add(account);
94
95                  return;
96              }
97          }
98
99          throw new AccountIDNotRecognisedException("Account ID " + id + " not found."); //if
    account not found
100     }
101
102     @Override
103     public void removeAccount(String handle) throws HandleNotRecognisedException {
104         for (Account account : arrOfActiveAccounts) {
105             if (account.getHandle().equals(handle)) // iterates through all accounts until the
    desired account is found
106             {
107                 //Delete all posts for a given account
108                 for (Post post : account.getPosts()) {
109                     arrOfPosts.remove(post);
110                     post.setEmptyPost();
111                 }
112
113                 //Deletes all endorsed posts for a given account
114                 for (EndorsedPost endorsedPost : account.getEndorsedPosts()) {
115                     arrOfEndorsedPosts.remove(endorsedPost);
```

```
116                     endorsedPost.setEmptyPost();
117                 }
118 
119                 //Deletes all comments for a given account
120                 for (Comment comment : account.getComments()) {
121                     arrOfComments.remove(comment);
122                     comment.setEmptyPost();
123                 }
124 
125                 arrOfActiveAccounts.remove(account);
126                 arrOfDeactivatedAccounts.add(account);
127 
128                 return;
129             }
130         }
131 
132         throw new HandleNotRecognisedException("Account handle " + handle + " not found."); //if account not found
133 
134     }
135 
136     @Override
137     public void changeAccountHandle(String oldHandle, String newHandle) throws HandleNotRecognisedException, IllegalHandleException, InvalidHandleException {
138         Account account = getAccountObject(oldHandle);
139         account.setHandle(newHandle); //changes handle from old one to new
140     }
141 
142     @Override
143     public void updateAccountDescription(String handle, String description) throws HandleNotRecognisedException {
144         Account account = getAccountObject(handle);
145         account.setDescription(description); //updates description
146     }
147 
148     @Override
149     public String showAccount(String handle) throws HandleNotRecognisedException {
150         Account account = getAccountObject(handle);
151 
152         int ID = account.getAccountID();
153         String desc = account.getDescription();
154         int PostCount = account.getPosts().size();
155         int EndorseCount = account.getNumEndorsements();
156 
157         String accountDetails = String.format("ID: %d \nHandle: %s \nDescription: %s \nPost Count: %d \nEndorse Count: %d", ID, handle, desc, PostCount, EndorseCount);
158 
159         return accountDetails;
160     }
161 
162     /**
163      * @param message the message to be contained within a post
164      * @return true if the message is of the correct length
165      * @throws InvalidPostException if the message length is greater than or equal
166      *                              to 100 characters
167      */
168     private boolean isMessageOfCorrectLength(String message) throws InvalidPostException {
169         if (message.length() > 100) {
170             throw new InvalidPostException("Post message contains: " + message.length()
171                     + " characters. Message can only contain up to 100 characters.");
172         } else if (message.length() == 0) {
173             throw new InvalidPostException("Post message contains: 0 characters. The post message cannot be empty.");
174         }
175         return true;
176     }
```

```java

    /**
     * This method is used to return the object of an account with a given handle.
     * If no account with that handle is found,
     *
     * @param handle handle to identify an account
     * @return the object relating to the account handle. Returns null if no account
     *         handle with the given handle was found.
     * @throws
     */
    private Account getAccountObject(String handle) throws HandleNotRecognisedException {
        Account accountObject = null;
        for (Account authorToCheck : arrOfActiveAccounts) {
            String handleToCheck = authorToCheck.getHandle();
            if (handleToCheck == handle) {
                accountObject = authorToCheck;
                break; // Searching is no longer needed.
            }
        }

        // Check if the account handle given was valid by checking if an Account with
        // that handle was found.
        if (accountObject == null) {
            throw new HandleNotRecognisedException("Handle: " + handle + " is not valid.");
        }

        return accountObject;
    }

    @Override
    public int createPost(String handle, String message) throws HandleNotRecognisedException,
InvalidPostException {
        boolean isMessageCorrectLength = isMessageOfCorrectLength(message);

        Account author = getAccountObject(handle);

        // create post object and append to arr of posts
        Post postObject = new Post(author, message);
        this.arrOfPosts.add(postObject);
        author.addPost(postObject);

        return postObject.getId();
    }

    /**
     * Checks if a post Id relates to a standard post.
     *
     * @param postID The id of the post to check
     * @return The object of the representing post if ID is a standard post.
     *         Otherwise returns null.
     */
    private Post searchForPost(int postID) {
        Post postObject = null;
        for (Post post : this.arrOfPosts) {
            int idToCheck = post.getId();

            if (idToCheck == postID) {
                postObject = post;
                break; // Stop searching for the post.
            }
        }

        return postObject;
    }

```

```java
    /**
     * Checks if a post Id relates to a comment post.
     *
     * @param commentID The id of the post to check
     * @return The object of the representing post if ID is a comment post.
     *         Otherwise returns null.
     */
    private Comment searchForComment(int commentID) {
        Comment commentObject = null;
        for (Comment comment : this.arrOfComments) {
            int idToCheck = comment.getId();

            if (idToCheck == commentID) {
                commentObject = comment;
                break; // Stop searching for the post.
            }
        }

        return commentObject;
    }

    /**
     * Checks if a post ID relates to an endorsed post
     *
     * @param endorsedPostID The ID of the post to check
     * @return The object of the representing post if ID is an endorsed post.
     *         Otherwise returns null.
     */
    private EndorsedPost searchForEndorsedPost(int endorsedPostID) {
        EndorsedPost endorsedPostObject = null;
        for (EndorsedPost endorsedPost : this.arrOfEndorsedPosts) {
            int idToCheck = endorsedPost.getId();

            if (idToCheck == endorsedPostID) {
                endorsedPostObject = endorsedPost;
                break; // Stop searching for the post.
            }
        }

        return endorsedPostObject;
    }

    /**
     * Returns whether the post with a given ID is an empty post. (a post that has
     * been deleted)
     *
     * @param postID the ID of the post to check if it is empty
     * @return true if the post is empty, false otherwise.
     */
    private boolean isPostEmpty(int postID) {
        Post postObject = searchForPost(postID);

        if (postObject == null) {
            Comment commentObject = searchForComment(postID);

            if (commentObject == null) {
                EndorsedPost endorsedPostObject = searchForEndorsedPost(postID);

                if (endorsedPostObject != null) {
                    return endorsedPostObject.isEmptyPost();
                }
            } else {
                return commentObject.isEmptyPost();
            }
        } else {
```

```
306             return postObject.isEmptyPost();
307         }
308 
309         return false;
310     }
311 
312     @Override
313     public int endorsePost(String handle, int id)
314             throws HandleNotRecognisedException, PostIDNotRecognisedException,
NotActionablePostException {
315         Account authorObject = getAccountObject(handle);
316 
317         // Search through all posts until the post with the given ID is found
318         Post postObject = searchForPost(id);
319 
320         // If a post with the given ID is not found, search for the id in comments
321         Comment commentObject = null;
322         if (postObject == null) {
323             commentObject = searchForComment(id);
324         }
325 
326         // If a comment with the given ID is not found, search for the id in endorsed
327         // posts.
328         EndorsedPost endorsedPostObject = null;
329         if (postObject == null && commentObject == null) {
330             endorsedPostObject = searchForEndorsedPost(id);
331 
332             // Throw exception if an endorsed post with the given id was found.
333             if (endorsedPostObject != null) {
334                 throw new NotActionablePostException(
335                         "You cannot endorse an endorsed post. ID of post you tried to endorse: " +
id);
336             }
337         }
338 
339         // Throw exception if no post with the given id was found
340         if (postObject == null && commentObject == null && endorsedPostObject == null) {
341             throw new PostIDNotRecognisedException("The post with ID: " + id + " was not found.");
342         }
343 
344         // If a post with given ID is found, check that the post is not an empty post.
345         // Throw exception if the corresponding ID is to an empty post.
346         if (isPostEmpty(id)) {
347             throw new NotActionablePostException("You cannot endorse an empty post.");
348         }
349 
350         // create an endorsed post object and append to arr of posts
351         if (postObject != null) {
352             endorsedPostObject = new EndorsedPost(authorObject, postObject);
353         }
354         else if (commentObject != null) {
355             endorsedPostObject = new EndorsedPost(authorObject, commentObject);
356         }
357 
358         this.arrOfEndorsedPosts.add(endorsedPostObject);
359         authorObject.addEndorsedPost(endorsedPostObject);
360 
361         return endorsedPostObject.getId();
362     }
363 
364     @Override
365     public int commentPost(String handle, int id, String message) throws
HandleNotRecognisedException,
366             PostIDNotRecognisedException, NotActionablePostException, InvalidPostException {
367         Account authorObject = getAccountObject(handle);
368 
```

```
        Post postObject = searchForPost(id);

        // If a post with given ID is not found, search through all comments until the
        // given post id is found.
        Comment commentObject = null;
        if (postObject == null) {
            postObject = searchForComment(id);

        }

        // If a post with given ID is not found, search through all endorsed posts until
        // the post width given id is found.
        EndorsedPost endorsedPostObject = null;
        if (postObject == null && commentObject == null) {
            endorsedPostObject = searchForEndorsedPost(id);

            // Throw exception if an endorsed post with the given id was found.
            if (endorsedPostObject != null) {
                throw new NotActionablePostException(
                        "You cannot endorse an endorsed post. ID of post you tried to endorse: " +
id);
            }
        }

        // Throw exception if no post with the given id was found
        if (postObject == null && commentObject == null && endorsedPostObject == null) {
            throw new PostIDNotRecognisedException("The post with ID: " + id + " was not found.");
        }

        // If a post with given ID is found, check that the post is not an empty post.
        // Throw exception if the post found is an empty post.
        if (isPostEmpty(id)) {
            throw new NotActionablePostException("You cannot endorse an empty post.");
        }

        // Check that the provided message is within the allowed length (0 to 100 chars)
        isMessageOfCorrectLength(message);

        // create post object and append to arr of posts
        commentObject = new Comment(authorObject, postObject, message);
        this.arrOfComments.add(commentObject);
        authorObject.addComment(commentObject);

        return commentObject.getId();
    }

    @Override
    public void deletePost(int id) throws PostIDNotRecognisedException {
        // If the post is an empty post, say it is not found
        if (isPostEmpty(id)) {
            throw new PostIDNotRecognisedException("The post with ID: " + id + " was not found.");
        }

        Post postObject = searchForPost(id);

        ArrayList<EndorsedPost> arrOfPostEndorsements  = new ArrayList<EndorsedPost>(0);

        // If a post with given ID is not found, search through all comments until the
        // post width given id is found.
        Comment commentObject = null;
        if (postObject == null) {
            commentObject = searchForComment(id);
        }
        else if (postObject != null) {
            //Remove the post from an account's list of posts
```

```
            Account authorObject = postObject.getAuthor();
            authorObject.removePost(postObject);

            //Get an array of endorsements for this post.
            arrOfPostEndorsements = postObject.getArrayOfEndorsements();

            // Turn the post into an empty post
            postObject.setEmptyPost();
            arrOfEmptyPosts.add(postObject);
        }

        // If a post with given ID is not found, search through all endorsed posts until
        // the post with the given id is found.
        EndorsedPost endorsedPostObject = null;
        if (postObject == null && commentObject == null) {
            endorsedPostObject = searchForEndorsedPost(id);
        }
        else if (commentObject != null){
            //Remove the post from an account's list of posts
            Account authorObject = commentObject.getAuthor();
            authorObject.removeComment(commentObject);

            //Get an array of endorsements for this post.
            arrOfPostEndorsements = commentObject.getArrayOfEndorsements();

            // Turn the post into an empty post
            commentObject.setEmptyPost();
            arrOfEmptyPosts.add(commentObject);
        }

        // Throw exception if no post with the given id was found
        if (postObject == null && commentObject == null && endorsedPostObject == null) {
            throw new PostIDNotRecognisedException("The post with ID: " + id + " was not found.");
        }
        else if (endorsedPostObject != null) {
            //Remove the post from an account's list of posts
            Account authorObject = endorsedPostObject.getAuthor();
            authorObject.removeEndorsedPost(endorsedPostObject);

            //Get an array of endorsements for this post.
            arrOfPostEndorsements = endorsedPostObject.getArrayOfEndorsements();

            // Turn the post into an empty post
            endorsedPostObject.setEmptyPost();
            arrOfEmptyPosts.add(endorsedPostObject);
        }

        //Remove all endorsements for this post.
        for (EndorsedPost epObject : arrOfPostEndorsements) {
            arrOfEndorsedPosts.remove(epObject);
        }

    }

    @Override
    public String showIndividualPost(int id) throws PostIDNotRecognisedException {
        // Check the id is valid, and get the post object.
        //Initaalize variables
        Account author = null;
        String handle = null;
        Integer commentId = null;
        Integer numEndorsements = null;
        Integer numComments = null;
        String message = null;

```

```
498         // Search through all posts until the post with the given ID is found
499         Post postObject = searchForPost(id);
500 
501         // If a post with given ID is not found, search through all comments until the
502         // post width given id is found.
503         Comment commentObject = null;
504         if (postObject == null) {
505             commentObject = searchForComment(id);
506 
507         } else {
508             author = postObject.getAuthor();
509             handle = (!postObject.isEmptyPost()) ? author.getHandle() : "";
510             commentId = postObject.getId();
511             numEndorsements = postObject.getNumberOfEndorsements();
512             numComments = postObject.getNumberOfComments();
513             message = postObject.getMessage();
514         }
515 
516         // If a post with given ID is not found, search through all endorsed posts until
517         // the post width given id is found.
518         EndorsedPost endorsedPostObject = null;
519         if (postObject == null && commentObject == null) {
520             endorsedPostObject = searchForEndorsedPost(id);
521         }
522         else if(commentObject != null)  {
523             author = commentObject.getAuthor();
524             handle = (!commentObject.isEmptyPost()) ? author.getHandle() : "";
525             commentId = commentObject.getId();
526             numEndorsements = commentObject.getNumberOfEndorsements();
527             numComments = commentObject.getNumberOfComments();
528             message = commentObject.getMessage();
529         }
530 
531         // Throw exception if a post with the given id was not found.
532         if (postObject == null && commentObject == null && endorsedPostObject == null) {
533             throw new PostIDNotRecognisedException("The post with ID: " + id + " was not found.");
534         } else if(endorsedPostObject != null) {
535             author = endorsedPostObject.getAuthor();
536             handle = (!endorsedPostObject.isEmptyPost()) ? author.getHandle() : "";
537             commentId = endorsedPostObject.getId();
538             numEndorsements = endorsedPostObject.getNumberOfEndorsements();
539             numComments = endorsedPostObject.getNumberOfComments();
540             message = endorsedPostObject.getMessage();
541         }
542 
543         String postDetails = String.format("ID: %d"
544             + "\n " + "Account: %s"
545             + "\n " + "No. endorsements: %d | No. comments: %d"
546             + "\n " + "%s", commentId, handle, numEndorsements, numComments, message);
547         return postDetails;
548     }
549 
550     @Override
551     public StringBuilder showPostChildrenDetails(int id)
552             throws PostIDNotRecognisedException, NotActionablePostException {
553         // Search through all posts until the post with the given ID is found
554         Post postObject = searchForPost(id);
555 
556         // If a post with given ID is not found, search through all comments until the
557         // post width given id is found.
558         Comment commentObject = null;
559         if (postObject == null) {
560             commentObject = searchForComment(id);
561 
```

```
562         }
563 
564         // If a post with given ID is not found, search through all endorsed posts until
565         // the post width given id is found.
566         EndorsedPost endorsedPostObject = null;
567         if (postObject == null && commentObject == null) {
568             endorsedPostObject = searchForEndorsedPost(id);
569 
570             // Throw exception if an endorsed post with the given id was found.
571             if (endorsedPostObject != null) {
572                 throw new NotActionablePostException(
573                         "You cannot show the children of an endorsed post.");
574             }
575         }
576 
577         // Output the initial post
578         String stringToReturn = "";
579         stringToReturn += showIndividualPost(id);
580 
581         // For each comment in the post, display the comment along with all of it's
582         // comments
583         for (Comment comment : postObject.getArrayOfComments()) {
584 
585             stringToReturn += showComments(comment, 1);
586         }
587 
588         // Convert the string to stringbuilder
589         StringBuilder sb = new StringBuilder();
590         sb.append(stringToReturn);
591 
592         return sb;
593 
594     }
595 
596     private String showComments(Comment commentObject, int commentLevel) {
597         String stringToReturn = "";
598 
599         // Calculate the spacing to the left of the message
600         String spacing = "";
601         String idSpacing = "";
602         for (int i = 0; i < commentLevel; i++) {
603             spacing += "    ";
604 
605             if (i > 0) {
606                 idSpacing += "    ";
607             }
608         }
609 
610         // Add to the string to return the comment of which this function was called by
611         Account author = commentObject.getAuthor();
612         String handle = (!commentObject.isEmptyPost()) ? author.getHandle() : "";
613         Integer commentId = commentObject.getId();
614         Integer numEndorsements = commentObject.getNumberOfEndorsements();
615         Integer numComments = commentObject.getNumberOfComments();
616         String message = commentObject.getMessage();
617 
618         stringToReturn += String.format("\n " + idSpacing + "| > ID: %d"
619                 + "\n " + spacing + "Account: %s"
620                 + "\n " + spacing + "No. endorsements: %d | No. comments: %d"
621                 + "\n " + spacing + "%s", commentId, handle, numEndorsements, numComments,
message);
622 
623         // Call this function again for any further comments on this comment post.
624         for (Comment comment : commentObject.getArrayOfComments()) {
625             stringToReturn += showComments(comment, commentLevel + 1);
```

```
626         }
627 
628         return stringToReturn;
629     }
630 
631     @Override
632     public int getNumberOfAccounts() {
633         return this.arrOfActiveAccounts.size();
634     }
635 
636     @Override
637     public int getTotalOriginalPosts() {
638         Integer numPosts = 0;
639         for (Post post : arrOfPosts) {
640             if (!post.isEmptyPost()) {
641                 numPosts++;
642             }
643         }
644 
645         return numPosts;
646     }
647 
648     @Override
649     public int getTotalEndorsmentPosts() {
650         Integer numPosts = 0;
651         for (Post post : arrOfEndorsedPosts) {
652             if (!post.isEmptyPost()) {
653                 numPosts++;
654             }
655         }
656 
657         return numPosts;
658     }
659 
660     @Override
661     public int getTotalCommentPosts() {
662         Integer numPosts = 0;
663         for (Post post : arrOfComments) {
664             if (!post.isEmptyPost()) {
665                 numPosts++;
666             }
667         }
668 
669         return numPosts;
670     }
671 
672     @Override
673     public int getMostEndorsedPost() {
674         int greatestNumberOfEndorsements = -1;
675         int postID = -1;
676         // Search through all posts and find the most endorsed post
677         for (Post post : this.arrOfPosts) {
678             if (post.getNumberOfEndorsements() >= greatestNumberOfEndorsements) {
679                 greatestNumberOfEndorsements = post.getNumberOfEndorsements();
680                 postID = post.getId();
681             }
682         }
683 
684         // Search through all comments and find the most endorsed comment
685         for (Comment comment : this.arrOfComments) {
686             if (comment.getNumberOfEndorsements() >= greatestNumberOfEndorsements) {
687                 greatestNumberOfEndorsements = comment.getNumberOfEndorsements();
688                 postID = comment.getId();
689             }
690         }
```

```
691
692         return postID;
693     }
694
695     @Override
696     public int getMostEndorsedAccount() {
697         Account mostEndorsedAccount = null;
698         int numEndorsementsOfMaxAccount = 0;
699
700         for (Account account : arrOfActiveAccounts) {
701             if (account.getNumEndorsements() >= numEndorsementsOfMaxAccount) {
702                 mostEndorsedAccount = account;
703             }
704         }
705
706         return mostEndorsedAccount.getAccountID();
707     }
708
709     @Override
710     public void erasePlatform() {
711         this.arrOfPosts  = new ArrayList<Post>(0);
712         this.arrOfEndorsedPosts = new ArrayList<EndorsedPost>(0);
713         this.arrOfComments = new ArrayList<Comment>(0);
714         this.arrOfEmptyPosts = new ArrayList<Post>(0);
715
716         this.arrOfActiveAccounts = new ArrayList<Account>(0);
717         this.arrOfDeactivatedAccounts = new ArrayList<Account>(0);
718
719         Post.setNextPostId(0);
720     }
721
722     @Override
723     public void savePlatform(String filename) throws IOException {
724         try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(filename))){
725             oos.writeObject(this.arrOfPosts);
726             oos.writeObject(this.arrOfEndorsedPosts);
727             oos.writeObject(this.arrOfComments);
728             oos.writeObject(this.arrOfEmptyPosts);
729
730             oos.writeObject(this.arrOfActiveAccounts);
731             oos.writeObject(this.arrOfDeactivatedAccounts);
732
733         } catch (IOException e) {
734             new IOException("Error when saving platform.");
735         }
736     }
737
738     @Override
739     public void loadPlatform(String filename) throws IOException, ClassNotFoundException {
740         try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(filename))) {
741
742             //Update the fields of the current object
743             this.arrOfPosts = (ArrayList<Post>) ois.readObject();
744             this.arrOfEndorsedPosts = (ArrayList<EndorsedPost>) ois.readObject();
745             this.arrOfComments = (ArrayList<Comment>) ois.readObject();
746             this.arrOfEmptyPosts = (ArrayList<Post>) ois.readObject();
747
748             this.arrOfActiveAccounts = (ArrayList<Account>) ois.readObject();
749             this.arrOfDeactivatedAccounts = (ArrayList<Account>) ois.readObject();
750
751         } catch (IOException e) {
752             new IOException("Error when loading platform.", e);
753         } catch (ClassNotFoundException e) {
754             new ClassNotFoundException("Error when loading platform.", e);
755         }
```

```
    }

}

```

src/socialmedia/Post.java

```java
package socialmedia;

import java.util.ArrayList;
import java.io.Serializable;

public class Post implements Serializable {
    private Account author;
    private String message;
    private static int nextPostID;
    private int postID;
    private ArrayList<Comment> arrOfComments = new ArrayList<Comment>(0);
    private ArrayList<EndorsedPost> arrOfEndorsements  = new ArrayList<EndorsedPost>(0);

    private boolean isEmptyPost;


    /**<p>Creates a new instance of the Post class.</p>
     *
     * Parameters:
     * handle (String): The handle of the user posting the post
     * message (String): The message to post
    */
    public Post(Account author, String message) throws InvalidPostException {
        this.author = author;
        setMessage(message);
        author.addPost(this); //Add the post to the list of posts created by the author
        this.isEmptyPost = false;
        //Increment and set the post id. (this means the id will start from 1)
        //TODO: if the id somehow reaches 2147483648 the program will crash
        this.postID = nextPostID;
        nextPostID++;
    }

    public Post(Account author, Post postObject) {
        this.author = author;
        this.message = postObject.getMessage();
        addEndorseeHandleToMessage(author.getHandle());

        author.addPost(this); //Add the post to the list of posts created by the author
        this.isEmptyPost = false;
        //Increment and set the post id. (this means the id will start from 1)
        this.postID = nextPostID;
        nextPostID++;
    }

    public Account getAuthor() {
        return author;
    }

    public String getMessage() {
        return message;
    }

    public int getId() {
        return postID;
    }

    public ArrayList<Comment> getArrayOfComments() {
        return arrOfComments;
    }
```

```java

    public int getNumberOfComments() {
        return arrOfComments.size();
    }

    public int getNumberOfEndorsements() {
        return arrOfEndorsements.size();
    }

    public ArrayList<EndorsedPost> getArrayOfEndorsements() {
        return this.arrOfEndorsements;
    }

    public void setAuthor(Account newAuthor) {
        author = newAuthor;
        //TODO: validate
    }

    /**Sets the message of the post to a new message. The message can be up to 100 characters long.
     *
     * @param newMessage (String) the new post message.
     *
     * @outputs (String) if the message is greater than 100 characters long, a warning message is output to the console and the message is not set.
     */
    public void setMessage(String newMessage) throws InvalidPostException {
        if (newMessage.length() <= 100) {
            message = newMessage;
        } else if (newMessage.length() > 100) {
            throw new InvalidPostException("Post message contains: " + newMessage.length()
                    + " characters. Message can only contain up to 100 characters.");
        } else {
            throw new InvalidPostException("Post message contains: 0 characters. The post message cannot be empty.");
        }
    }

    public void setEmptyPost() {
        this.author = null;
        this.message = "The original content was removed from the system and is no longer available.";
        this.isEmptyPost = true;
        this.arrOfEndorsements  = new ArrayList<EndorsedPost>(0);
    }

    public void setPostId(int newPostId) {
        postID = newPostId;
    }

    public static void setNextPostId(int newNextPostId) {
        nextPostID = newNextPostId;
    }

    public void addComment(Comment commentObject) {
         if (!this.isEmptyPost()) {
            arrOfComments.add(commentObject);
        }
    }

    public void removeComment(Comment commentObject) {
        arrOfComments.remove(commentObject);
    }

    public void addEndorsedPost(EndorsedPost endorsedPostObject) {
        if (!this.isEmptyPost()) {
```

```
123             arrOfEndorsements.add(endorsedPostObject);
124         }
125     }
126
127     public void removeEndorsedPost(EndorsedPost endorsedPostObject) {
128         arrOfEndorsements.remove(endorsedPostObject);
129     }
130
131     public void addEndorseeHandleToMessage(String endorseeHandle) {
132         this.message = "EP@" + endorseeHandle + ": " + this.getMessage();
133     }
134
135     public boolean isEmptyPost() {
136         return this.isEmptyPost;
137     }
138 }
```

src/socialmedia/EndorsedPost.java

```java
package socialmedia;

public class EndorsedPost extends Post {
    private Post postObject;

    public EndorsedPost(Account authorObject, Post postObject) {
        //Call Post constructor
        super(authorObject, postObject);
        this.postObject = postObject;

        //Change the post message to include handle of user to endorse
        postObject.addEndorsedPost(this);
    }

    public Post getParentPost()
    {
        return postObject;
    }

    @Override
    /**<p>DO NOT USE</p>
     * Overrides the existing superclass method, outputs a warning message to the console:
     * "The operation: 'addComment' cannot be performed on an endorsed post.\n
     *  operation FAILED!"
    */
    public void addComment(Comment commentObject) {
        //System.out.println("The operation: 'addComment' cannot be performed on an endorsed post.\noperation FAILED!");
    }

    @Override
    /**<p>DO NOT USE</p>
     * Overrides the existing superclass method, outputs a warning message to the console:
     * "The operation: 'removeComment' cannot be performed on an endorsed post.\n
     *  operation FAILED!"
    */
    public void removeComment(Comment commentObject) {
        //System.out.println("The operation: 'removeComment' cannot be performed on an endorsed post.\noperation FAILED!");
    }

    @Override
    /**<p>DO NOT USE</p>
     * Overrides the existing superclass method, outputs a warning message to the console:
     * "The operation: 'addEndorsement' cannot be performed on an endorsed post.\n
     *  operation FAILED!"
    */
    public void addEndorsedPost(EndorsedPost endorsedPostObject) {
        //System.out.println("The operation: 'addEndorsement' cannot be performed on an endorsed post.\noperation FAILED!");
    }

    @Override
    /**<p>DO NOT USE</p>
     * Overrides the existing superclass method, outputs a warning message to the console:
     * "The operation: 'removeComment' cannot be performed on an endorsed post.\n
     *  operation FAILED!"
    */
    public void removeEndorsedPost(EndorsedPost endorsedPostObject) {
        //System.out.println("The operation: 'removeEndorsedPost' cannot be performed on an endorsed post.\noperation FAILED!");
    }
```

```
59
60      //TODO better implementation of removing the ability to endorse and comment on posts
61  }
62
```

src/socialmedia/Comment.java

```
1  package socialmedia;
2
3  public class Comment extends Post {
4      private Post postCommentingToObject;
5
6      public Comment(Account authorObject, Post postCommentingToObject, String messageToComment)
   throws InvalidPostException{
7          super(authorObject, messageToComment);
8          this.postCommentingToObject = postCommentingToObject;
9          postCommentingToObject.addComment(this);
10     }
11
12     public Post getParentPost()
13     {
14         return postCommentingToObject;
15     }
16 }
17
```

src/socialmedia/Account.java

```java
package socialmedia;

import java.util.ArrayList;
import java.io.Serializable;

public class Account implements Serializable {
    private int accountID;
    private String handle;
    private String description;
    private ArrayList<Post> Posts = new ArrayList<Post>(0); //used to track which posts the author has created
    private ArrayList<Comment> Comments = new ArrayList<Comment>(0);
    private ArrayList<EndorsedPost> EndorsedPosts = new ArrayList<EndorsedPost>();
    private int numEndorsements = 0;

    /**<p>Creates a new instance of the Account class.</p>
     *
     * Parameters:
     * ID: The ID of the account to be created
     * Username (String): The handle of the account
     * bio (String): The bio of the new account
    */
    public Account(int ID, String Username, String bio)
    {
        this.accountID = ID;
        this.handle = Username;
        this.description = bio;
        this.numEndorsements = 0;
    }

    public int getAccountID() {
        return this.accountID;
    }

    public String getHandle() {
        return this.handle;
    }

    public String getDescription() {
        return this.description;
    }

    public void setAccountID(int newID) {
        this.accountID = newID;
    }

    public void setHandle(String newHandle) {
        this.handle = newHandle;
    }

    public void setDescription(String newDescription) {
        this.description = newDescription;
    }

    public void giveEndorsement()
    {
        this.numEndorsements = this.numEndorsements + 1; //increments number of endorsements
    }

    public int getNumEndorsements()
    {
```

```java
        return this.numEndorsements;
    }

    public ArrayList<Post> getPosts()
    {
        return this.Posts;
    }

    public void addPost(Post postObject) {
        Posts.add(postObject);
    }

    public void removePost(Post postObject) {
        Posts.remove(postObject);
    }

    public ArrayList<Comment> getComments()
    {
        return this.Comments;
    }

    public void addComment(Comment commentObject) {
        Comments.add(commentObject);
    }

    public void removeComment(Comment commentObject) {
        Comments.remove(commentObject);
    }

    public ArrayList<EndorsedPost> getEndorsedPosts()
    {
        return this.EndorsedPosts;
    }

    public void addEndorsedPost(EndorsedPost endorsedPostObject) {
        EndorsedPosts.add(endorsedPostObject);
    }

    public void removeEndorsedPost(EndorsedPost endorsedPostObject) {
        EndorsedPosts.remove(endorsedPostObject);
    }

}

```

src/SocialMediaPlatformTestApp.java

```java
import java.io.IOException;

import socialmedia.AccountIDNotRecognisedException;
import socialmedia.SocialMedia;
import socialmedia.HandleNotRecognisedException;
import socialmedia.IllegalHandleException;
import socialmedia.InvalidHandleException;
import socialmedia.InvalidPostException;
import socialmedia.NotActionablePostException;
import socialmedia.PostIDNotRecognisedException;
import socialmedia.SocialMediaPlatform;

//testing

/**
 * A short program to illustrate an app testing some minimal functionality of a
 * concrete implementation of the SocialMediaPlatform interface -- note you will
 * want to increase these checks, and run it on your SocialMedia class (not the
 * BadSocialMedia class).
 *
 *
 * @author Diogo Pacheco
 * @version 1.0
 */
public class SocialMediaPlatformTestApp {

    /**
     * Test method.
     *
     * @param args not used
     */
    public static void main(String[] args) {
        System.out.println("The system compiled and started the execution...");

        SocialMediaPlatform platform = new SocialMedia();

        assert (platform.getNumberOfAccounts() == 0) : "Innitial SocialMediaPlatform not empty as required.";
        assert (platform.getTotalOriginalPosts() == 0) : "Innitial SocialMediaPlatform not empty as required.";
        assert (platform.getTotalCommentPosts() == 0) : "Innitial SocialMediaPlatform not empty as required.";
        assert (platform.getTotalEndorsmentPosts() == 0) : "Innitial SocialMediaPlatform not empty as required.";

        // Test the platform after initialization
        testPlatform(platform);

        // Erase the platform and re-test it.
        platform.erasePlatform();
        testPlatform(platform);


        // Save the platform, erase the current platform, load the new platform and test
        // it again.
        try{
            platform.savePlatform("test_platform.ser");
            platform.erasePlatform();
        } catch (IOException e) {
            new IOException("error saving platform");
        }

```

```
59          try{
60              platform.loadPlatform("test_platform.ser");
61              testLoadedPlatform(platform);
62          } catch (IOException e) {
63              new IOException("error loading platform");
64          } catch (ClassNotFoundException e) {
65              new ClassNotFoundException("class not found when trying to load platform.");
66          }
67
68
69          //Save the current platform, load the new platform then test it.
70          try{
71              platform.savePlatform("test_platform.ser");
72          } catch (IOException e) {
73              new IOException("error saving platform");
74          }
75
76          try{
77              platform.loadPlatform("test_platform.ser");
78              testLoadedPlatform(platform);
79          } catch (IOException e) {
80              new IOException("error loading platform");
81          } catch (ClassNotFoundException e) {
82              new ClassNotFoundException("class not found when trying to load platform.");
83          }
84
85      }
86
87      private static void testPlatform(SocialMediaPlatform platform) {
88
89          //Check that an account can be created then can be deleted from the platform
90          try {
91              Integer id;
92              id = platform.createAccount("my_handle");
93              assert (platform.getNumberOfAccounts() == 1) : "number of accounts registered in the
    system does not match";
94
95              platform.removeAccount(id);
96              assert (platform.getNumberOfAccounts() == 0) : "number of accounts registered in the
    system does not match";
97
98          } catch (IllegalHandleException e) {
99              assert (false) : "IllegalHandleException thrown incorrectly";
100         } catch (InvalidHandleException e) {
101             assert (false) : "InvalidHandleException thrown incorrectly";
102         } catch (AccountIDNotRecognisedException e) {
103             assert (false) : "AccountIDNotRecognizedException thrown incorrectly";
104         }
105
106         // Create a post then delete the post, check that the post is added to the platform then
    deleted.
107         try {
108             Integer postID2;
109             Integer accountID = platform.createAccount("my_handle");
110             assert (platform.getNumberOfAccounts() == 1) : "number of accounts registered in the
    system does not match";
111
112             postID2 = platform.createPost("my_handle", "Hello.");
113             assert (platform.getTotalOriginalPosts()) == 1 : "number of posts registered in the
    system does not match";
114
115             platform.deletePost(postID2);
116             assert (platform.getTotalOriginalPosts()) == 0 : "number of posts registered in the
    system does not match";
117
118             platform.removeAccount(accountID);
119             assert (platform.getNumberOfAccounts() == 0) : "number of accounts registered in the
```

```
system does not match";

        } catch (PostIDNotRecognisedException e) {
            assert (false) : "PostIDNotRecognisedException thrown incorrectly";
        } catch (HandleNotRecognisedException e) {
            assert (false) : "HandleNotRecognisedException thrown incorrectly";
        } catch (InvalidPostException e) {
            assert (false) : "InvalidPostException thrown incorrectly";
        } catch (IllegalHandleException e) {
            assert (false) : "IllegalHandleException thrown incorrectly";
        } catch (InvalidHandleException e) {
            assert (false) : "InvalidHandleException thrown incorrectly";
        } catch (AccountIDNotRecognisedException e) {
            assert (false) : "AccountIDNotRecognizedException thrown incorrectly";
        }

        // Create a new account to use in future tests
        try {
            Integer id;
            id = platform.createAccount("my_handle");
            assert (platform.getNumberOfAccounts() == 1) : "number of accounts registered in the
system does not match";

        } catch (IllegalHandleException e) {
            assert (false) : "IllegalHandleException thrown incorrectly";
        } catch (InvalidHandleException e) {
            assert (false) : "InvalidHandleException thrown incorrectly";
        }

        // Create a post then endorse that post, delete the endorsed post then the
        // original post. Check that both posts are deleted.
        try {
            Integer postID3;
            postID3 = platform.createPost("my_handle", "You Too.");
            assert (platform.getTotalOriginalPosts()) == 1 : "number of posts registered in the
system does not match";

            int endorsedPostID = platform.endorsePost("my_handle", postID3);
            assert (platform.getTotalEndorsmentPosts()) == 1
                    : "number of endorsed posts registered in the system does not match";

            platform.deletePost(endorsedPostID);
            assert (platform.getTotalEndorsmentPosts()) == 0
                    : "number of endorsed posts registered in the system does not match";

            platform.deletePost(postID3);
            assert (platform.getTotalOriginalPosts()) == 0 : "number of posts registered in the
system does not match";

        } catch (HandleNotRecognisedException e) {
            assert (false) : "HandleNotRecognisedException thrown incorrectly";
        } catch (InvalidPostException e) {
            assert (false) : "InvalidPostException thrown incorrectly";
        } catch (PostIDNotRecognisedException e) {
            assert (false) : "PostIDNotRecognisedException thrown incorrectly";
        } catch (NotActionablePostException e) {
            assert (false) : "NotActionablePostException thrown incorrectly";
        }

        // Create a post, endorse that post then delete the original post.
        // Check that both the original post and the endorsed posts are deleted.
        Integer postID5;
        try {
            postID5 = platform.createPost("my_handle", "You Too.");
            assert (platform.getTotalOriginalPosts()) == 1 : "number of posts registered in the
system does not match";
```

```
181
182             int endorsedPostId = platform.endorsePost("my_handle", postID5);
183             assert (platform.getTotalEndorsmentPosts()) == 1
184                     : "number of endorsed posts registered in the system does not match";
185
186             platform.deletePost(postID5);
187             assert (platform.getTotalOriginalPosts()) == 0
188                     : "number of original posts registered in the system does not match";
189             assert (platform.getTotalEndorsmentPosts()) == 0
190             : "number of endorsed posts registered in the system does not match";
191
192         } catch (HandleNotRecognisedException e) {
193             assert (false) : "HandleNotRecognisedException thrown incorrectly";
194         } catch (InvalidPostException e) {
195             assert (false) : "InvalidPostException thrown incorrectly";
196         } catch (PostIDNotRecognisedException e) {
197             System.out.println(e);
198             assert (false) : "PostIDNotRecognisedException thrown incorrectly";
199         } catch (NotActionablePostException e) {
200             assert (false) : "NotActionablePostException thrown incorrectly";
201         }
202
203         // Create a post, comment on that post, delete the comment and the post.
204         // Check that both post and comment are deleted.
205         Integer postID6;
206         try {
207             postID6 = platform.createPost("my_handle", "What's the weather today?");
208             assert (platform.getTotalOriginalPosts()) == 1 : "number of posts registered in the
    system does not match";
209
210             Integer commentID = platform.commentPost("my_handle", postID6, "sunny.");
211             assert (platform.getTotalCommentPosts()) == 1
212                     : "number of comment posts registered in the system does not match";
213
214             platform.deletePost(commentID);
215             assert (platform.getTotalCommentPosts()) == 0
216                     : "number of endorsed posts registered in the system does not match";
217             assert (platform.getTotalOriginalPosts()) == 1
218                     : "number of original posts registered in the system does not match";
219
220             platform.deletePost(postID6);
221             assert (platform.getTotalOriginalPosts()) == 0
222                     : "number of original posts registered in the system does not match";
223             assert (platform.getTotalCommentPosts()) == 0
224                     : "number of endorsed posts registered in the system does not match";
225
226         } catch (HandleNotRecognisedException e) {
227             assert (false) : "HandleNotRecognisedException thrown incorrectly";
228         } catch (InvalidPostException e) {
229             assert (false) : "InvalidPostException thrown incorrectly";
230         } catch (PostIDNotRecognisedException e) {
231             assert (false) : "PostIDNotRecognisedException thrown incorrectly";
232         } catch (NotActionablePostException e) {
233             assert (false) : "NotActionablePostException thrown incorrectly";
234         }
235
236         // Create a post, comment on that post, delete the post.
237         // Check that the original post is deleted, but the comment is not.
238         // Then delete the comment.
239         Integer postID7;
240         try {
241             postID7 = platform.createPost("my_handle", "What's the weather today?");
242             assert (platform.getTotalOriginalPosts()) == 1 : "number of posts registered in the
    system does not match";
243
244             Integer commentID = platform.commentPost("my_handle", postID7, "sunny.");
```

```
245             assert (platform.getTotalCommentPosts()) == 1
246                     : "number of comment posts registered in the system does not match";
247 
248             platform.deletePost(postID7);
249             assert (platform.getTotalOriginalPosts()) == 0
250                     : "number of original posts registered in the system does not match";
251             assert (platform.getTotalCommentPosts()) == 1
252                     : "number of comment posts registered in the system does not match";
253 
254             platform.deletePost(commentID);
255             assert (platform.getTotalCommentPosts()) == 0
256                     : "number of comment posts registered in the system does not match";
257             assert (platform.getTotalOriginalPosts()) == 0
258                     : "number of original posts registered in the system does not match";
259 
260         } catch (HandleNotRecognisedException e) {
261             assert (false) : "HandleNotRecognisedException thrown incorrectly";
262         } catch (InvalidPostException e) {
263             assert (false) : "InvalidPostException thrown incorrectly";
264         } catch (PostIDNotRecognisedException e) {
265             assert (false) : "PostIDNotRecognisedException thrown incorrectly";
266         } catch (NotActionablePostException e) {
267             assert (false) : "NotActionablePostException thrown incorrectly";
268         }
269 
270         // Try to comment on an empty post
271         // Check that a postnotactionable exception is thrown.
272         try {
273             Integer postID = platform.createPost("my_handle", "How u doin?");
274             assert (platform.getTotalOriginalPosts()) == 1 : "number of posts registered in the
    system does not match";
275 
276             platform.deletePost(postID);
277             assert (platform.getTotalOriginalPosts()) == 0
278                     : "number of original posts registered in the system does not match";
279 
280             platform.commentPost("my_handle", postID, "Good thanks u?");
281             assert (platform.getTotalCommentPosts()) == 0
282                     : "number of comment posts registered in the system does not match";
283         } catch (HandleNotRecognisedException e) {
284             assert (false) : "HandleNotRecognisedException thrown incorrectly";
285         } catch (InvalidPostException e) {
286             assert (false) : "InvalidPostException thrown incorrectly";
287         } catch (PostIDNotRecognisedException e) {
288             assert (false) : "PostIDNotRecognisedException thrown incorrectly";
289         } catch (NotActionablePostException e) {
290             assert (true);
291         }
292 
293         // Try to endorse an empty post
294         // Check that a postnotactionable exception is thrown.
295         try {
296             Integer postID = platform.createPost("my_handle", "How u doin?");
297             assert (platform.getTotalOriginalPosts()) == 1 : "number of posts registered in the
    system does not match";
298 
299             platform.deletePost(postID);
300             assert (platform.getTotalOriginalPosts()) == 0
301                     : "number of original posts registered in the system does not match";
302 
303             platform.endorsePost("my_handle", postID);
304             assert (platform.getTotalEndorsmentPosts()) == 0
305                     : "number of endorsed posts registered in the system does not match";
306 
307         } catch (HandleNotRecognisedException e) {
```

```
            assert (false) : "HandleNotRecognisedException thrown incorrectly";
        } catch (InvalidPostException e) {
            assert (false) : "InvalidPostException thrown incorrectly";
        } catch (PostIDNotRecognisedException e) {
            assert (false) : "PostIDNotRecognisedException thrown incorrectly";
        } catch (NotActionablePostException e) {
            assert (true);
        }

        // Try to delete an empty post
        // Check that a postidnotrecognised exception is thrown.
        try {
            Integer postID = platform.createPost("my_handle", "How u doin?");
            assert (platform.getTotalOriginalPosts()) == 1 : "number of posts registered in the system does not match";

            platform.deletePost(postID);
            assert (platform.getTotalOriginalPosts()) == 0
                    : "number of original posts registered in the system does not match";

            platform.deletePost(postID);
            assert (platform.getTotalOriginalPosts()) == 0
                    : "number of original posts registered in the system does not match";

        } catch (HandleNotRecognisedException e) {
            assert (false) : "HandleNotRecognisedException thrown incorrectly";
        } catch (InvalidPostException e) {
            assert (false) : "InvalidPostException thrown incorrectly";
        } catch (PostIDNotRecognisedException e) {
            assert (true);
        }

        // Output a post as a string
        // Check that the output string is the desired output.
        try {
            Integer postID = platform.createPost("my_handle", "Hello.");
            assert (platform.getTotalOriginalPosts()) == 1 : "number of posts registered in the system does not match";

            platform.showIndividualPost(postID);
            assert (platform.showIndividualPost(postID).equals(
                    "ID: " + postID + "\n Account: my_handle\n No. endorsements: 0 | No. comments: 0\n Hello.")) == true
                    : "message does not match inputs.";

            platform.deletePost(postID);
            assert (platform.getTotalOriginalPosts()) == 0 : "number of posts registered in the system does not match";

        } catch (PostIDNotRecognisedException e) {
            assert (false) : "PostIDNotRecognisedException thrown incorrectly";
        } catch (HandleNotRecognisedException e) {
            assert (false) : "HandleNotRecognisedException thrown incorrectly";
        } catch (InvalidPostException e) {
            assert (false) : "InvalidPostException thrown incorrectly";
        }

        // Output a post as a string with both endorsements and comments.
        // check that not exceptions are thrown.
        try {
            Integer postID = platform.createPost("my_handle", "Hello.");
            assert (platform.getTotalOriginalPosts()) == 1 : "number of posts registered in the system does not match";

            Integer commentID1 = platform.commentPost("my_handle", postID, "Hi.");
            Integer commentID2 = platform.commentPost("my_handle", postID, "Hey.");
```

```
369
370             Integer epID1 = platform.endorsePost("my_handle", postID);
371             Integer epID2 = platform.endorsePost("my_handle", postID);
372
373             platform.showIndividualPost(postID);
374             assert (platform.showIndividualPost(postID).equals("ID: " + postID
375                     + "\n Account: my_handle\n No. endorsements: 2 | No. comments: 2\n Hello."))
== true
376                     : "message does not match inputs.";
377
378             platform.deletePost(postID);
379             assert (platform.getTotalOriginalPosts()) == 0 : "number of posts registered in the
system does not match";
380             assert (platform.getTotalEndorsmentPosts() == 0)
381                     : "number of endorsed posts registered in the system does not match";
382
383             platform.deletePost(commentID1);
384             assert (platform.getTotalCommentPosts() == 1)
385                     : "number of endorsed posts registered in the system does not match";
386             platform.deletePost(commentID2);
387             assert (platform.getTotalCommentPosts() == 0)
388                     : "number of endorsed posts registered in the system does not match";
389
390         } catch (PostIDNotRecognisedException e) {
391             assert (false) : "PostIDNotRecognisedException thrown incorrectly";
392         } catch (HandleNotRecognisedException e) {
393             assert (false) : "HandleNotRecognisedException thrown incorrectly";
394         } catch (InvalidPostException e) {
395             assert (false) : "InvalidPostException thrown incorrectly";
396         } catch (NotActionablePostException e) {
397             assert (false) : "NotActionablePostException thrown incorrectly";
398         }
399
400         // Create a post with message length greater than 100 characters
401         // Check that an invalidpost exception is thrown.
402         try {
403             Integer postID = platform.createPost("my_handle",
404                     "mqqlkaifyqrixjordaaorufwiruuvtkewyvsadje
pnoexviwoppkssps zmuehcblosibmgiqocwzksnhrlgqjqhgajuxfwrqixmjb");
405             assert (platform.getTotalOriginalPosts()) == 0 : "number of posts registered in the
system does not match";
406
407         } catch (HandleNotRecognisedException e) {
408             assert (false) : "HandleNotRecognisedException thrown incorrectly";
409         } catch (InvalidPostException e) {
410             assert (true);
411         }
412
413         // Create a post with a message length of 0 characters
414         // Check that an invalid post exception is thrown.
415         try {
416             Integer postID = platform.createPost("my_handle", "");
417             assert (platform.getTotalOriginalPosts()) == 0 : "number of posts registered in the
system does not match";
418
419         } catch (HandleNotRecognisedException e) {
420             assert (false) : "HandleNotRecognisedException thrown incorrectly";
421         } catch (InvalidPostException e) {
422             assert (true);
423         }
424
425         // Create a message of length 1 char
426         // Check that no exception is thrown.
427         try {
428             Integer postID = platform.createPost("my_handle", "1");
429             assert (platform.getTotalOriginalPosts()) == 1 : "number of posts registered in the
system does not match";
```

```
430
431         } catch (HandleNotRecognisedException e) {
432             assert (false) : "HandleNotRecognisedException thrown incorrectly";
433         } catch (InvalidPostException e) {
434             assert (false) : "InvalidPostException thrown incorrectly";
435         }
436
437         // Create a message of length 100 chars
438         // Check that no exception is thrown.
439         try {
440             Integer postID = platform.createPost("my_handle",
441                     "ntkfdnzeuygzhcjmgsxzqiadunpwaljcgxfhtlwo
yxobiunqucrnmtuuikodoqmkbjhsfwhbgcjvyputfevipftvxfrbkqivkrfv");
442             assert (platform.getTotalOriginalPosts()) == 2 : "number of posts registered in the
system does not match";
443
444         } catch (HandleNotRecognisedException e) {
445             assert (false) : "HandleNotRecognisedException thrown incorrectly";
446         } catch (InvalidPostException e) {
447             assert (false) : "InvalidPostException thrown incorrectly";
448         }
449
450         // Create a post, with 3 level 1 comments, with 3 level 1 comments, 2 level 2 comments and
1 level 3 comment
451         // Check that no exceptions are thrown.
452         try {
453             platform.erasePlatform();
454             Integer accountID1 = platform.createAccount("testAcc1", "my bio");
455             Integer accountID2 = platform.createAccount("testAcc2", "a bio");
456             Integer accountID3 = platform.createAccount("testAcc3", "the bio");
457             Integer accountID4 = platform.createAccount("testAcc4", "no bio");
458             Integer accountID5 = platform.createAccount("testAcc5", "what is this?");
459             assert (platform.getNumberOfAccounts()) == 5 : "Number of original posts in the
platform does not match";
460
461             Integer postID1 = platform.createPost("testAcc1", "post1");
462             assert (platform.getTotalOriginalPosts()) == 1 : "Number of original posts in the
platform does not match";
463
464             Integer commentID1 = platform.commentPost("testAcc2", postID1, "comment 1 msg");
465             Integer commentID2 = platform.commentPost("testAcc3", postID1, "comment 2 msg");
466             Integer commentID3 = platform.commentPost("testAcc4", postID1, "comment 3 msg");
467             Integer commentID4 = platform.commentPost("testAcc1", commentID1, "comment 4 msg");
468             Integer commentID5 = platform.commentPost("testAcc1", commentID2, "comment 5 msg");
469             Integer commentID6 = platform.commentPost("testAcc5", commentID3, "comment 6 msg");
470             Integer commentID7 = platform.commentPost("testAcc4", commentID5, "comment 7 msg");
471             Integer commentID8 = platform.commentPost("testAcc1", commentID6, "comment 8 msg");
472             Integer commentID9 = platform.commentPost("testAcc5", commentID8, "comment 9 msg");
473             assert (platform.getTotalCommentPosts()) == 9 : "Number of comments in the platform
does not match";
474
475         } catch (HandleNotRecognisedException e) {
476             assert (false) : "HandleNotRecognisedException thrown incorrectly";
477         } catch (InvalidPostException e) {
478             assert (false) : "InvalidPostException thrown incorrectly";
479         } catch (PostIDNotRecognisedException e) {
480             assert (false) : "PostIDNotRecognisedException thrown incorrectly";
481         } catch (NotActionablePostException e) {
482             assert (false) : "NotActionablePostException thrown incorrectly";
483         } catch (IllegalHandleException e) {
484             assert (false) : "IllegalHandleException thrown incorrectly";
485         } catch (InvalidHandleException e) {
486             assert (false) : "InvalidHandleException thrown incorrectly";
487         }
488
489         // Create a post tree containing a post, 3 level 1 comments, each has 1 level 2
490         // comment. Endorse each post once. Delete one of the level 1 comments.
```

```
491         // Check that no exceptions are thrown.
492         try {
493             platform.erasePlatform();
494             Integer accountID1 = platform.createAccount("acc1");
495             Integer accountID2 = platform.createAccount("acc2");
496             Integer accountID3 = platform.createAccount("acc3");
497             assert (platform.getNumberOfAccounts()) == 3 : "Number of accounts in the platform
    does not match";
498 
499             Integer postID1 = platform.createPost("acc1", "post1");
500             assert (platform.getTotalOriginalPosts()) == 1 : "Number of original posts in the
    platform does not match";
501 
502             Integer commentID1 = platform.commentPost("acc1", postID1, "acc1 comment 1");
503             Integer commentID2 = platform.commentPost("acc2", postID1, "acc1 comment 2");
504             Integer commentID3 = platform.commentPost("acc3", postID1, "acc1 comment 3");
505             Integer commentID4 = platform.commentPost("acc1", commentID2, "acc1 comment 4");
506             Integer commentID5 = platform.commentPost("acc2", commentID3, "acc1 comment 5");
507             Integer commentID6 = platform.commentPost("acc3", commentID1, "acc1 comment 6");
508             assert (platform.getTotalCommentPosts()) == 6 : "Number of comments in the platform
    does not match";
509 
510             Integer endorsedPostID2 = platform.endorsePost("acc2", commentID3);
511             Integer endorsedPostID1 = platform.endorsePost("acc2", commentID3);
512 
513             platform.deletePost(commentID3);
514             assert (platform.getTotalCommentPosts()) == 5 : "Number of comments in the platform
    does not match";
515 
516             //System.out.println(platform.showPostChildrenDetails(postID1));
517 
518         } catch (HandleNotRecognisedException e) {
519             assert (false) : "HandleNotRecognisedException thrown incorrectly";
520         } catch (InvalidPostException e) {
521             assert (false) : "InvalidPostException thrown incorrectly";
522         } catch (PostIDNotRecognisedException e) {
523             assert (false) : "PostIDNotRecognisedException thrown incorrectly";
524         } catch (NotActionablePostException e) {
525             assert (false) : "NotActionablePostException thrown incorrectly";
526         } catch (IllegalHandleException e) {
527             assert (false) : "IllegalHandleException thrown incorrectly";
528         } catch (InvalidHandleException e) {
529             assert (false) : "InvalidHandleException thrown incorrectly";
530         }
531 
532         // Create an account, create posts of that account then delete the account, the posts
    should be deleted too.
533         // Check that no exceptions are thrown and all assertions are correct.
534         try{
535             platform.erasePlatform();
536             Integer accID = platform.createAccount("anAccount");
537             assert(platform.getNumberOfAccounts()) == 1 : "Number of accounts in the platform does
    not match";
538 
539             Integer postID = platform.createPost("anAccount", "anAccount Post 1");
540             assert(platform.getTotalOriginalPosts()) == 1 : "Total original posts in the platform
    does not match";
541             Integer commentID = platform.commentPost("anAccount", postID, "anAccount Comment 1");
542             assert(platform.getTotalCommentPosts()) == 1 : "total comment posts in the platform
    does not match";
543             Integer epID = platform.endorsePost("anAccount", postID);
544             Integer epID2 = platform.endorsePost("anAccount", commentID);
545             assert (platform.getTotalEndorsmentPosts()) == 2 : "total endorsement posts in the
    platform does not match";
546 
547             platform.removeAccount(accID);
548             assert(platform.getNumberOfAccounts()) == 0 : "total number of accounts in the
    platform does not match";
```

```
549             assert(platform.getTotalOriginalPosts()) == 0 : "total original posts in the platform
    does not match";
550             assert(platform.getTotalCommentPosts()) == 0 : "total comment posts in the platform
    does not match";
551             assert (platform.getTotalEndorsmentPosts()) == 0 : "total endorsement posts in the
    platform does not match";
552
553
554         } catch (IllegalHandleException e) {
555             assert (false) : "The account handle input is invalid, the handle input is already in
    use.";
556         } catch (InvalidHandleException e) {
557             assert (false) : "the handle input is not valid, the handle must be of the correct
    length";
558         } catch (HandleNotRecognisedException e) {
559             assert (false) : "The handle input does not exist in the system.";
560         } catch (InvalidPostException e) {
561             assert (false) : "The post input is empty or exceeds the permitted message length";
562         } catch (PostIDNotRecognisedException e) {
563             assert (false) : "The post with the given id does not exist in the system";
564         } catch (NotActionablePostException e) {
565             assert (false) : "This action cannot be performed on the post with the given id";
566         } catch (AccountIDNotRecognisedException e) {
567             assert (false) : "The account id input does not exist in the system";
568         }
569
570         //Check that you cannot endorse an endorsed post
571         // Make sure not actionable post exception is thrown.
572         try{
573             platform.createAccount("testingAccount");
574             Integer postID = platform.createPost("testingAccount", "a message");
575             Integer epID = platform.endorsePost("testingAccount", postID);
576             platform.endorsePost("testingAccount", epID);
577
578
579         }  catch (HandleNotRecognisedException e) {
580             assert (false) : "HandleNotRecognisedException thrown incorrectly";
581         } catch (InvalidPostException e) {
582             assert (false) : "InvalidPostException thrown incorrectly";
583         } catch (PostIDNotRecognisedException e) {
584             assert (false) : "PostIDNotRecognisedException thrown incorrectly";
585         } catch (NotActionablePostException e) {
586             assert (true) : "NotActionablePostException thrown incorrectly";
587         } catch (IllegalHandleException e) {
588             assert (false) : "IllegalHandleException thrown incorrectly";
589         } catch (InvalidHandleException e) {
590             assert (false) : "InvalidHandleException thrown incorrectly";
591         }
592
593         // Check that endorsed posts are removed correctly from an account when an account is
    deleted.
594         try {
595             platform.erasePlatform();
596             Integer accID = platform.createAccount("aHandleToTest");
597             assert (platform.getNumberOfAccounts() == 1) : "number of accounts in the system does
    not match";
598
599             Integer postID = platform.createPost("aHandleToTest", "none");
600             assert (platform.getTotalOriginalPosts() == 1) : "number of original posts in the
    system does not match";
601
602             Integer epID1 = platform.endorsePost("aHandleToTest", postID);
603             Integer epID2 = platform.endorsePost("aHandleToTest", postID);
604             assert (platform.getTotalEndorsmentPosts() == 2) : "number of endorsement posts in the
    system does not match";
605
606             platform.deletePost(epID1);
607             assert (platform.getTotalEndorsmentPosts() == 1) : "number of endorsement posts in the
```

```
system does not match";

            platform.removeAccount("aHandleToTest");
            assert (platform.getNumberOfAccounts() == 0) : "number of accounts in the system does not match";
            assert (platform.getTotalOriginalPosts() == 0) : "number of original posts in the system does not match";
            assert (platform.getTotalEndorsmentPosts() == 0) : "number of endorsement posts in the system does not match";

        }  catch (HandleNotRecognisedException e) {
            assert (false) : "HandleNotRecognisedException thrown incorrectly";
        } catch (InvalidPostException e) {
            assert (false) : "InvalidPostException thrown incorrectly";
        } catch (PostIDNotRecognisedException e) {
            assert (false) : "PostIDNotRecognisedException thrown incorrectly";
        } catch (NotActionablePostException e) {
            assert (false) : "NotActionablePostException thrown incorrectly";
        } catch (IllegalHandleException e) {
            assert (false) : "IllegalHandleException thrown incorrectly";
        } catch (InvalidHandleException e) {
            assert (false) : "InvalidHandleException thrown incorrectly";
        }

        //Creates a handle of length 0 chars
        try {
            Integer accountID = platform.createAccount("");
        } catch (IllegalHandleException e) {
            assert (false) : "Illegal handle exception thrown incorrectly";
        } catch (InvalidHandleException e) {
            assert (true);
        }

        //Creates a handle of length 31 chars (too long)
        try {
            Integer accountID = platform.createAccount("abcdefghijklmnopqrstuvwxyz12345");
        } catch (IllegalHandleException e) {
            assert (false) : "Illegal handle exception thrown incorrectly";
        } catch (InvalidHandleException e) {
            assert (true);
        }

        //Creates 2 accounts with the same handle
        try {
            Integer accountID1 = platform.createAccount("testAccount");
            Integer accountID2 = platform.createAccount("testAccount");
        } catch (IllegalHandleException e) {
            assert (true);
        } catch (InvalidHandleException e) {
            assert (false) : "Invalid handle exception thrown incorrectly";
        }

        //Tests showing account
        try {
            Integer testAccountID = platform.createAccount("showingAccount");
            String accountDetails = platform.showAccount("showingAccount");

            assert (accountDetails.equals("ID: " + testAccountID + " \nHandle: showingAccount \nDescription:  \nPost Count: 0 \nEndorse Count: 0"));
        } catch (HandleNotRecognisedException e) {
            assert (false) : "Handle not recognised exception thrown incorrectly";
        }
        catch (IllegalHandleException e) {
            assert (false) : "Illegal handle exception thrown incorrectly";
        }
        catch (InvalidHandleException e) {
```

```
669             assert (false) : "Invalid handle exception thrown incorrectly";
670         }
671 
672         //Setup tests for loaded platform
673         // No exceptions should be thrown.
674         try {
675             platform.erasePlatform();
676             Integer accountID1 = platform.createAccount("acc1");
677             Integer accountID2 = platform.createAccount("acc2");
678             Integer accountID3 = platform.createAccount("acc3");
679             assert (platform.getNumberOfAccounts()) == 3 : "Number of accounts in the platform
does not match";
680 
681             Integer postID1 = platform.createPost("acc1", "post1");
682             assert (platform.getTotalOriginalPosts()) == 1 : "Number of original posts in the
platform does not match";
683 
684             Integer commentID1 = platform.commentPost("acc1", postID1, "acc1 comment 1");
685             Integer commentID2 = platform.commentPost("acc2", postID1, "acc1 comment 2");
686             Integer commentID3 = platform.commentPost("acc3", postID1, "acc1 comment 3");
687             Integer commentID4 = platform.commentPost("acc1", commentID2, "acc1 comment 4");
688             Integer commentID5 = platform.commentPost("acc2", commentID3, "acc1 comment 5");
689             Integer commentID6 = platform.commentPost("acc3", commentID1, "acc1 comment 6");
690             assert (platform.getTotalCommentPosts()) == 6 : "Number of comments in the platform
does not match";
691 
692             Integer endorsedPostID2 = platform.endorsePost("acc2", commentID3);
693             Integer endorsedPostID1 = platform.endorsePost("acc2", commentID3);
694 
695             platform.deletePost(commentID3);
696             assert (platform.getTotalCommentPosts()) == 5 : "Number of comments in the platform
does not match";
697 
698             //System.out.println(platform.showPostChildrenDetails(postID1));
699 
700         } catch (HandleNotRecognisedException e) {
701             assert (false) : "HandleNotRecognisedException thrown incorrectly";
702         } catch (InvalidPostException e) {
703             assert (false) : "InvalidPostException thrown incorrectly";
704         } catch (PostIDNotRecognisedException e) {
705             assert (false) : "PostIDNotRecognisedException thrown incorrectly";
706         } catch (NotActionablePostException e) {
707             assert (false) : "NotActionablePostException thrown incorrectly";
708         } catch (IllegalHandleException e) {
709             assert (false) : "IllegalHandleException thrown incorrectly";
710         } catch (InvalidHandleException e) {
711             assert (false) : "InvalidHandleException thrown incorrectly";
712         }
713 
714     }
715 
716     private static void testLoadedPlatform(SocialMediaPlatform platform) {
717         //Check that the platform has the same number of posts/accounts as the previous platform.
718         assert (platform.getNumberOfAccounts()) == 3 : "Number of accounts in the platform does
not match";
719         assert (platform.getTotalOriginalPosts()) == 1 : "Number of original posts in the platform
does not match";
720         assert (platform.getTotalCommentPosts()) == 5 : "Number of comments in the platform does
not match";
721         assert (platform.getTotalEndorsmentPosts()) == 0 : "Number of endorsement posts in the
platform does not match";
722 
723 
724     }
725 
726 
727 
```

```
728
729 }
730
```