```java
package socialmedia;

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

public class Post implements Serializable  {

    //class variables
    private static int idCounter = 0;

    private int endorsementCount;
    private int commentCount;

    private final String message;
    private final int id;
    private final Account account;
    private final PostType postType;

    //static list of all posts across all accounts
    public static List<Post> posts = new ArrayList<>();
    //list of all endorsements associated with this post
    public List<Endorsement> endorsements = new ArrayList<>();
    //list of all comments associated with this post
    public List<Comment> comments = new ArrayList<>();

    //constructor
    public Post(String message, Account account, PostType postType) {
        this.message = message;
        this.id = idCounter++;
        this.account = account;
        this.postType = postType;
        this.endorsementCount = 0;
        this.commentCount = 0;

    }
```

```
37
38      //endorsement related getters and setters
39      public void addEndorsement(Endorsement
    endorsement) {
40          endorsements.add(endorsement);
41          posts.add(endorsement);
42          endorsementCount++;
43      }
44      public void removeEndorsement() {
45          //dont need to reference the endorsement
    because it is already in the list of endorsements
46          //if we remove it from the list here it
    creates a concurrent modification exception
47          endorsementCount--;
48      }
49      public List<Endorsement> getEndorsements() {
50          return endorsements;
51      }
52      public int getEndorsementCount() {
53          return endorsementCount;
54      }
55
56      //comment related getters and setters
57      public void addComment(Comment comment) {
58          comments.add(comment);
59          posts.add(comment);
60          commentCount++;
61      }
62      public void removeComment() {
63          //dont need to reference the comment because
    it is already in the list of comments
64          //if we remove it from the list here it
    creates a concurrent modification exception
65          commentCount--;
66      }
67      public List<Comment> getComments() {
68          return comments;
69      }
70      public int getCommentCount() {
71          return commentCount;
72      }
```

```java

    //generic getters
    public int getID() {
        return id;
    }
    public PostType getPostType() {
        return postType;
    }
    public String getMessage() {
        return message;
    }
    public String getHandle() {
        return account.getHandle();
    }
    public int getAccountId() {
        return account.getId();
    }
    public Account getAccount() {
        return account;
    }
    public static void resetCounter() {
        idCounter = 0;
    }
}

```

```java
package socialmedia;

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

public class Account implements Serializable {

    private static int idCounter = 0;

    private String handle;
    private String description;
    private final int id;
    private int postCount;

    public static List<Account> accounts = new ArrayList<>();

    public Account(String handle) {
        this.handle = handle;
        this.description = "";
        this.id = idCounter++;
        this.postCount = 0;
    }
    public Account(String handle, String description) {
        this.handle = handle;
        this.description = description;
        this.id = accounts.size();
        this.postCount = 0;
    }

    public String getDescription() {
        return description;
    }
    public void setDescription(String description) {
        this.description = description;
    }
    public String getHandle() {
        return handle;
    }
```

```
40      public void setHandle(String handle) {
41          this.handle = handle;
42      }
43      public int getId() {
44          return id;
45      }
46      public int getPostCount() {
47          return postCount;
48      }
49      public void addPost() {
50          this.postCount++;
51      }
52      public void removePost() {
53          this.postCount--;
54      }
55      public static void resetPlatform() {
56
57          for (Post post : Post.posts) {
58              post.endorsements.clear();
59              post.comments.clear();
60          }
61          Post.posts.clear();
62          Post.resetCounter();
63          idCounter = 0;
64          accounts.clear();
65
66      }
67  }
68
```

```java
package socialmedia;

public class Comment extends Post {

    private Post post; // the post that this comment is associated with

    public Comment(String message, Account account, Post post) {
        //the message of the comment
        //the account that made the comment
        //the post that the comment is associated with

        super(message, account, PostType.COMMENT); // call the constructor of the parent class
        this.post = post; // set the post that this comment is associated with
        post.addComment(this); // add this comment to the list of comments of the post
        }

    public Post getPost() {
        return post;
        //returns the post that this comment is associated with
    }
    public void setPost(Post post) {
        this.post = post;
        //sets the post that this comment is associated with
        //this is only used when the post is deleted
    }

}
```

```java
package socialmedia;

public enum PostType {
    COMMENT, POST, ENDORSEMENT, NULL
}

```

```java
package socialmedia;

public class Endorsement extends Post {

    private final Post post; // the post that this endorsement is associated with

    public Endorsement(String message, Account account, Post post) {
        //the message of the endorsement
        //the account that made the endorsement
        //the post that the endorsement is associated with

        //call the constructor of the parent class
        super(message, account, PostType.ENDORSEMENT);
        this.post = post;
        post.addEndorsement(this);
    }

    public Post getPost() {
        return post;
    }

}


```

```java
package socialmedia;

import java.io.IOException;
import java.util.Iterator;
import java.util.List;
import java.util.Objects;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;

public class SocialMedia implements SocialMediaPlatform {
    /**
     The users, i.e., accounts registered in the system, will be able to post original messages
     and comments, which can also be endorsed by endorsement posts.
     Accounts have a unique numerical identifier, but also a unique string handle to be more easily identified throughout
     the system. They can have a description field to add personal information they want to share. Posts ( original,
     comments, and endorsements) have a unique numerical identifier and contain a message with up to 100-characters.
     The post ID is a sequential number such that its ordering is a proxy for a post’s chronology. They are always associated
     with an author, i.e., the account who posted it . To allow the creation of meaningful conversation trees, posts must keep
     track of the list of endorsements and comments they received. Endorsements and comments are also categorized as
     posts, but with special features. For instance, comments always have to point to another post ( original or comment).
     Endorsements automatically replicate the endorsed message and also refer to original or
```

```
23 comment posts. Endorsements
24      are not endorsed or commented. The system should
    provide basic analytics such as the most popular
   post and the
25      most popular account.
26      */
27
28     @Override
29     public int createAccount(String handle) throws
   IllegalHandleException, InvalidHandleException {
30
31         //throwing exceptions if the handle is
   invalid
32
33         //if the handle hasn't been specified
34         if (handle == null) {
35             throw new InvalidHandleException("Invalid
    handle");
36         }
37         //if the handle is an empty string
38         if (handle.isEmpty()){
39             throw new InvalidHandleException("No
   handle given");
40         }
41         //if the handle length is greater than 30
42         if(handle.length() > 30){
43             throw new InvalidHandleException("Handle
   too long");
44         }
45         //if the handle contains any whitespace
46         if(handle.contains(" ")){
47             throw new InvalidHandleException("Handle
   contains spaces");
48         }
49
50         //checks for duplicate handle identifiers in
   all existing accounts
51         for (Account account : Account.accounts) {
52             //if the handle already exists throw an
   exception
53              if (account.getHandle().equals(handle
```

```
53 )) {
54                     throw new IllegalHandleException("
   Handle already exists");
55                 }
56             }
57 
58             //creating the new account
59             Account newAccount = new Account(handle);
60             Account.accounts.add(newAccount);
61 
62             //returning the new account's ID
63             return newAccount.getId();
64         }
65 
66         @Override
67         public int createAccount(String handle, String
   description) throws IllegalHandleException,
   InvalidHandleException {
68 
69             //throwing exceptions if the handle is
   invalid
70 
71             //if the handle hasn't been specified
72             if (handle == null) {
73                 throw new InvalidHandleException("Invalid
    handle");
74             }
75             //if the handle is an empty string
76             if (handle.isEmpty()){
77                 throw new InvalidHandleException("No
   handle given");
78             }
79             //if the handle length is greater than 30
80             if(handle.length() > 30){
81                 throw new InvalidHandleException("Handle
   too long");
82             }
83             //if the handle contains any whitespace
84             if(handle.contains(" ")){
85                 throw new InvalidHandleException("Handle
   contains spaces");
```

```
 86          }
 87
 88          //checks for duplicate handle identifiers in
     all existing accounts
 89          for (Account account : Account.accounts) {
 90              //if the handle already exists throw an
    exception
 91              if (account.getHandle().equals(handle
    )) {
 92                  throw new IllegalHandleException("
    Handle already exists");
 93              }
 94          }
 95
 96          //creating the new account
 97          Account newAccount = new Account(handle);
 98          Account.accounts.add(newAccount);
 99
100          //returning the new account's ID
101          return newAccount.getId();
102
103      }
104
105      @Override
106      public void removeAccount(int id) throws
    AccountIDNotRecognisedException {
107
108          //search through all the accounts and find
    the one with the id
109          //remove all the comments the account has
    made, replace the comments with a generic response
110          //remove all the endorsements the post has
    made
111          //replace the comments on the posts with a
    generic response
112          //remove the post from the account
113          //then remove the account
114
115
116          boolean found = false;
117          //checking through all existing accounts
```

```
118         for (Account account: Account.accounts) {
119             if (Objects.equals(account.getId(), id
    )) {
120                 //make sure the account exists so we
     can remove it and all its posts, comments and
    endorsements
121                 //delete the posts in the same way
    which the DeletePost method does
122
123                 //Iterator is used so that if a post
     is deleted the pointer doesn't lose its sequential
    position
124                 //loops though existing posts
125                 Iterator<Post> postIterator = Post.
    posts.iterator();
126                 while (postIterator.hasNext()) {
127                     Post post = postIterator.next();
128                     if (Objects.equals(post.getID
    (), id)) {
129
130                         //if there are any comments
    on this post
131                         //set the post that the
    comments refer to a generic empty post
132                         for (Comment comment: post.
    comments) {
133                             //replace any comments
    on this post with a generic response
134                             comment.setPost(new Post
    ("The original content was removed from the system
    and is no longer available.", null, PostType.NULL));
135                         }
136
137                         //remove the endorsements if
     there are any
138                         Iterator<Endorsement>
    endorsementIterator = post.getEndorsements().
    iterator();
139
140                         while (endorsementIterator.
    hasNext()) {
```

```
141                                 Endorsement endorsement
     = endorsementIterator.next();
142                                 endorsement.getAccount
    ().removePost(); //decrements 1  from the account
    endorsement counter which posted
143                                 endorsementIterator.
    remove(); //remove the endorsement using the
    iterator
144                             }
145
146                             post.getAccount().removePost
    (); //decrements 1 from the number of posts on the
    account that posted it
147                             postIterator.remove(); //
    remove the post using the iterator
148                         }
149
150                         //remove all the accounts
    comments on any post in the system
151                         //so if this account has
    commented on a post that another account has made
152
153                         //use an iterator to remove the
    comments
154                         Iterator<Comment>
    commentIterator = post.comments.iterator();
155                         while (commentIterator.hasNext
    ()) {
156                             Comment comment =
    commentIterator.next();
157                             if (Objects.equals(comment.
    getID(), id)) {
158                                 //replace any comments
    on this accounts comment with a generic response
159                                 for (Comment comment2:
    comment.comments) {
160                                     comment2.setPost(new
     Post("The original content was removed from the
    system and is no longer available.", null, PostType.
    NULL));
161                                 }
```

```
162
163                             //use an iterator to
    remove the endorsements
164                             Iterator<Endorsement>
    endorsementIterator = comment.getEndorsements().
    iterator();
165                             while (
    endorsementIterator.hasNext()) {
166                                 Endorsement
    endorsement = endorsementIterator.next();
167                                 endorsement.
    getAccount().removePost(); //remove a count of 1
    from the account which posted
168                                 endorsementIterator.
    remove(); //remove the endorsement using the
    iterator
169                             }
170
171                             //remove the comment
172                             //we don't need to
    remove a count from the account which posted the
    comment as it is being deleted anyway
173                             post.removeComment();
174                             commentIterator.remove
    (); //remove the comment using the iterator
175                         }
176                     }
177
178                     //remove all endorsements by
    this account on any post in the system
179                     //using an iterator to remove
    the endorsements
180                     Iterator<Endorsement>
    endorsementIterator = post.endorsements.iterator();
181                     while (endorsementIterator.
    hasNext()) {
182                         Endorsement endorsement =
    endorsementIterator.next();
183                         if (Objects.equals(
    endorsement.getID(), id)) {
184                             post.removeEndorsement
```

```
184 (); //remove a count of 1 from the posts'
    endorsement count
185                                 endorsementIterator.
    remove(); //remove the endorsement using the
    iterator
186                             }
187                         }
188                 }
189                 // removes the account
190                 Account.accounts.remove(account);
191                 found = true;
192                 break; // break now we have found
    the account and removed it
193             }
194         }
195
196         //if the account was not found throw an
    exception
197         if (!found) {
198             throw new
    AccountIDNotRecognisedException("Account not found"
    );
199         }
200     }
201
202     @Override
203     public void removeAccount(String handle) throws
    HandleNotRecognisedException {
204
205         // same as above but search with handle
    rather than id
206         boolean found = false;
207         for (Account account: Account.accounts) {
208             if (Objects.equals(account.getHandle(),
    handle)) {
209                 //make sure the account exists so we
     can remove it and all its posts, comments and
    endorsements
210                 //delete the posts in the same way
    which the DeletePost method does
211
```

```
212                 Iterator<Post> postIterator = Post.
    posts.iterator();
213                 while (postIterator.hasNext()) {
214                     Post post = postIterator.next();
215                     if (Objects.equals(post.
    getHandle(), handle)) {
216
217                         //if there are any comments
    on this post
218                         //set the post that the
    comments refer to to a generic empty post
219                         for (Comment comment: post.
    comments) {
220                             //replace any comments
    on this post with a generic response
221                             comment.setPost(new Post
    ("The original content was removed from the system
    and is no longer available.", null, PostType.NULL));
222                         }
223
224                         //remove the endorsements if
     there are any
225                         Iterator<Endorsement>
    endorsementIterator = post.getEndorsements().
    iterator();
226
227                         while (endorsementIterator.
    hasNext()) {
228                             Endorsement endorsement
     = endorsementIterator.next();
229                             endorsement.getAccount
    ().removePost(); //remove a count of 1 from the
    account which posted
230                             endorsementIterator.
    remove(); //remove the endorsement using the
    iterator
231                         }
232
233                         //remove the post
234                         post.getAccount().removePost
    (); //minus 1 from the number of posts on the
```

```
234 account that posted it
235                         postIterator.remove(); //
    remove the post using the iterator
236                     }
237
238                     //remove all the accounts
    comments on any post in the system
239                     //so if this account has
    commented on a post that another account has made
240
241                     //use an iterator to remove the
    comments
242                     Iterator<Comment>
    commentIterator = post.comments.iterator();
243                     while (commentIterator.hasNext
    ()) {
244                         Comment comment =
    commentIterator.next();
245                         if (Objects.equals(comment.
    getHandle(), handle)) {
246                             //replace any comments
    on this accounts comment with a generic response
247                             for (Comment comment2:
    comment.comments) {
248                                 comment2.setPost(new
     Post("The original content was removed from the
    system and is no longer available.", null, PostType.
    NULL));
249                             }
250
251                             //use an iterator to
    remove the endorsements
252                             Iterator<Endorsement>
    endorsementIterator = comment.getEndorsements().
    iterator();
253                             while (
    endorsementIterator.hasNext()) {
254                                 Endorsement
    endorsement = endorsementIterator.next();
255                                 endorsement.
    getAccount().removePost(); //remove a count of 1
```

```
255 from the account which posted
256                                 endorsementIterator.
    remove(); //remove the endorsement using the
    iterator
257                                 }
258
259                                 //remove the comment
260                                 //we don't need to
    remove a count from the account which posted the
    comment as it is being deleted anyway
261                                 post.removeComment();
262                                 commentIterator.remove
    (); //remove the comment using the iterator
263                             }
264                         }
265
266                         //remove all endorsements by
    this account on any post in the system
267                         //using an iterator to remove
    the endorsements
268                         Iterator<Endorsement>
    endorsementIterator = post.endorsements.iterator();
269                         while (endorsementIterator.
    hasNext()) {
270                             Endorsement endorsement =
    endorsementIterator.next();
271                             if (Objects.equals(
    endorsement.getHandle(), handle)) {
272                                 post.removeEndorsement
    (); //remove a count of 1 from the posts'
    endorsement count
273                                 endorsementIterator.
    remove(); //remove the endorsement using the
    iterator
274                             }
275                         }
276                 }
277                 // removes the account
278                 Account.accounts.remove(account);
279                 found = true;
280                 break; // break now we have found
```

```
280 the account and removed it
281             }
282         }
283 
284         //if the account was not found throw an
    exception
285         if (!found) {
286             throw new HandleNotRecognisedException("
    Handle not found");
287         }
288     }
289 
290     @Override
291     public void changeAccountHandle(String oldHandle
    , String newHandle)
292             throws HandleNotRecognisedException,
    IllegalHandleException, InvalidHandleException {
293 
294         //search through all the accounts and find
    the one with the handle
295         //then update the handle
296         boolean found = false;
297 
298         //check if new handle is valid
299         if (newHandle == null) {
300             throw new InvalidHandleException("
    Invalid handle");
301         }
302         if (newHandle.isEmpty()){
303             throw new InvalidHandleException("No
    handle given");
304         }
305         if(newHandle.length() > 30){
306             throw new InvalidHandleException("Handle
     too long");
307         }
308         if(newHandle.contains(" ")){
309             throw new InvalidHandleException("Handle
     contains spaces");
310         }
311 
```

```
312         //check if the new handle already exists
313         for (Account account : Account.accounts) {
314             if (account.getHandle().equals(newHandle
    )) {
315                 throw new IllegalHandleException("
    Handle already exists");
316             }
317         }
318 
319 
320         //update the handle
321         //search through all the accounts and find
    the one with the current handle
322         for (Account account: Account.accounts) {
323             if (account.getHandle().equals(oldHandle
    )) {
324                 //we set the handle and then break
    from the loop
325                 account.setHandle(newHandle);
326                 found = true;
327                 break; //we can stop looking now we
    have found it
328             }
329         }
330         if (!found) {
331             throw new HandleNotRecognisedException("
    cant find handle");
332         }
333     }
334 
335     @Override
336     public void updateAccountDescription(String
    handle, String description) throws
    HandleNotRecognisedException {
337 
338         //update the description of the account
339         //search through all the accounts and find
    the one with the handle
340         //then update the description
341         boolean found = false;
342 
```

```
343         for (Account account: Account.accounts) {
344             if (account.getHandle().equals(handle
    )) {
345                 account.setDescription(description);
346                 found = true;
347                 break; //we can stop looking now we
    have found it
348             }
349         }
350         if (!found) {
351             throw new HandleNotRecognisedException("
    Cant find handle");
352         }
353 
354     }
355 
356     @Override
357     public String showAccount(String handle) throws
    HandleNotRecognisedException {
358 
359         //search through all the accounts and find
    the one with the handle
360         //then create the string with all details of
     the account
361 
362         boolean found = false;
363         //creating the string to return and a string
     builder to make it
364         String accountDetails;
365         StringBuilder sb = new StringBuilder();
366 
367         //search through all the posts and if it is
    by our account add the endorsement count to the
    total
368         int endorseCount = 0;
369         for (Post post: Post.posts) {
370             if (post.getHandle().equals(handle)) {
371                 endorseCount += post.
    getEndorsementCount();
372             }
373         }
```

```
374
375         //search through all the accounts and find
    the one with the handle
376         for (Account account: Account.accounts) {
377             if (account.getHandle().equals(handle
    )) {
378
379                 //once we have found the right
    account we can create the string
380                 sb.append("ID: [").append(account.
    getId()).append("] \n");
381                 sb.append("Handle: [").append(
    account.getHandle()).append("] \n");
382
383                 if (Objects.equals(account.
    getDescription(), "")) {
384                     //if the description is empty we
     add a message saying there is no description
385                     sb.append("Description: No
    Description \n");
386                 } else {
387                     //otherwise, we add it to the
    string
388                     sb.append("Description: [").
    append(account.getDescription()).append("] \n");
389                 }
390
391                 sb.append("Post count: [").append(
    account.getPostCount()).append("] \n"); //number of
    posts
392                 sb.append("Endorse count: [").append
    (endorseCount).append("] \n"); //number of
    endorsements the account has received
393                 found = true;
394                 break; //we can stop looking now we
    have found it
395             }
396         }
397         if (!found) {
398             throw new HandleNotRecognisedException("
    Cant find handle");
```

```
399         }
400         //assemble the string builder into a string
    and return it
401         accountDetails = sb.toString();
402         return accountDetails;
403     }
404 
405     @Override
406     public int createPost(String handle, String
    message) throws HandleNotRecognisedException,
    InvalidPostException {
407 
408         int postID = 0;
409         boolean found = false;
410 
411         //check if the message is valid
412         if (message == null || message.isEmpty()) {
413             throw new InvalidPostException("Message
    is empty");
414         }
415         if (message.length() > 100) {
416             throw new InvalidPostException("Message
    is too long");
417         }
418 
419         //search through all the accounts and find
    the one with the handle
420         //if it does not exist throw an exception
421         for (Account account: Account.accounts) {
422             if (account.getHandle().equals(handle
    )) {
423                 Post post = new Post(message,
    account, PostType.POST); //create a new post
424                 Post.posts.add(post); //add the post
     to the list of posts
425 
426                 account.addPost(); //add one to the
    number of posts for the account
427                 postID = post.getID(); //get the id
    of the new post
428 
```

```
429                 found = true;
430                 break; //we can stop looking now we
    have found it
431             }
432         }
433
434         if (!found) {
435             throw new HandleNotRecognisedException("
    Cant find account");
436         }
437
438         //return the id of the new post
439         return postID;
440     }
441
442     @Override
443     public int endorsePost(String handle, int id)
444             throws HandleNotRecognisedException,
    PostIDNotRecognisedException,
    NotActionablePostException{
445
446         boolean found = false;
447         int postID = 0;
448         Account endorsingAccount = null;
449
450         //search through all the accounts and find
    the one with the handle then save it so that we can
    associate it with the new post
451         for (Account account: Account.accounts) {
452             if (account.getHandle().equals(handle
    )) {
453                 endorsingAccount = account;
454                 found = true;
455                 break; //we can stop looking now
    know the account trying to endorse exists
456             }
457         }
458
459         if (!found) {
460             throw new HandleNotRecognisedException("
    Cant find handle");
```

```
461         }
462
463         //search through all the posts and find the
    one with the id
464         //if the post we are trying to endorse is an
     endorsement post then throw an exception
465         found = false;
466         for (Post post: Post.posts) {
467             if (post.getID() == id) {
468                 if (post.getPostType() == PostType.
    ENDORSEMENT) {
469                     throw new
    NotActionablePostException("Cant endorse an
    endorsement");
470                 }
471                 Endorsement newEndorsement = new
    Endorsement(
472                         "EP@" +
473                         post.getHandle() +
474                         ": " +
475                         post.getMessage(),
476                         endorsingAccount,
477                         post); //create a new post
478
479                 endorsingAccount.addPost(); // add 1
     to the count of posts for the account
480                 postID = newEndorsement.getID(); //
    get the id of the new endorsement post
481                 found = true;
482                 break; //we can stop looking now
    know the post exists
483             }
484         }
485
486         if (!found) {
487             throw new PostIDNotRecognisedException("
    Cant find post");
488         }
489
490         return postID;
491     }
```

```
492
493     @Override
494     public int commentPost(String handle, int id,
    String message) throws HandleNotRecognisedException,
495             PostIDNotRecognisedException,
    NotActionablePostException, InvalidPostException {
496
497         boolean found = false;
498         int postID = 0;
499         Account commentingAccount = null;
500
501         //check the comment message is valid
502         if (message == null || message.isEmpty()) {
503             throw new InvalidPostException("Message
    is empty");
504         }
505         if (message.length() > 100) {
506             throw new InvalidPostException("Message
    is too long");
507         }
508
509         for (Account account: Account.accounts) {
510             if (account.getHandle().equals(handle
    )) {
511                 commentingAccount = account;
512                 found = true;
513                 break; //we can stop looking now
    know the account trying to endorse exists
514             }
515         }
516         if (!found) {
517             throw new HandleNotRecognisedException("
    Cant find handle");
518         }
519
520         //search through all the posts and find the
    one with the id
521
522         found = false;
523
524         for (Post post: Post.posts) {
```

```
525             if (post.getID() == id) {
526                 if (post.getPostType() == PostType.
    ENDORSEMENT) {
527                     throw new
    NotActionablePostException("Cant comment an
    endorsement");
528                 }
529                 //make a new comment on this post
530                 Comment newComment = new Comment(
    message, commentingAccount, post);
531 
532                 commentingAccount.addPost(); //add 1
     to the number of posts
533                 postID = newComment.getID(); //get
    the id of the new post
534                 found = true;
535                 break; //we can stop looking now
    know the post exists
536             }
537         }
538 
539         if (!found) {
540             throw new PostIDNotRecognisedException("
    Cant find post");
541         }
542 
543         return postID;
544     }
545 
546     @Override
547     public void deletePost(int id) throws
    PostIDNotRecognisedException {
548 
549         //search through all the posts and find the
    one with the id
550         boolean found = false;
551         for (Post post: Post.posts) {
552             if (post.getID() == id) {
553                 //remove all the endorsements
554                 //use an iterator to remove all the
    endorsements
```

```
555                 Iterator<Endorsement>
    endorsementIterator = post.endorsements.iterator();
556                 while (endorsementIterator.hasNext
    ()) {
557                     Endorsement endorsement =
    endorsementIterator.next();
558                     endorsement.getAccount().
    removePost(); //remove a count of 1 from the account
     which posted
559                     endorsementIterator.remove(); //
    remove the endorsement
560                 }
561                 //change the post the comments refer
     to to a generic empty post
562                 for (Comment comment: post.comments
    ) {
563                     comment.setPost(new Post("The
    original content was removed from the system and is
    no longer available.", null, PostType.NULL));
564                 }
565
566                 post.getAccount().removePost(); //
    minus 1 from the number of posts on the account that
     posted it
567
568                 Post.posts.remove(post); //remove
    the post
569                 found = true;
570                 break; //we can stop looking now
    know the post exists
571                 }
572         }
573
574         if (!found) {
575             throw new PostIDNotRecognisedException("
    Cant find post");
576         }
577     }
578
579     @Override
580     public String showIndividualPost(int id) throws
```

```
580 PostIDNotRecognisedException {
581
582         //create a string builder to build the
    string and initialise an empty string
583         StringBuilder sb = new StringBuilder();
584         String postDetails;
585
586
587         boolean found = false;
588
589         for (Post post: Post.posts) {
590             if (post.getID() == id) {
591                 sb.append("ID: [").append(post.getID
    ()).append("] \n");
592                 sb.append("Account: [").append(post.
    getAccount().getHandle()).append("] \n");
593                 sb.append("No. endorsements: [").
    append(post.getEndorsementCount()).append("] | No.
    comments: [").append(post.getCommentCount()).append(
    "] \n");
594                 sb.append(post.getMessage());
595                 found = true;
596                 break; //we can stop looking now we
    have finished building the string
597             }
598         }
599         if (!found) {
600             throw new PostIDNotRecognisedException("
    Cant find post");
601         }
602         postDetails = sb.toString();
603         return postDetails;
604     }
605
606     @Override
607     public StringBuilder showPostChildrenDetails(int
     id)
608             throws PostIDNotRecognisedException,
    NotActionablePostException {
609
610         StringBuilder sb = new StringBuilder(); //
```

```
610 create the stringbuilder
611         boolean found = false;
612 
613         for (Post currentPost: Post.posts) { //loop
    through all the posts to search for the one with our
     id
614             if (currentPost.getID() == id) {
615                 found = true; //we found the post so
     we no longer need to throw the exception
616 
617                 if (currentPost.getPostType() ==
    PostType.ENDORSEMENT) {
618                     //if the post is an endorsement
    then we cant show the children
619                     throw new
    NotActionablePostException("Endorsement posts do not
     have children");
620                 }
621 
622                 //start calling the childrenDetails
    method
623                 sb = childrenDetails(currentPost, 0
    , sb);
624             }
625         }
626         if (!found) {
627             throw new PostIDNotRecognisedException("
    Cant find post");
628         }
629 
630         //for all the comments on the post call
    showPostChildrenDetails recursively
631         //return the string builder
632         return sb;
633 
634     }
635 
636     public StringBuilder childrenDetails(Post
    currentPost, int indentLevel, StringBuilder sb) {
637 
638         int postID = currentPost.getID();
```

```
639         if (indentLevel == 0) {
640             //if the post is a top level post then
    we dont need to add the indent
641             sb.append("ID: ").append(postID).append(
    "\n");
642         } else {
643             //otherwise we indent it
644             sb.append("    ".repeat(indentLevel)).
    append("| \n");
645             sb.append("    ".repeat(indentLevel)).
    append("| > ID: ").append(postID).append("\n");
646         }
647
648         //add the account handle, number of
    endorsements, number of comments and the message
649         sb.append("    ".repeat(indentLevel)).append
    ("Account: ").append(currentPost.getAccount().
    getHandle()).append("\n");
650         sb.append("    ".repeat(indentLevel)).append
    ("No. endorsements: ").append(currentPost.
    getEndorsementCount()).append(" | No. comments: ").
    append(currentPost.getCommentCount()).append("\n");
651         sb.append("    ".repeat(indentLevel)).append
    (currentPost.getMessage()).append("\n");
652
653         //if the post has comments then we call the
    method recursively on each of them
654         if (currentPost.getCommentCount() > 0) {
655             for (Post comment : currentPost.comments
    ) {
656                 //for each comment we call the
    method again with the indent level increased by 1
657                 childrenDetails(comment, indentLevel
     + 1, sb);
658             }
659         }
660
661         //return the string builder
662         return sb;
663     }
664
```

```
665     @Override
666     public int getNumberOfAccounts() {
667 
668         int count = 0;
669         //increments counter for each account found
670         for (Account ignored : Account.accounts) {
671             count ++;
672         }
673         return count;
674     }
675 
676     @Override
677     public int getTotalOriginalPosts() {
678 
679         int total = 0;
680 
681         for (Post post: Post.posts) {
682             //for each post we check if it has a
    PostType of POST and only POST
683             //If so, add 1 to the total
684             if (post.getPostType() == PostType.POST
    ) {
685                 total++;
686             }
687         }
688         return total;
689     }
690 
691     @Override
692     public int getTotalEndorsmentPosts() {
693 
694         int total = 0;
695 
696         for (Post post: Post.posts) {
697             //for each post we check if it has a
    PostType of ENDORSEMENT and if it does we add 1 to
    the total
698             if (post.getPostType() == PostType.
    ENDORSEMENT) {
699                 total++;
700             }
```

```
701         }
702         return total;
703     }
704 
705     @Override
706     public int getTotalCommentPosts() {
707 
708         int total = 0;
709 
710         for (Post post: Post.posts) {
711             //for each post we check if it has a PostType of COMMENT and if it does we add 1 to the total
712             if (post.getPostType() == PostType.COMMENT) {
713                 total++;
714             }
715         }
716         return total;
717     }
718 
719     @Override
720     public int getMostEndorsedPost() {
721 
722         int currentMax = 0;
723         int currentMaxID = 0;
724 
725         for (Post post: Post.posts) {
726             //loop through each post and check if the endorsement count is greater than the current max
727             if (post.getEndorsementCount() > currentMax) {
728                 //if it is then set the current max to the endorsement count and set the current max id to the post id
729                 currentMax = post.getEndorsementCount();
730                 currentMaxID = post.getID();
731 
732             }
```

```
733         }
734         return currentMaxID;
735     }
736 
737     @Override
738     public int getMostEndorsedAccount() {
739 
740 
741         int currentMax = 0;
742         int currentMaxID = 0;
743         int total = 0;
744 
745         //checks for each account
746         for (Account account: Account.accounts) {
747             //checks for each post attributed to the
     account
748             for (Post post: Post.posts) {
749                 //increments endorsement counter
    only when endorsement is of correct type
750                 if (post.getPostType() != PostType.
    ENDORSEMENT && post.getAccountId() == account.getId
    ()) {
751                     total += post.
    getEndorsementCount();
752                 }
753             }
754             //updates the total
755             if (total > currentMax) {
756                 currentMax = total;
757                 currentMaxID = account.getId();
758             }
759         }
760         return currentMaxID;
761     }
762 
763     @Override
764     public void erasePlatform() {
765 
766         //erases all internal counters in social
    media
767         Account.resetPlatform();
```

```
768
769     }
770
771     @Override
772     public void savePlatform(String filename) throws
     IOException {
773
774         ObjectOutputStream oos = new
    ObjectOutputStream(new FileOutputStream(filename));
775
776             //writing the account list object to the
     byte stream
777             try {
778                 //only need to serialize post and
    account objects since endorsement and comments
    extend post
779                 //using Post.posts and Account.
    account ArrayList of objects to serialize all the
    included objects
780                 oos.writeObject(Post.posts);
781                 oos.writeObject(Account.accounts);
782             } catch (IOException e) {
783                 e.printStackTrace();
784                 throw new IOException("Could not
    serialize objects correctly");
785             }
786
787             //closes the object output stream
788             oos.close();
789
790     }
791
792
793     @Override
794     public void loadPlatform(String filename) throws
     IOException, ClassNotFoundException {
795
796         //load the platform from the file created in
     savePlatform
797
798         ObjectInputStream ois = new
```

```
798 ObjectInputStream(new FileInputStream(filename));
799
800         try {
801             Post.posts = (List<Post>) ois.readObject
    ();
802             List<Account> accounts = (List<Account
    >) ois.readObject();
803         } catch (IOException |
    ClassNotFoundException e) {
804             e.printStackTrace();
805             throw new IOException("Could not
    deserialize objects correctly");
806         }
807
808         //closes the object output stream
809         ois.close();
810     }
811 }
812
```

```java
package socialmedia;

import java.io.IOException;

/**
 * BadSocialMedia is a minimally compiling, but non-functioning implementor of
 * the SocialMediaPlatform interface.
 *
 * @author Diogo Pacheco
 * @version 1.0
 */
public class BadSocialMedia implements SocialMediaPlatform {

    @Override
    public int createAccount(String handle) throws IllegalHandleException, InvalidHandleException {
        // TODO Auto-generated method stub
        return 0;
    }

    @Override
    public int createAccount(String handle, String description) throws IllegalHandleException, InvalidHandleException {
        // TODO Auto-generated method stub
        return 0;
    }

    @Override
    public void removeAccount(int id) throws AccountIDNotRecognisedException {
        // TODO Auto-generated method stub

    }

    @Override
    public void removeAccount(String handle) throws HandleNotRecognisedException {
        // TODO Auto-generated method stub
```

```
35
36      }
37
38      @Override
39      public void changeAccountHandle(String oldHandle
    , String newHandle)
40              throws HandleNotRecognisedException,
    IllegalHandleException, InvalidHandleException {
41          // TODO Auto-generated method stub
42
43      }
44
45      @Override
46      public void updateAccountDescription(String
    handle, String description) throws
    HandleNotRecognisedException {
47          // TODO Auto-generated method stub
48
49      }
50
51      @Override
52      public String showAccount(String handle) throws
    HandleNotRecognisedException {
53          // TODO Auto-generated method stub
54          return null;
55      }
56
57      @Override
58      public int createPost(String handle, String
    message) throws HandleNotRecognisedException,
    InvalidPostException {
59          // TODO Auto-generated method stub
60          return 0;
61      }
62
63      @Override
64      public int endorsePost(String handle, int id)
65              throws HandleNotRecognisedException,
    PostIDNotRecognisedException,
    NotActionablePostException {
66          // TODO Auto-generated method stub
```

```
 67          return 0;
 68      }
 69
 70      @Override
 71      public int commentPost(String handle, int id,
    String message) throws HandleNotRecognisedException,
 72              PostIDNotRecognisedException,
    NotActionablePostException, InvalidPostException {
 73          // TODO Auto-generated method stub
 74          return 0;
 75      }
 76
 77      @Override
 78      public void deletePost(int id) throws
    PostIDNotRecognisedException {
 79          // TODO Auto-generated method stub
 80
 81      }
 82
 83      @Override
 84      public String showIndividualPost(int id) throws
    PostIDNotRecognisedException {
 85          // TODO Auto-generated method stub
 86          return null;
 87      }
 88
 89      @Override
 90      public StringBuilder showPostChildrenDetails(int
     id)
 91              throws PostIDNotRecognisedException,
    NotActionablePostException {
 92          // TODO Auto-generated method stub
 93          return null;
 94      }
 95
 96      @Override
 97      public int getNumberOfAccounts() {
 98          // TODO Auto-generated method stub
 99          return 0;
100      }
101
```

```
102     @Override
103     public int getTotalOriginalPosts() {
104         // TODO Auto-generated method stub
105         return 0;
106     }
107 
108     @Override
109     public int getTotalEndorsmentPosts() {
110         // TODO Auto-generated method stub
111         return 0;
112     }
113 
114     @Override
115     public int getTotalCommentPosts() {
116         // TODO Auto-generated method stub
117         return 0;
118     }
119 
120     @Override
121     public int getMostEndorsedPost() {
122         // TODO Auto-generated method stub
123         return 0;
124     }
125 
126     @Override
127     public int getMostEndorsedAccount() {
128         // TODO Auto-generated method stub
129         return 0;
130     }
131 
132     @Override
133     public void erasePlatform() {
134         // TODO Auto-generated method stub
135 
136     }
137 
138     @Override
139     public void savePlatform(String filename) throws
    IOException {
140         // TODO Auto-generated method stub
141 
```

```
142     }
143
144     @Override
145     public void loadPlatform(String filename) throws
    IOException, ClassNotFoundException {
146         // TODO Auto-generated method stub
147
148     }
149
150 }
151
```

```java
package socialmedia;

import java.io.IOException;

/**
 * BadMiniSocialMedia is a minimally compiling, but non-functioning implementor
 * of the MiniSocialMediaPlatform interface.
 *
 * @author Diogo Pacheco
 * @version 1.0
 */
public class BadMiniSocialMedia implements MiniSocialMediaPlatform {

    @Override
    public int createAccount(String handle) throws IllegalHandleException, InvalidHandleException {
        // TODO Auto-generated method stub
        return 0;
    }

    @Override
    public void removeAccount(int id) throws AccountIDNotRecognisedException {
        // TODO Auto-generated method stub

    }

    @Override
    public void changeAccountHandle(String oldHandle, String newHandle)
            throws HandleNotRecognisedException, IllegalHandleException, InvalidHandleException {
        // TODO Auto-generated method stub

    }

    @Override
    public String showAccount(String handle) throws HandleNotRecognisedException {
```

```
35          // TODO Auto-generated method stub
36          return null;
37      }
38
39      @Override
40      public int createPost(String handle, String
    message) throws HandleNotRecognisedException,
    InvalidPostException {
41          // TODO Auto-generated method stub
42          return 0;
43      }
44
45      @Override
46      public int endorsePost(String handle, int id)
47              throws HandleNotRecognisedException,
    PostIDNotRecognisedException,
    NotActionablePostException {
48          // TODO Auto-generated method stub
49          return 0;
50      }
51
52      @Override
53      public int commentPost(String handle, int id,
    String message) throws HandleNotRecognisedException,
54              PostIDNotRecognisedException,
    NotActionablePostException, InvalidPostException {
55          // TODO Auto-generated method stub
56          return 0;
57      }
58
59      @Override
60      public void deletePost(int id) throws
    PostIDNotRecognisedException {
61          // TODO Auto-generated method stub
62
63      }
64
65      @Override
66      public String showIndividualPost(int id) throws
    PostIDNotRecognisedException {
67          // TODO Auto-generated method stub
```

```
 68          return null;
 69      }
 70
 71      @Override
 72      public StringBuilder showPostChildrenDetails(int
     id)
 73              throws PostIDNotRecognisedException,
    NotActionablePostException {
 74          // TODO Auto-generated method stub
 75          return null;
 76      }
 77
 78      @Override
 79      public int getMostEndorsedPost() {
 80          // TODO Auto-generated method stub
 81          return 0;
 82      }
 83
 84      @Override
 85      public int getMostEndorsedAccount() {
 86          // TODO Auto-generated method stub
 87          return 0;
 88      }
 89
 90      @Override
 91      public void erasePlatform() {
 92          // TODO Auto-generated method stub
 93
 94      }
 95
 96      @Override
 97      public void savePlatform(String filename) throws
     IOException {
 98          // TODO Auto-generated method stub
 99
100      }
101
102      @Override
103      public void loadPlatform(String filename) throws
     IOException, ClassNotFoundException {
104          // TODO Auto-generated method stub
```

```
105
106     }
107
108 }
109
```

```java
package socialmedia;

/**
 * SocialMediaPlatform interface. This interface is a more elaborated version of
 * the MiniSocialMediaPlatform. The no-argument constructor of a class
 * implementing this interface should initialise the SocialMediaPlatform as an
 * empty platform with no initial accounts nor posts within it. For Pair
 * submissions.
 *
 * @author Diogo Pacheco
 * @version 1.0
 *
 */
public interface SocialMediaPlatform extends MiniSocialMediaPlatform {

    // Account-related methods **********************************

    /**
     * The method creates an account in the platform with the given handle and
     * description.
     * <p>
     * The state of this SocialMediaPlatform must be be unchanged if any exceptions
     * are thrown.
     *
     * @param handle      account's handle.
     * @param description account's description.
     * @throws IllegalHandleException if the handle already exists in the platform.
     * @throws InvalidHandleException if the new handle is empty, has more than 30
     *                                characters, or has white spaces.
     * @return the ID of the created account.
```

```
31      */
32     int createAccount(String handle, String
   description) throws IllegalHandleException,
   InvalidHandleException;
33
34     /**
35      * The method removes the account with the
   corresponding handle from the
36      * platform. When an account is removed, all of
   their posts and likes should
37      * also be removed.
38      * <p>
39      * The state of this SocialMediaPlatform must be
   be unchanged if any exceptions
40      * are thrown.
41      *
42      * @param handle account's handle.
43      * @throws HandleNotRecognisedException if the
   handle does not match to any
44      *                                      account
   in the system.
45      */
46     void removeAccount(String handle) throws
   HandleNotRecognisedException;
47
48     /**
49      * The method updates the description of the
   account with the respective handle.
50      * <p>
51      * The state of this SocialMediaPlatform must be
   be unchanged if any exceptions
52      * are thrown.
53      *
54      * @param handle      handle to identify the
   account.
55      * @param description new text for description.
56      * @throws HandleNotRecognisedException if the
   handle does not match to any
57      *                                      account
   in the system.
58      */
```

```
59      void updateAccountDescription(String handle,
    String description) throws
    HandleNotRecognisedException;
60
61      // End Post-related methods
     ****************************************
62
63      // Analytics-related methods
     ****************************************
64
65      /**
66       * This method returns the current total number
    of accounts present in the
67       * platform. Note, this is NOT the total number
    of accounts ever created since
68       * the current total should discount deletions.
69       *
70       * @return the total number of accounts in the
    platform.
71       */
72      int getNumberOfAccounts();
73
74      /**
75       * This method returns the current total number
    of original posts (i.e.,
76       * disregarding endorsements and comments)
    present in the platform. Note, this
77       * is NOT the total number of posts ever created
    since the current total should
78       * discount deletions.
79       *
80       * @return the total number of original posts in
    the platform.
81       */
82      int getTotalOriginalPosts();
83
84      /**
85       * This method returns the current total number
    of endorsement posts present in
86       * the platform. Note, this is NOT the total
    number of endorsements ever created
```

```
 87      * since the current total should discount
    deletions.
 88      *
 89      * @return the total number of endorsement posts
     in the platform.
 90      */
 91     int getTotalEndorsmentPosts();
 92
 93     /**
 94      * This method returns the current total number
    of comments posts present in the
 95      * platform. Note, this is NOT the total number
    of comments ever created since
 96      * the current total should discount deletions.
 97      *
 98      * @return the total number of comments posts in
     the platform.
 99      */
100     int getTotalCommentPosts();
101
102     // End Management-related methods
     ****************************************
103
104 }
```

```java
package socialmedia;

/**
 * Thrown when attempting to create a post which the message is empty or has
 * more characters than the system's limit.
 *
 * @author Diogo Pacheco
 * @version 1.0
 *
 */
public class InvalidPostException extends Exception {

    /**
     * Constructs an instance of the exception with no message
     */
    public InvalidPostException() {
        // do nothing
    }

    /**
     * Constructs an instance of the exception containing the message argument
     *
     * @param message message containing details regarding the exception cause
     */
    public InvalidPostException(String message) {
        super(message);
    }

}

```

```java
package socialmedia;

/**
 * Thrown when attempting to assign an account handle already in use in the
 * system.
 *
 * @author Diogo Pacheco
 * @version 1.0
 *
 */
public class IllegalHandleException extends Exception {

    /**
     * Constructs an instance of the exception with no message
     */
    public IllegalHandleException() {
        // do nothing
    }

    /**
     * Constructs an instance of the exception containing the message argument
     *
     * @param message message containing details regarding the exception cause
     */
    public IllegalHandleException(String message) {
        super(message);
    }

}

```

```java
package socialmedia;

/**
 * Thrown when attempting to assign an account handle empty or having more than
 * the system limit of characters. A handle must be a single word, i.e., no
 * white spaces allowed.
 *
 * @author Diogo Pacheco
 * @version 1.0
 *
 */
public class InvalidHandleException extends Exception {

    /**
     * Constructs an instance of the exception with no message
     */
    public InvalidHandleException() {
        // do nothing
    }

    /**
     * Constructs an instance of the exception containing the message argument
     *
     * @param message message containing details regarding the exception cause
     */
    public InvalidHandleException(String message) {
        super(message);
    }

}

```

```java
package socialmedia;

import java.io.IOException;
import java.io.Serializable;

/**
 * MiniSocialMediaPlatform interface. The no-argument constructor of a class
 * implementing this interface should initialise the MiniSocialMediaPlatform as
 * an empty platform with no initial accounts nor posts within it. For Solo
 * submissions ONLY.
 *
 * @author Diogo Pacheco
 * @version 1.0
 *
 */
public interface MiniSocialMediaPlatform extends Serializable {

    // Account-related methods
    ****************************************

    /**
     * The method creates an account in the platform with the given handle.
     * <p>
     * The state of this SocialMediaPlatform must be be unchanged if any exceptions
     * are thrown.
     *
     * @param handle account's handle.
     * @throws IllegalHandleException if the handle already exists in the platform.
     * @throws InvalidHandleException if the new handle is empty, has more than 30
     *                                characters, or has white spaces.
     * @return the ID of the created account.
     *
```

```
32      */
33     int createAccount(String handle) throws
   IllegalHandleException, InvalidHandleException;
34
35     /**
36      * The method removes the account with the
   corresponding ID from the platform.
37      * When an account is removed, all of their posts
    and likes should also be
38      * removed.
39      * <p>
40      * The state of this SocialMediaPlatform must be
   be unchanged if any exceptions
41      * are thrown.
42      *
43      * @param id ID of the account.
44      * @throws AccountIDNotRecognisedException if the
    ID does not match to any
45      *
   account in the system.
46      */
47     void removeAccount(int id) throws
   AccountIDNotRecognisedException;
48
49     /**
50      * The method replaces the oldHandle of an
   account by the newHandle.
51      * <p>
52      * The state of this SocialMediaPlatform must be
   be unchanged if any exceptions
53      * are thrown.
54      *
55      * @param oldHandle account's old handle.
56      * @param newHandle account's new handle.
57      * @throws HandleNotRecognisedException if the
   old handle does not match to any
58      *                                      account
   in the system.
59      * @throws IllegalHandleException       if the
   new handle already exists in the
60      *                                      platform.
```

```
61      * @throws InvalidHandleException       if the
    new handle is empty, has more
62      *                                      than 30
    characters, or has white spaces.
63      */
64     void changeAccountHandle(String oldHandle,
    String newHandle)
65             throws HandleNotRecognisedException,
    IllegalHandleException, InvalidHandleException;
66
67     /**
68      * The method creates a formatted string
    summarising the stats of the account
69      * identified by the given handle. The template
    should be:
70      *
71      * <pre>
72      * ID: [account ID]
73      * Handle: [account handle]
74      * Description: [account description]
75      * Post count: [total number of posts, including
     endorsements and replies]
76      * Endorse count: [sum of endorsements received
    by each post of this account]
77      * </pre>
78      *
79      * @param handle handle to identify the account.
80      * @return the account formatted summary.
81      * @throws HandleNotRecognisedException if the
    handle does not match to any
82      *                                      account
    in the system.
83      */
84     String showAccount(String handle) throws
    HandleNotRecognisedException;
85
86     // End Account-related methods
     ****************************************
87
88     // Post-related methods
     ****************************************
```

```
 89
 90      /**
 91       * The method creates a post for the account identified by the given handle with
 92       * the following message.
 93       * <p>
 94       * The state of this SocialMediaPlatform must be be unchanged if any exceptions
 95       * are thrown.
 96       *
 97       * @param handle  handle to identify the account.
 98       * @param message post message.
 99       * @throws HandleNotRecognisedException if the handle does not match to any
100       *                                      account in the system.
101       * @throws InvalidPostException         if the message is empty or has more than
102       *                                      100 characters.
103       * @return the sequential ID of the created post.
104       */
105      int createPost(String handle, String message) throws HandleNotRecognisedException,
         InvalidPostException;
106
107      /**
108       * The method creates an endorsement post of an existing post, similar to a
109       * retweet on Twitter. An endorsement post is a special post. It contains a
110       * reference to the endorsed post and its message is formatted as:
111       * <p>
112       * <code>"EP@" + [endorsed account handle] + ": " + [endorsed message]</code>
113       * <p>
114       * The state of this SocialMediaPlatform must be be unchanged if any exceptions
```

```
115      * are thrown.
116      *
117      * @param handle of the account endorsing a post
    .
118      * @param id     of the post being endorsed.
119      * @return the sequential ID of the created post
    .
120      * @throws HandleNotRecognisedException if the
    handle does not match to any
121      *                                      account
    in the system.
122      * @throws PostIDNotRecognisedException if the
    ID does not match to any post in
123      *                                      the
    system.
124      * @throws NotActionablePostException   if the
    ID refers to a endorsement post.
125      *
    Endorsement posts are not endorsable.
126      *
    Endorsements are not transitive. For
127      *                                      instance
    , if post A is endorsed by post
128      *                                      B, and
    an account wants to endorse B, in
129      *                                      fact,
    the endorsement must refers to A.
130      */
131     int endorsePost(String handle, int id)
132             throws HandleNotRecognisedException,
    PostIDNotRecognisedException,
    NotActionablePostException, InvalidPostException;
133
134     /**
135      * The method creates a comment post referring
    to an existing post, similarly to
136      * a reply on Twitter. A comment post is a
    special post. It contains a reference
137      * to the post being commented upon.
138      * <p>
139      * The state of this SocialMediaPlatform must be
```

```
139  be unchanged if any exceptions
140      * are thrown.
141      *
142      * @param handle  of the account commenting a
    post.
143      * @param id      of the post being commented.
144      * @param message the comment post message.
145      * @return the sequential ID of the created post
    .
146      * @throws HandleNotRecognisedException if the
    handle does not match to any
147      *                                      account
    in the system.
148      * @throws PostIDNotRecognisedException if the
    ID does not match to any post in
149      *                                      the
    system.
150      * @throws NotActionablePostException   if the
    ID refers to a endorsement post.
151      *
    Endorsement posts are not endorsable.
152      *
    Endorsements cannot be commented. For
153      *                                      instance
    , if post A is endorsed by post
154      *                                      B, and
    an account wants to comment B, in
155      *                                      fact,
    the comment must refers to A.
156      * @throws InvalidPostException         if the
    comment message is empty or has
157      *                                      more
    than 100 characters.
158      */
159     int commentPost(String handle, int id, String
    message) throws HandleNotRecognisedException,
160             PostIDNotRecognisedException,
    NotActionablePostException, InvalidPostException;
161
162     /**
163      * The method removes the post from the platform
```

```
163 . When a post is removed, all
164      * its endorsements should be removed as well.
    All replies to this post should
165      * be updated by replacing the reference to this
     post by a generic empty post.
166      * <p>
167      * The generic empty post message should be "The
     original content was removed
168      * from the system and is no longer available
    .". This empty post is just a
169      * replacement placeholder for the post which a
    reply refers to. Empty posts
170      * should not be linked to any account and
    cannot be acted upon, i.e., it cannot
171      * be available for endorsements or replies.
172      * <p>
173      * The state of this SocialMediaPlatform must be
     be unchanged if any exceptions
174      * are thrown.
175      *
176      * @param id ID of post to be removed.
177      * @throws PostIDNotRecognisedException if the
    ID does not match to any post in
178      *                                        the
    system.
179      */
180     void deletePost(int id) throws
    PostIDNotRecognisedException;
181
182     /**
183      * The method generates a formated string
    containing the details of a single
184      * post. The format is as follows:
185      *
186      * <pre>
187      * ID: [post ID]
188      * Account: [account handle]
189      * No. endorsements: [number of endorsements
    received by the post] | No. comments: [number of
    comments received by the post]
190      * [post message]
```

```
191      * </pre>
192      *
193      * @param id of the post to be shown.
194      * @return a formatted string containing post's
    details.
195      * @throws PostIDNotRecognisedException if the
    ID does not match to any post in
196      *                                        the
    system.
197      */
198     String showIndividualPost(int id) throws
    PostIDNotRecognisedException;
199
200     /**
201      * The method builds a StringBuilder showing the
     details of the current post and
202      * all its children posts. The format is as
    follows (you can use tabs or spaces to represent
    indentation):
203      *
204      * <pre>
205      * {@link #showIndividualPost(int)
    showIndividualPost(id)}
206      * |
207      * [for reply: replies to the post sorted by ID]
208      * |  > {@link #showIndividualPost(int)
    showIndividualPost(reply)}
209      * </pre>
210      *
211      * See an example:
212      *
213      * <pre>
214      * ID: 1
215      * Account: user1
216      * No. endorsements: 2 | No. comments: 3
217      * I like examples.
218      * |
219      * | > ID: 3
220      *     Account: user2
221      *     No. endorsements: 0 | No. comments: 1
222      *     No more than me...
```

```
223      *     |
224      *     | > ID: 5
225      *         Account: user1
226      *         No. endorsements: 0 | No. comments: 1
227      *         I can prove!
228      *         |
229      *         | > ID: 6
230      *             Account: user2
231      *             No. endorsements: 0 | No. comments: 0
232      *             prove it
233      * | > ID: 4
234      *     Account: user3
235      *     No. endorsements: 4 | No. comments: 0
236      *     Can't you do better than this?
237      *
238      * | > ID: 7
239      *     Account: user5
240      *     No. endorsements: 0 | No. comments: 1
241      *     where is the example?
242      *     |
243      *     | > ID: 10
244      *         Account: user1
245      *         No. endorsements: 0 | No. comments: 0
246      *         This is the example!
247      * </pre>
248      *
249      * Continuing with the example, if the method is called for post ID=5
250      * ({@code showIndividualPost(5)}), the return would be:
251      *
252      * <pre>
253      * ID: 5
254      * Account: user1
255      * No. endorsements: 0 | No. comments: 1
256      * I can prove!
257      * |
```

```
258      * | > ID: 6
259      *     Account: user2
260      *     No. endorsements: 0 | No. comments: 0
261      *     prove it
262      * </pre>
263      *
264      * @param id of the post to be shown.
265      * @return a formatted StringBuilder containing
    the details of the post and its
266      *         children.
267      * @throws PostIDNotRecognisedException if the
    ID does not match to any post in
268      *                                        the
    system.
269      * @throws NotActionablePostException   if the
    ID refers to an endorsement post.
270      *
    Endorsement posts do not have children
271      *                                        since
    they are not endorsable nor
272      *
    commented.
273      */
274     StringBuilder showPostChildrenDetails(int id)
    throws PostIDNotRecognisedException,
    NotActionablePostException;
275
276     // End Post-related methods
     ************************************
277
278     // Analytics-related methods
     ************************************
279
280     /**
281      * This method identifies and returns the post
    with the most number of
282      * endorsements, a.k.a. the most popular post.
283      *
284      * @return the ID of the most popular post.
285      */
286     int getMostEndorsedPost();
```

```
287
288     /**
289      * This method identifies and returns the
    account with the most number of
290      * endorsements, a.k.a. the most popular account
    .
291      *
292      * @return the ID of the most popular account.
293      */
294     int getMostEndorsedAccount();
295
296     // End Analytics-related methods
     ****************************************
297
298     // Management-related methods
     ****************************************
299
300     /**
301      * Method empties this SocialMediaPlatform of
    its contents and resets all
302      * internal counters.
303      */
304     void erasePlatform();
305
306     /**
307      * Method saves this SocialMediaPlatform's
    contents into a serialised file, with
308      * the filename given in the argument.
309      *
310      * @param filename location of the file to be
    saved
311      * @throws IOException if there is a problem
    experienced when trying to save the
312      *                     store contents to the
    file
313      */
314     void savePlatform(String filename) throws
    IOException;
315
316     /**
317      * Method should load and replace this
```

```
317 SocialMediaPlatform's contents with the
318      * serialised contents stored in the file given
    in the argument.
319      * <p>
320      * The state of this SocialMediaPlatform's must
    be be unchanged if any
321      * exceptions are thrown.
322      *
323      * @param filename location of the file to be
    loaded
324      * @throws IOException            if there is a
    problem experienced when trying
325      *                                to load the
    store contents from the file
326      * @throws ClassNotFoundException if required
    class files cannot be found when
327      *                                loading
328      */
329     void loadPlatform(String filename) throws
    IOException, ClassNotFoundException;
330
331     // End Management-related methods
     ****************************************
332
333 }
334
```

```java
package socialmedia;

/**
 * Thrown when attempting to act upon an not-actionable post.
 *
 * @author Diogo Pacheco
 * @version 1.0
 *
 */
public class NotActionablePostException extends Exception {

    /**
     * Constructs an instance of the exception with no message
     */
    public NotActionablePostException() {
        // do nothing
    }

    /**
     * Constructs an instance of the exception containing the message argument
     *
     * @param message message containing details regarding the exception cause
     */
    public NotActionablePostException(String message) {
        super(message);
    }

}

```

```java
package socialmedia;

/**
 * Thrown when attempting to use an account handle that does not exit in the
 * system.
 *
 * @author Diogo Pacheco
 * @version 1.0
 *
 */
public class HandleNotRecognisedException extends Exception {

    /**
     * Constructs an instance of the exception with no message
     */
    public HandleNotRecognisedException() {
        // do nothing
    }

    /**
     * Constructs an instance of the exception containing the message argument
     *
     * @param message message containing details regarding the exception cause
     */
    public HandleNotRecognisedException(String message) {
        super(message);
    }

}

```

```java
package socialmedia;

/**
 * Thrown when attempting to use a post ID that does not exit in the system.
 *
 * @author Diogo Pacheco
 * @version 1.0
 *
 */
public class PostIDNotRecognisedException extends Exception {

    /**
     * Constructs an instance of the exception with no message
     */
    public PostIDNotRecognisedException() {
        // do nothing
    }

    /**
     * Constructs an instance of the exception containing the message argument
     *
     * @param message message containing details regarding the exception cause
     */
    public PostIDNotRecognisedException(String message) {
        super(message);
    }

}

```

```java
package socialmedia;

/**
 * Thrown when attempting to use an account ID that does not exit in the system.
 *
 * @author Diogo Pacheco
 * @version 1.0
 *
 */
public class AccountIDNotRecognisedException extends Exception {

    /**
     * Constructs an instance of the exception with no message
     */
    public AccountIDNotRecognisedException() {
        // do nothing
    }

    /**
     * Constructs an instance of the exception containing the message argument
     *
     * @param message message containing details regarding the exception cause
     */
    public AccountIDNotRecognisedException(String message) {
        super(message);
    }

}

```