1 Account.java

```java
package socialmedia;


import java.io.Serializable;

/**
 * @author at753,ag811
 * This is a class of accounts, it keeps all the needed attributes of an account.
 */
public class Account implements Serializable {

    int id;
    String Handle;
    String Description;
    boolean Exists;
    int endorsementCount;

    /**
     * @return returns a boolean to show if an account is deleted or not.
     */
    public boolean isExists() {
        return Exists;
    }

    /**
     * @param exists is a boolean to show if an account is deleted or not.
     */
    public void setExists(boolean exists) {
        Exists = exists;
    }

    /**
     * @return returns the id of an account.
     */
    public int getId() {
        return id;
    }

    /**
     * @param id the id of an account.
     */
    public void setId(int id) {
        this.id = id;

    }

    /**
     * @return returns the handle of an account.
     */
    public String getHandle() {
        return Handle;
    }
```

```
53
54      /**
55       * @param handle the handle of an account.
56       */
57      public void setHandle(String handle) {
58          Handle = handle;
59      }
60
61      /**
62       * @return returns the description of an account.
63       */
64      public String getDescription() {
65          return Description;
66      }
67
68      /**
69       * @param description the description of an account
70       */
71      public void setDescription(String description) {
72          Description = description;
73      }
74
75      /**
76       * Constructor method for Account class.
77       */
78      public Account() {
79          super();
80          // TODO Auto-generated constructor stub
81      }
82
83  }
```

2 Post.java

```
1   package socialmedia;
2
3   import java.io.Serializable;
4
5   /**
6    * @author at753,ag811 This is a class of posts, it keeps all the needed attributes of
7    *         a class.
8    */
9   public class Post implements Serializable {
10     int id;
11     int parentId;
12     int endorsements;
13     int comments;
14     Account account;
15     String message;
16     boolean exists;
17     boolean endorsedPost;
18
19     /**
20      * @return returns a true if that post exists, false if it's deleted.
```

```
21      */
22     public boolean isExists() {
23        return exists;
24     }
25
26     /**
27      * @return returns a true if that post is endorsed, false if it's not.
28      */
29
30     public boolean isEndorsedPost() {
31        return endorsedPost;
32     }
33
34     /**
35      * @param endorsedPost is a boolean to show if a post is an endorsement or not
36      */
37     public void setEndorsedPost(boolean endorsedPost) {
38        this.endorsedPost = endorsedPost;
39     }
40
41     /**
42      * @param exists will be true if it's a deleting procedure, false if it's a
43      *               creating procedure.
44      */
45     public void setExists(boolean exists) {
46        this.exists = exists;
47     }
48
49     /**
50      * @return returns the id of a post.
51      */
52     public int getId() {
53        return id;
54     }
55
56     /**
57      * @param id is the id of a post.
58      */
59     public void setId(int id) {
60        this.id = id;
61     }
62
63     /**
64      * @return returns the parent posts id of a post.
65      */
66     public int getParentId() {
67        return parentId;
68     }
69
70     /**
71      * @param parentId is to set when assigning a post as a child.
72      */
73     public void setParentId(int parentId) {
74        this.parentId = parentId;
75     }
```

```
76
77      /**
78       * @return returns the count of endorsements this account got.
79       */
80      public int getEndorsements() {
81         return endorsements;
82      }
83
84      /**
85       * @param endorsements is the new endorsement count.
86       */
87      public void setEndorsements(int endorsements) {
88         this.endorsements = endorsements;
89      }
90
91      /**
92       * @return returns the comment count of a post.
93       */
94      public int getComments() {
95         return comments;
96      }
97
98      /**
99       * @param comments is the new comment count
100      */
101     public void setComments(int comments) {
102        this.comments = comments;
103     }
104
105     /**
106      * @return returns the author account of this post.
107      */
108     public Account getAccount() {
109        return account;
110     }
111
112     /**
113      * @param account is the author of a post.
114      */
115     public void setAccount(Account account) {
116        this.account = account;
117     }
118
119     /**
120      * @return returns the message of the post.
121      */
122     public String getMessage() {
123        return message;
124     }
125
126     /**
127      * @param message is the message of a post.
128      */
129     public void setMessage(String message) {
130        this.message = message;
```

```
    }

    /**
     * This is a constructor method for post class.
     */
    public Post() {
        super();
        // TODO Auto-generated constructor stub
    }
}
```

3 SocialMedia.java

```
package socialmedia;

import java.io.*;
import java.util.ArrayList;
import java.util.Iterator;

/**
 * SocialMedia is a minimally compiling, but non-functioning implementor of the
 * SocialMediaPlatform interface.
 *
 * @author Diogo Pacheco
 * @version 1.0
 */
public class SocialMedia implements SocialMediaPlatform {
    int accountId = 1;
    int postId = 1;

    ArrayList<Account> accounts = new ArrayList<Account>();
    ArrayList<Post> posts = new ArrayList<Post>();

    /**
     * The method finds an account from the arraylist of accounts. It takes in a
     * string called handle. The method iterates through all the objects in the
     * arraylist, comparing the parameter that was inputted with the value of the
     * attribute Handle that has been stored for each object in the arraylist. When
     * a match is found, the Account object with the match is returned.
     *
     * @param handle The Handle of the account, the user is searching for
     *
     */

    private Account findAccountByHandle(String handle) {
        for (Account account : accounts) {
            if (account.Handle.equals(handle)) {
                return account;
            }
        }

        return null;
    }

```

```
42      /**
43       * The method finds an account from the arraylist of accounts. It takes in an ID
44       * as a parameter and compares it with every ID attribute in the arraylist
45       * accounts to see if there is a match. If there is, then the matching Account
46       * object is returned.
47       *
48       * @param id The ID of the account, the user is searching for
49       * @throws AccountIDNotRecognisedException if the ID does not match any IDs in
50       *                                         the system.
51       */
52
53      private Account findAccountById(int id) throws AccountIDNotRecognisedException {
54         for (Account account : accounts) {
55            if (account.getId() == id) {
56               return account;
57            }
58         }
59         throw new AccountIDNotRecognisedException("ID not recognised.");
60      }
61
62      private Post findPostById(int id) throws PostIDNotRecognisedException {
63         for (Post post : posts) {
64            if (post.getId() == id) {
65               return post;
66            }
67         }
68         throw new PostIDNotRecognisedException("ID not recognised");
69      }
70
71      @Override
72      public int createAccount(String handle) throws IllegalHandleException, InvalidHandleException {
73
74         // Check if handle meets specification requirements i.e. under 30 chars, can't
75         // be null
76         // or have white spaces
77         if (handle.length() > 30 || handle.contains(" ") || handle == "") {
78            throw new InvalidHandleException(
79                  "Handle must be under 30 characters, cannot contain whitespace, cannot be null.");
80         }
81         if (findAccountByHandle(handle) != null) {
82            throw new IllegalHandleException("Handle already exists on the platform.");
83         } else {
84
85            // Creates Account object and stores in arraylist accounts if all validation is
86            // successful
87            Account account = new Account();
88            account.id = accountId;
89            account.Handle = handle;
90            accounts.add(account);
91            accountId++;
92            account.setExists(true);
93
94            return account.id;
95         }
96      }
```

```
97
98      @Override
99      public int createAccount(String handle, String description) throws IllegalHandleException,
            InvalidHandleException {
100
101        // Validation
102
103        if (handle.length() > 30 || handle.contains(" ") || handle == "") {
104           throw new InvalidHandleException(
105                 "Handle must be under 30 characters, cannot contain whitespace, cannot be null.");
106        }
107        if (findAccountByHandle(handle) != null) {
108           throw new IllegalHandleException("Handle already exists on the platform.");
109        } else {
110
111           // Creates Account object and stores in arraylist accounts if all validation is
112           // successful
113           Account account = new Account();
114           account.id = accountId;
115           account.Handle = handle;
116           account.setExists(true);
117           account.setDescription(description);
118           accounts.add(account);
119           accountId++;
120           account.setExists(true);
121
122           return account.id;
123        }
124     }
125
126     @Override
127     public void removeAccount(int id) throws AccountIDNotRecognisedException {
128
129        // Tries to remove account with given account ID if unsuccessful
130        // an exception is thrown
131        try {
132           for (Post p : posts) {
133              if (p.getAccount().id == id) {
134                 deletePost(id);
135              }
136           }
137           accounts.remove(findAccountById(id));
138        } catch (Exception AccountIDNotRecognisedException) {
139           throw new AccountIDNotRecognisedException("Account ID not recognised.");
140        }
141     }
142
143     @Override
144     public void removeAccount(String handle) throws HandleNotRecognisedException {
145        // Tries to remove account with given account handle if unsuccessful
146        // an exception is thrown
147        Iterator<Post> iter = posts.iterator();
148        if (findAccountByHandle(handle) == null) {
149           throw new HandleNotRecognisedException("Account handle not recognised.");
150        }
```

```
151         while (iter.hasNext()) {
152            Post p = iter.next();
153            if (p.getAccount().getHandle().equals(handle)) {
154               try {
155
156                  deletePost(p.getId());
157               } catch (PostIDNotRecognisedException e) {
158                  e.printStackTrace();
159
160               }
161            }
162         }
163         accounts.remove(findAccountByHandle(handle));
164
165      }
166
167      @Override
168      public void changeAccountHandle(String oldHandle, String newHandle)
169            throws HandleNotRecognisedException, IllegalHandleException, InvalidHandleException {
170
171         // Validation
172         if (newHandle.length() > 30 || newHandle.contains(" ") || newHandle == "") {
173            throw new InvalidHandleException(
174                  "Handle must be under 30 characters, cannot contain whitespace, cannot be null.");
175         }
176         if (findAccountByHandle(newHandle) != null) {
177            throw new IllegalHandleException("Handle already exists on the platform.");
178         } else {
179            try {
180
181               // find the Account object with oldHandle as the matching Handle attribute value
182               // and set the Handle attribute with the value of newHandle
183               // throw an exception if not found
184
185               findAccountByHandle(oldHandle).Handle = newHandle;
186            } catch (Exception HandleNotRecognisedException) {
187               throw new HandleNotRecognisedException("Handle not recognised.");
188            }
189         }
190      }
191
192      @Override
193      public void updateAccountDescription(String handle, String description) throws
           HandleNotRecognisedException {
194         if (findAccountByHandle(handle) == null) {
195            throw new HandleNotRecognisedException("Handle not found in the platform.");
196         }
197
198         // Find account object with matching handle and change the value of its
199         // attribute description
200         // throw an exception if unsuccessful
201         findAccountByHandle(handle).setDescription(description);
202      }
203
204      @Override
```

```
205     public String showAccount(String handle) throws HandleNotRecognisedException {
206
207        // Initialisation of variables
208        String showAccount = "";
209        Account accountToShow = findAccountByHandle(handle);
210        if (accountToShow == null) {
211           throw new HandleNotRecognisedException("Handle not found in platform."); // Thrown when an unknown
                handle
212           // is inputted
213        }
214        int postCount = 0;
215        int endorsementCount = 0;
216        for (Post p : posts) { // Iterate through the arraylist posts
217
218           // Comparing currently iterated object's Account attribute with searched Account
219           // If true then the endorsement and post counts are updated
220           if (p.getAccount().equals(accountToShow)) {
221              postCount++;
222              endorsementCount = p.getEndorsements() + endorsementCount;
223           }
224
225        }
226        // Creating and storing a formatted string
227        showAccount = "ID:" + accountToShow.getId() + "\n" + "Handle:" + accountToShow.getHandle() + "\n"
228              + "Description:" + accountToShow.getDescription() + "\n" + "Post Count:" + postCount + "\n"
229              + "Endorse Count:" + endorsementCount;
230
231        return showAccount;
232     }
233
234     @Override
235     public int createPost(String handle, String message) throws HandleNotRecognisedException,
           InvalidPostException {
236
237        // Exception thrown when handle is not found in system
238        if (findAccountByHandle(handle) == null) {
239           throw new HandleNotRecognisedException("Handle not found in platform, Please try again");
240        }
241
242        // Validation
243        if (message.isEmpty() || message.length() > 100) {
244           throw new InvalidPostException("Message was greater than 100 characters or empty");
245        } else {
246
247           // New Post Object created and stored in arraylist posts if validation
248           // successful
249           Post post = new Post();
250           post.id = postId;
251           post.account = findAccountByHandle(handle);
252           post.message = message;
253           post.endorsedPost = false;
254           post.exists = true;
255           posts.add(post);
256           postId++;
257           return post.getId();
```

```
258         }
259
260     }
261
262     @Override
263     public int endorsePost(String handle, int id)
264           throws HandleNotRecognisedException, PostIDNotRecognisedException, NotActionablePostException {
265
266        // Exception thrown when handle is not found in system
267        if (findAccountByHandle(handle) == null) {
268           throw new HandleNotRecognisedException("Handle not found in platform, Please try again");
269        } else {
270           for (Post p : posts) {
271              // Exception thrown when the same user tries to endorse the same post
272              if (p.account.getHandle().equals(handle) && p.parentId == id && p.endorsedPost) {
273                 throw new NotActionablePostException("Can't endorse same post twice");
274              }
275           }
276           for (Post p : posts) { // Iterating through posts
277              if (p.getId() == id) {
278                 if (p.getParentId() != 0) {
279
280                    // Exceptions thrown when user tries to endorse an endorsed post
281                    // or a comment
282                    if (p.isEndorsedPost()) {
283                       throw new NotActionablePostException("Can't endorse another endorsed post");
284                    }
285                    throw new NotActionablePostException("Cannot endorse a comment.");
286                 }
287
288                 // Exception thrown when user tries to endorse a deleted post
289                 if (!p.isExists()) {
290                    throw new NotActionablePostException("Post has been deleted, cannot comment");
291                 }
292
293                 // Creates a new Post object and stores it in arraylist posts to store the
294                 // endorsement when no exceptions are thrown
295                 // Unlike normal Post objects, endorsed posts have the attribute endorsedPost
296                 // set to true to differentiate them from original posts
297                 String endorsedPost = "EP@" + p.account.getHandle() + ": " + p.getMessage();
298                 Post post = new Post();
299                 post.id = postId;
300                 post.parentId = p.getId();
301                 post.account = findAccountByHandle(handle);
302                 post.message = endorsedPost;
303                 post.endorsedPost = true;
304                 post.exists = true;
305                 posts.add(post);
306                 p.endorsements++;
307                 p.getAccount().endorsementCount++;
308                 postId++;
309                 return post.getId();
310
311              }
312
```

```
313         }
314
315         // Exception thrown if Post ID is not found in the system
316         throw new PostIDNotRecognisedException("Post ID not found in the platform");
317      }
318   }
319
320   @Override
321   public int commentPost(String handle, int id, String message) throws HandleNotRecognisedException,
322         PostIDNotRecognisedException, NotActionablePostException, InvalidPostException {
323
324      // Validation
325      if (message.length() > 100 || message.isEmpty()) {
326         throw new InvalidPostException("Message cannot be empty or greater than 100 characters");
327      }
328      if (findAccountByHandle(handle) == null) {
329         throw new HandleNotRecognisedException("Handle not found in the platform");
330      } else {
331         for (Post p : posts) { // Iterating through posts
332            if (p.getId() == id) {
333
334               // Exceptions thrown if user tries to comment on an endorsed post
335               // or a deleted post
336               if (p.isEndorsedPost()) {
337                  throw new NotActionablePostException("Can't comment on an endorsed post");
338               }
339               if (!p.isExists()) {
340                  throw new NotActionablePostException("Post has been deleted, cannot comment");
341               }
342
343               // New Post Object created and stored in arraylist posts if no exceptions thrown
344               // with the parentId attribute set to id as both endorsed posts and original
345               // posts
346               // have null parentId attributes, so that it is possible to differentiate
347               // between the
348               // different type of posts.
349               p.comments++;
350               Post post = new Post();
351               post.id = postId;
352               post.parentId = id;
353               post.account = findAccountByHandle(handle);
354               post.message = message;
355               post.endorsedPost = false;
356               post.exists = true;
357               posts.add(post);
358               postId++;
359               return post.getId();
360            }
361         }
362
363         // Exception thrown when Post ID is not found in the system
364         throw new PostIDNotRecognisedException("Post ID not found in the platform");
365      }
366
367   }
```

```
368
369     @Override
370     public void deletePost(int id) throws PostIDNotRecognisedException {
371        Iterator<Post> iter2 = posts.iterator();
372
373        // Initialisation
374        int commentCounter = 0;
375
376        for (Post i : posts) {
377
378           // Iterating through posts
379           if (i.getParentId() == id && !i.isEndorsedPost()) {
380              commentCounter++;
381           }
382           if (i.getParentId() == id && i.isEndorsedPost()) {
383              // TODO ERROR CAUSE
384              // posts.remove(i);
385              i.setMessage("-This was an endorsement but the original post has been deleted.-");
386           }
387
388        }
389        while (iter2.hasNext()) { // Iterating through posts
390           Post p = iter2.next();
391           if (p.getId() == id) {
392
393              // If the counter value has not been updated, it means that the post has no
394              // comments
395              if (p.isEndorsedPost()) {
396                 for (Post post : posts) {
397                    if ((post.id == p.parentId && p.endorsedPost)) {
398                       post.endorsements--;
399                       post.account.endorsementCount--;
400                    }
401                 }
402              } // so the post can be removed entirely from the system
403              if (commentCounter == 0) {
404                 // We were supposed to remove the post completely but after struggling with
405                 // ConcurrentModificationException for days, we decided to only change the
406                 // message.
407                 //posts.remove(p); // Remove post completely.
408                 p.setExists(false);
409              }
410
411              // Placeholder message set instead if post has comment posts
412              p.setMessage("The original content was removed from the system and is no longer available.");
413              p.setExists(false);
414
415              // Update user total endorsements as this post doesn't exist anymore
416              // So it should not contribute to user's total endorsements
417              p.getAccount().endorsementCount = p.getAccount().endorsementCount - p.endorsements;
418              p.setEndorsements(0); // Update value of post endorsements to 0 as it doesn't exist now
419              return;
420           }
421
422        }
```

```
423         // Exception thrown when Post ID is not found in the system
424         throw new PostIDNotRecognisedException("Post ID not recognised.");
425 
426     }
427 
428     @Override
429     public String showIndividualPost(int id) throws PostIDNotRecognisedException {
430         for (Post p : posts) { // Iterates through posts
431             // returns a formatted string if currently iterated object has the same id value
432             // as the parameter
433             if (p.getId() == id) {
434                 if (!p.isExists()) {
435                     String formattedString = "ID: " + " " + "\n" + "Account: " + " " + "\n"
436                             + "No. endorsements: " + p.getEndorsements() + " | " + "No. comments: " +
                                p.getComments() + "\n"
437                             + "Content deleted.";
438                     return formattedString;
439 
440                 }
441                 else {
442                     String formattedString = "ID: " + p.getId() + "\n" + "Account: " + p.getAccount().getHandle()
                        + "\n"
443                             + "No. endorsements: " + p.getEndorsements() + " | " + "No. comments: " + p.getComments()
444                             + "\n" + p.getMessage();
445                     return formattedString;
446 
447                 }
448 
449             }
450         }
451 
452         // Exception thrown if Post ID not found in system
453         throw new PostIDNotRecognisedException("Post ID not found in the platform");
454     }
455 
456     @Override
457     public StringBuilder showPostChildrenDetails(int id)
458             throws PostIDNotRecognisedException, NotActionablePostException {
459 
460         // Initialising a StringBuilder object with the parent object as the value
461         StringBuilder sb = new StringBuilder(showIndividualPost(id));
462         if (sb.isEmpty()) {
463 
464             // Exception thrown when Post ID is not found in system
465             throw new PostIDNotRecognisedException("Post ID not found in platform");
466         }
467 
468         for (Post p : posts) {
469 
470             // Exception thrown when an endorsed post is selected as endorsed posts cannot
471             // have
472             // comments or endorsements
473             if (p.getId() == id && p.isEndorsedPost()) {
474                 throw new NotActionablePostException("Endorsed posts cannot have comments");
475             }
```

```
476
477         if (p.getParentId() == id && !p.isEndorsedPost()) { // Checks if parent object had any comments
478
479             // Converting object to string to allow for string manipulation for formatting
480             // purposes
481             String indent = showPostChildrenDetails(p.getId()).toString();
482             indent = indent.replaceAll("\n", "\n\t");
483             sb.append("\n|");
484             sb.append("\n| > ");
485             sb.append(indent); // Appending formatted string back to StringBuilder Object
486         }
487       }
488       return sb;
489     }
490
491     @Override
492     public int getNumberOfAccounts() {
493       int numberOfAccounts = 0;
494       for (Account account : accounts) { // Iterating through accounts
495         if (account.isExists()) {
496           numberOfAccounts++; // Increments counter if account exists
497         }
498       }
499       return numberOfAccounts;
500     }
501
502     @Override
503     public int getTotalOriginalPosts() {
504       int originalPosts = 0;
505       for (Post post : posts) { // Iterating through posts
506         if (post.isExists()) {
507           if (post.endorsedPost || post.parentId > 0) {
508             continue;
509             // Validating it is not an endorsed post or a comment post
510           } else {
511             originalPosts++; // Increments counter if conditions met
512           }
513         }
514       }
515       return originalPosts;
516
517     }
518
519     @Override
520     public int getTotalEndorsmentPosts() {
521       int endorsementPosts = 0;
522       for (Post post : posts) { // Iterating through posts
523         if (post.isExists()) { // Check if post still exists
524           if (post.endorsedPost) { // Check if post is an endorsed post
525             endorsementPosts++; // Increment when conditions met
526           }
527         }
528       }
529       return endorsementPosts;
530     }
```

```
531
532     @Override
533     public int getTotalCommentPosts() {
534        int commentPosts = 0;
535        for (Post post : posts) { // Iterate through posts
536           if (post.isExists()) { // Check if post still exists
537              if (post.parentId > 0 && !post.endorsedPost) { // Check if post is not an endorsed and is a
                      comment
538                 commentPosts++; // Increment when conditions met
539              }
540           }
541        }
542        return commentPosts;
543     }
544
545     @Override
546     public int getMostEndorsedPost() {
547        Post mostEndorsedPost = new Post(); // Create new object called mostEndorsedPost
548        mostEndorsedPost.endorsements = 0;
549        for (Post post : posts) { // Iterate through posts
550           if (post.isExists()) { // Check if post still exists
551              // Compare mostEndorsedPost's endorsement attribute with currently iterated
552              // Post's endorsements
553              // attribute
554              // If greater update mostEndorsedPost to be the currently iterated Post object
555              if (post.endorsements > mostEndorsedPost.endorsements) {
556                 mostEndorsedPost = post;
557              }
558           }
559        }
560        return mostEndorsedPost.id;
561     }
562
563     @Override
564     public int getMostEndorsedAccount() {
565        int mostEndorsedAccountId = 0;
566        for (Account account : accounts) { // Iterated through accounts
567           if (account.isExists()) { // Check if account still exists
568
569              // Compare endorsement counts of all existing Account objects in accounts
570              // Keep updating mostEndorsedAccountId to store the id of the object with
571              // the highest value for the attribute endorsementCount
572              if (account.endorsementCount > mostEndorsedAccountId) {
573                 mostEndorsedAccountId = account.getId();
574              }
575           }
576        }
577        return mostEndorsedAccountId;
578     }
579
580     @Override
581     public void erasePlatform() {
582
583        // Resets variables to starting defaults and clears both arraylists
584        accountId = 1;
```

```
585         postId = 1;
586         accounts.clear();
587         posts.clear();
588 
589     }
590 
591     @Override
592     public void savePlatform(String filename) throws IOException {
593         if (!filename.endsWith(".ser")) {
594             filename = filename.concat(".ser"); // Ensures files are of a .ser format
595         }
596         try {
597 
598             // Creating new arraylist called platform and adding posts and accounts to it
599             ArrayList<Object> platform = new ArrayList<Object>();
600             platform.add(accounts);
601             platform.add(posts);
602 
603             // Writing platform to a file
604             FileOutputStream fos = new FileOutputStream(filename);
605             ObjectOutputStream oos = new ObjectOutputStream(fos);
606             oos.writeObject(platform);
607             oos.close();
608 
609         } catch (IOException e) {
610             e.printStackTrace();
611             throw new IOException();
612         }
613 
614     }
615 
616     @SuppressWarnings("unchecked")
617     @Override
618     public void loadPlatform(String filename) throws IOException, ClassNotFoundException {
619         if (!filename.endsWith(".ser")) {
620             filename = filename.concat(".ser"); // Ensures files are of a .ser format
621         }
622         try {
623 
624             // Reading data from file
625             FileInputStream readData = new FileInputStream(filename);
626             ObjectInputStream readStream = new ObjectInputStream(readData);
627 
628             // Data that was read is stored in an object arraylist called platform
629             ArrayList<Object> platform = (ArrayList<Object>) readStream.readObject();
630             readStream.close();
631 
632             // Elements of platform stored as "sub-arraylists" accounts and posts
633             accounts = (ArrayList<Account>) platform.get(0);
634             posts = (ArrayList<Post>) platform.get(1);
635 
636         } catch (IOException e) {
637             e.printStackTrace();
638             throw new IOException();
639 
```

```
640         } catch (ClassNotFoundException e) {
641             e.printStackTrace();
642             throw new ClassNotFoundException();
643         }
644 
645     }
646 
647 }
```