```java
package socialmedia;

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

public class SocialMedia implements SocialMediaPlatform {
    private ArrayList<SocialAccount> accountList = new ArrayList<SocialAccount>();
    private ArrayList<SocialPost> postList = new ArrayList<SocialPost>();

    //--- ACCOUNT METHODS ----

    public int createAccount(String handle, String description) throws IllegalHandleException, InvalidHandleException {
        if(validHandleCheck(handle)){ //Check if the handle is valid
            if(legalHandleCheck(handle)){ //Check if the handle is legal
                SocialAccount newAccount = new SocialAccount(handle, description); //Create an account with a description
                accountList.add(newAccount); //Add the new account to the list
                return newAccount.getID(); //Return the accounts ID
            } else {
                throw new IllegalHandleException(); //Throw an exception if the handle is illegal
            }
        } else {
            throw new InvalidHandleException(); //Throw an exception if the handle is invalid
        }
    }

    public int createAccount(String handle) throws IllegalHandleException, InvalidHandleException {
        if(validHandleCheck(handle)){ //Check if the handle is valid
            if(legalHandleCheck(handle)){ //Check if the handle is legal
```

```
29                  SocialAccount newAccount = new
   SocialAccount(handle); //Create an account with a
   description
30                  accountList.add(newAccount); //Add
   the new account to the list
31                  return newAccount.getID(); //Return
   the accounts ID
32              } else {
33                  throw new IllegalHandleException();
   //Throw an exception if the handle is illegal
34              }
35          } else {
36              throw new InvalidHandleException(); //
   Throw an exception if the handle is invalid
37          }
38      }
39
40      /**
41      This method checks if a handle is valid (non
   empty, not over 30 characters, and contains no
   whitespaces), returning false if it is invalid.
42
43      @param handle - the handle you wish to check
44      @return boolean on true or false based on whether
    is passes the checks
45      */
46      private boolean validHandleCheck(String handle){
47          if(handle.length() != 0 && handle.length
   () <= 30){ //Check if the handle is not empty and
   less than 30 characters
48              for(int index = 0; index < handle.length
   (); index++){ //For every letter in the handle
49                  if(handle.charAt(index) == ' '){ //
   Check if it is a whitespace
50                      return false; //If it is, return
   false
51                  }
52              }
53              return true; //If you make it through the
    loop, then return true
54          } else {
55              return false; //If the handle length is 0
    or over 30, then return false
56          }
```

```
57      }
58
59      /**
60      This method checks if a handle is legal (is not
    an already used handle) returning false if it is
    illegal.
61
62      @param handle - the handle you wish to check
63      @return boolean on true or false based on
    whether is passes the checks
64      */
65      private boolean legalHandleCheck(String handle){
66          for (SocialAccount currentAccount :
    accountList){ //For every account in the accountList
67              if(currentAccount.getHandle() == handle
    ){ //Check if its handle is the same as the new
    handle
68                  return false; //If it is, return
    false
69              }
70          }
71          return true; //If you make it through the
    loop, there are no duplicates, so return true
72      }
73
74      /**
75      This method checks the accountList for an
    account with a matching handle, returning the index
    in the list, or throwing a
    HandleNotRecognisedException if there is no match
76
77      @param handle - the handle to look for
78      @return the index of the account within the
    accountList that has the matching handle, or -1 if
    there are no matches
79      */
80      private int findHandleIndex(String handle)
    throws HandleNotRecognisedException {
81          for (int index = 0; index < accountList.size
    (); index++){ //For every account in the accountList
82              if(accountList.get(index).getHandle
    () == handle){ //Check if its handle is the same as
    the inputted handle
83                  return index; //Return the index
```

```
 83 where the match occurs
 84             }
 85         }
 86         return -1; //Return -1 if you have not found
     a matching handle
 87     }
 88 
 89     /**
 90     This method checks the accountList for an
    account with a matching id, returning the index in
    the list, or throwing a
    AccountIDNotRecognisedException if there is no match
 91 
 92     @param id - the id to look for
 93     @return the index of the account within the
    accountList that has the matching handle, or -1 if
    there are no matches
 94     */
 95     private int findIDIndex(int id){
 96         for (int index = 0; index < accountList.size
    (); index++){ //For every account in the accountList
 97             if(accountList.get(index).getID() == id
    ){ //Check if its id is the same as the inputted id
 98                 return index; //Return the index
    where the match occurs
 99             }
100         }
101         return -1;
102     }
103 
104     public void removeAccount(String handle) throws
    HandleNotRecognisedException { //Remove account by a
     handle
105         int index = findHandleIndex(handle);
106         if(index != -1){
107             int deleteIndex = 0;
108             int accountID = accountList.get(index).
    getID(); //Get the account ID of the handle
109             for(SocialPost post : postList){ //For
    every post in the postlist
110                 if(post.getHandleID() == accountID
    ){ //If the posts handleID is the same as the
    accountID
111                     deleteChildrenAndParentIDs(post
```

```
111 ); //Handle related posts based on the deletion of
    the post
112                     postList.remove(deleteIndex); //
    Remove the post from the list
113                 } else {
114                     deleteIndex++; //Increment the
    deleteIndex by 1 if there is no match
115                 }
116             }
117             accountList.remove(index);
118         } else {
119             throw new HandleNotRecognisedException
    ();
120         }
121     }
122 
123     public void removeAccount(int id) throws
    AccountIDNotRecognisedException { //Remove account
    by ID
124         int index = findIDIndex(id);
125         if(index != -1){
126             ArrayList<SocialPost> postsToRemove =
    new ArrayList<SocialPost>();
127             for(SocialPost post : postList){ //For
    every post in the postlist
128                 if(post.getHandleID() == id){ //If
    the posts handleID is the same as the id
129                     postsToRemove.add(post); //Add
    the post to a list of posts to remove
130                 }
131             }
132             for(SocialPost deletingPost :
    postsToRemove){
133                 deleteChildrenAndParentIDs(
    deletingPost);
134             }
135             postList.removeAll(postsToRemove);
136             accountList.remove(index);
137         } else {
138             throw new
    AccountIDNotRecognisedException();
139         }
140     }
141 
```

```
142     public void updateAccountDescription(String handle, String description) throws HandleNotRecognisedException { //Update description by a handle
143         int index = findHandleIndex(handle);
144         if(index != -1){
145             accountList.get(index).setDescription(description);
146         } else {
147             throw new HandleNotRecognisedException();
148         }
149     }
150 
151     public void updateAccountDescription(int id, String description) throws AccountIDNotRecognisedException { //Update description by an ID
152         int index = findIDIndex(id);
153         if(index != -1){
154             accountList.get(index).setDescription(description);
155         } else {
156             throw new AccountIDNotRecognisedException();
157         }
158     }
159 
160     public void changeAccountHandle(String oldHandle, String newHandle) throws HandleNotRecognisedException, IllegalHandleException, InvalidHandleException {
161         int index = findHandleIndex(oldHandle); //Find the index of the matching oldHandle
162         if(index != -1){ //If a matching index is found
163             if(validHandleCheck(newHandle)){ //Check if the new handle is valid
164                 if(legalHandleCheck(newHandle)){ //Check if the new handle is legal
165                     accountList.get(index).setHandle(newHandle); //Set the handle at the index to the new handle
166                 } else {
```

```
167                     throw new IllegalHandleException
    (); //Throw an exception if the handle is illegal
168                 }
169             } else {
170                 throw new InvalidHandleException();
    //Throw an exception if the handle is invalid
171             }
172         }
173     }
174 
175     public String showAccount(String handle) throws
    HandleNotRecognisedException{
176         int index = findHandleIndex(handle); //Find
    the index of the account
177         String accountInfo = "";
178         if(index != -1){ //If there is an account
179             SocialAccount account = accountList.get(
    index); //Get the account from the list
180             accountInfo = "ID: " + account.getID() +
181                 " Handle: " + account.getHandle() +
182                 " Description: " + account.
    getDescription() +
183                 " Post count: " + countTotalPosts(
    account) +
184                 " Endorse count: " +
    getAccountEndorsements(account);
185         } else {
186             throw new HandleNotRecognisedException
    ();
187         }
188         return accountInfo;
189     }
190 
191     /**
192     This method takes a SocialAccount and counts the
     number of posts in the postList whose handleID
    matches the accountID
193 
194     @param account - the account whose total posts
    we are counting
195     @return count - the number of posts made by the
    account
196     */
197     private int countTotalPosts(SocialAccount
```

```
197 account){
198         int count = 0;
199         int accountID = account.getID();
200         for(SocialPost post : postList){ //For every
     post in the postList
201             if(accountID == post.getHandleID()){ //
    Check if post was created by the account
202                 count++;
203             }
204         }
205         return count;
206     }
207 
208     //--- END OF ACCOUNT METHODS ---
209 
210     //--- POST METHODS ---
211 
212     public int createPost(String handle, String
    message) throws HandleNotRecognisedException,
    InvalidPostException {
213         int index = findHandleIndex(handle); //Find
    the index of the handle to use
214         if(index != -1){ //If an index was found
215             int accountID = accountList.get(index).
    getID(); //Get the id of the handle
216             if(message.length() != 0 &&  message.
    length() <= 100){ //If the message is between 0 and
    100 characters
217                 SocialPost newPost = new SocialPost(
    accountID, message); //Make a new message using the
    accountID, and the message
218                 postList.add(newPost); //Add it to
    the list
219                 return newPost.getPostID(); //Return
     the posts ID
220             } else {
221                 throw new InvalidPostException(); //
    Throw an exception if the post is invalid
222             }
223         } else {
224             throw new HandleNotRecognisedException
    ();
225         }
226     }
```

```
227
228     /**
229     This method checks the list of posts, checking
    the id, and returning the index of the matching post
     within the list, or throwing an exception if it is
    not present
230
231     @param id - The id of the post you wish to find
232     @return return the index of the matching post
    within the list, or -1 if there is no matching post
233     */
234     private int findPostIndex(int id){
235         for(int index = 0; index < postList.size();
    index++){ //For every post within the list
236             if(id == postList.get(index).getPostID
    ()){ //Check if the post ID is the same as the id to
     search for
237                 return index; //Return the index if
    it is
238             }
239         }
240         return -1;
241     }
242
243     public int endorsePost(String handle, int id)
    throws HandleNotRecognisedException,
    PostIDNotRecognisedException,
    NotActionablePostException {
244         int accIndex = findHandleIndex(handle); //
    Find the index of the handle
245         if(accIndex != -1){ //If an index was found
246             int accountID = accountList.get(accIndex
    ).getID(); //Get the ID of the account making the
    endorsement
247             int postIndex = findPostIndex(id); //
    Find the index of the post to endorse
248             if(postIndex != -1){ //If a post was
    found
249                 SocialPost origPost = postList.get(
    postIndex);
250                 if((origPost instanceof Endorsement
    ) == false){ //If the post to endorse is not an
    Endorsement
251                     int origID = origPost.
```

```
251 getHandleID(); //Find the ID of the original post
252                     int origAccIndex = findIDIndex(
    origID); //Get the account index in the list based
    on the original accounts ID
253                     String origHandle = "";
254                     if(origAccIndex != -1){
255                         origHandle = accountList.get
    (origAccIndex).getHandle(); //Get the handle from
    the account within the list based on index
256                     } else {
257                         origHandle = "The author of
    this post cannot be found";
258                     }
259                     String endorseString =
    Endorsement.makeEndorsementString(origPost,
    origHandle); //This handle needs to be the handle of
     the original post
260                     SocialPost newEndorse = new
    Endorsement(origPost.getPostID(), accountID,
    endorseString); //Create a new endorsement with the
    PostID, the accountID, and the endorsement string
261                     origPost.addChildID(newEndorse.
    getPostID()); //Add the id of the endorsement to the
     childID list of the original post
262                     postList.add(newEndorse); //Add
    the endorsement to the post list
263                     return newEndorse.getPostID();
264                 } else {
265                     throw new
    NotActionablePostException();
266                 }
267             } else {
268                 throw new
    PostIDNotRecognisedException();
269             }
270         } else {
271             throw new HandleNotRecognisedException
    ();
272         }
273     }
274
275     public int commentPost(String handle, int id,
    String message) throws HandleNotRecognisedException
    , PostIDNotRecognisedException,
```

```
275 NotActionablePostException, InvalidPostException{
276         int accIndex = findHandleIndex(handle); //
    Find the index of the handle
277         if(accIndex != -1){ //If an index was found
278             int accountID = accountList.get(accIndex
    ).getID(); //Get the ID of the account making the
    comment
279             int postIndex = findPostIndex(id); //
    Find the index of the post to comment on
280             if(postIndex != -1){ //If a post was
    found
281                 SocialPost origPost = postList.get(
    postIndex); //Get the original post from the list
282                 if((origPost instanceof Endorsement
    ) == false){ //If the post to comment on is not an
    Endorsement
283                     if(message.length() != 0 ||
    message.length() <= 100){ //If the comment message
    is not empty and under 100 characters
284                         SocialPost newComment = new
    Comment(origPost.getPostID(), accountID, message);
    //Create a new comment using the original post ID,
    account ID and the message
285                         origPost.addChildID(
    newComment.getPostID()); //Add the comments id to
    the list of child ids of the original post
286                         postList.add(newComment);
287                         return newComment.getPostID
    ();
288                     } else {
289                         throw new
    InvalidPostException();
290                     }
291                 } else {
292                     throw new
    NotActionablePostException();
293                 }
294             } else {
295                 throw new
    PostIDNotRecognisedException();
296             }
297         } else {
298             throw new HandleNotRecognisedException
    ();
```

```
299         }
300     }
301 
302     public void deletePost(int id) throws
    PostIDNotRecognisedException {
303         int postIndex = findPostIndex(id); //Get the
     index of the post to delete
304         if(postIndex != -1){ //If a post is found
305             SocialPost postToDel = postList.get(
    postIndex); //Get the post you wish to delete
306             deleteChildrenAndParentIDs(postToDel);
307             postList.remove(postIndex);
308         } else {
309             throw new PostIDNotRecognisedException
    ();
310         }
311     }
312 
313     /**
314     This method is used during the deletion of a
    post, and takes the list of ChildIDs of a given post
    .
315     Comments on the post are set as orphans, and
    endorsements are deleted.
316     If the post is a child of another post, then the
     ID of the deleted post is removed from the parents
    ChildID list
317 
318     @param postToDel - the SocialPost you are
    deleting
319     */
320     private void deleteChildrenAndParentIDs(
    SocialPost postToDel){
321         ArrayList<Integer> childIDList = postToDel.
    getChildIDs(); //Get a list of all the child ID's
    related to the post to delete
322         for(int childID : childIDList){ //For every
    child ID related to the post to delete
323             SocialPost currentPost = postList.get(
    findPostIndex(childID)); //Find the post within the
    postList related to the ID
324             if(currentPost instanceof Endorsement){
    //If the post is an endorsement
325                 postList.remove(findPostIndex(
```

```
325 currentPost.getPostID())); //Remove the endorsement
    based on its index
326             } else { //If the post is a comment
327                 Comment currentComment = (Comment)
    currentPost;
328                 currentComment.setOrphan(); //Set
    the child as an orphan
329             }
330         }
331         if(postToDel instanceof Endorsement){ //If
    the post is an endorsement
332             Endorsement endorsement = (Endorsement)
    postToDel; //Downcast the postToDelete as an
    endorsement
333             int parentIndex = findPostIndex(
    endorsement.getParentID()); //Get the index of its
    parent
334             postList.get(parentIndex).removeChildID(
    endorsement.getPostID()); //Remove the child ID from
     the parents ChildIDList within the postList
335         } else if (postToDel instanceof Comment) {
    //If the post is a comment
336             Comment comment = (Comment)postToDel; //
    Downcast the postToDelete as a comment
337             int parentIndex = findPostIndex(comment.
    getParentID()); //Get the index of its parent
338             if(parentIndex != -1){ //If the child
    has a parent
339                 postList.get(parentIndex).
    removeChildID(comment.getPostID()); //Remove the
    child ID from the parents ChildIDList within the
    postList
340             }
341         }
342     }
343 
344     public String showIndividualPost(int id) throws
    PostIDNotRecognisedException {
345         int postIndex = findPostIndex(id); //Find
    the index of the post with the inputted id
346         if(postIndex != -1){
347             SocialPost postToShow = postList.get(
    postIndex); //Get the post to show from the list
348             return createPostString(postToShow);
```

```
349         } else {
350             throw new PostIDNotRecognisedException
    ();
351         }
352     }
353 
354     /**
355     This method takes a SocialPost and creates a
    string based on its information, returning it as a
    string
356 
357     @param postToShow - the posts information you
    wish to create from a string
358     @return makeString - the string containing
    information about the post
359     */
360     private String createPostString(SocialPost
    postToShow){
361         int accountIndex = findIDIndex(postToShow.
    getHandleID()); //Get the index of the authors
    account from the posts ID
362         String accountHandle = "";
363         if(accountIndex != -1){
364             accountHandle = accountList.get(
    accountIndex).getHandle(); //Get the handle from the
     account list, based on the id attached to the post
365         } else {
366             accountHandle = "The author of this post
     cannot be found"; //Mention that an author cannot
    be found if there is no matching account
367         }
368         String makeString = "ID: " + postToShow.
    getPostID() +
369             " Account: " + accountHandle +
370             " No. endorsements: " +
    getPostEndorsementNumber(postToShow) +
371             "  | No. comments: " +
    getPostCommentNumber(postToShow) + " " +
372             postToShow.getMessage();
373         return makeString;
374     }
375 
376     /**
377     This method takes a SocialPost and counts the
```

```
377 number of comments related to it
378
379     @param post - the post to count the number of
    comments for
380     @return count - the number of comments related
    to the post
381     */
382     private int getPostCommentNumber(SocialPost post
    ){
383         ArrayList<Integer> childList = post.
    getChildIDs(); //Get a list of childIDs from the
    list
384         int count = 0; //Begin counting the number
    of comments
385         for (int childID : childList){ //For every
    child ID
386             int index = findPostIndex(childID); //
    Find the index of the post ID
387             if(postList.get(index) instanceof
    Comment){ //If the post at the index of the id is a
    comment
388                 count++; //Add one to the count
389             }
390         }
391         return count;
392     }
393
394     /**
395     This method takes a SocialPost and counts the
    number of endorsements it has recieved
396
397     @param post - the post to count the number of
    endorsements for
398     @return count - the number of endorsements the
    post has recieved
399     */
400     private int getPostEndorsementNumber(SocialPost
    post){
401         ArrayList<Integer> childList = post.
    getChildIDs(); //Get a list of childIDs from the
    list
402         int count = 0; //Begin counting the number
    of endorsements
403         for (int childID : childList){ //For every
```

```
403 child ID
404             int index = findPostIndex(childID); //
    Find the index of the post ID
405             if(postList.get(index) instanceof
    Endorsement){ //If the post at the index of the id
    is an endorsement
406                 count++; //Add one to the count
407             }
408         }
409         return count;
410     }
411 
412     public StringBuilder showPostChildrenDetails(int
     id) throws PostIDNotRecognisedException,
    NotActionablePostException {
413         int index = findPostIndex(id);
414         StringBuilder totalString = new
    StringBuilder("");
415         if(index != -1){
416             SocialPost basePost = postList.get(index
    );
417             if((basePost instanceof Endorsement) ==
    false){
418                 appendInfo(basePost, 0, totalString
    );
419             } else {
420                 throw new NotActionablePostException
    ();
421             }
422         } else {
423             throw new PostIDNotRecognisedException
    ();
424         }
425         return totalString;
426     }
427 
428 
429     private void appendInfo(SocialPost post, int
    depth, StringBuilder builder){
430         String postString = createPostString(post);
431         String buffer = "|";
432         for(int depthCount = 0; depthCount < depth;
    depthCount++){
433             buffer = "   " + buffer;
```

```
434         }
435         String fullPostString = buffer + postString
     + "\n";
436         builder.append(fullPostString);
437         for(int childPostID : post.getChildIDs()){
438             int childIndex = findPostIndex(
    childPostID);
439             if(childIndex != -1){
440                 SocialPost postToAppend = postList.
    get(childIndex);
441                 if((postToAppend instanceof
    Endorsement) == false){
442                     appendInfo(postToAppend, depth
     + 1, builder);
443                 }
444             }
445         }
446     }
447 
448     //--- END OF POST METHODS ---
449 
450     //--- ANALYTIC METHODS ---
451 
452     public int getNumberOfAccounts(){return
    accountList.size();}
453 
454     public int getTotalOriginalPosts(){
455         int count = 0;
456         for(SocialPost post : postList){ //For every
     post in the postList
457             if((post instanceof Endorsement) || (
    post instanceof Comment)) == false){ //If the post
    is not an endorsement or a comment
458                 count++;
459             }
460         }
461         return count;
462     }
463 
464     public int getTotalEndorsmentPosts(){
465         int count = 0;
466         for(SocialPost post : postList){ //For every
     post in the postList
467             if(post instanceof Endorsement){ //If
```

```
467 the post is an endorsement
468                     count++;
469                 }
470             }
471             return count;
472         }
473 
474         public int getTotalCommentPosts(){
475             int count = 0;
476             for(SocialPost post : postList){ //For every
     post in the postList
477                 if(post instanceof Comment){ //If the
    post is an comment
478                     count++;
479                 }
480             }
481             return count;
482         }
483 
484         public int getMostEndorsedPost(){
485             int largestID = 0;
486             int largestCount = 0;
487             for(SocialPost post : postList){ //For every
     post
488                 if((post instanceof Endorsement) ==
    false){ //Check that it is not an endorsement
489                     int currentCount =
    getPostEndorsementNumber(post); //Count the number
    of endorsements for a given post
490                     if(currentCount > largestCount){ //
    If the post has more endorsements than the current
    leader
491                         largestID = post.getPostID(); //
    Set the id of the new leader
492                         largestCount = currentCount; //
    Set the count of the new leader
493                     }
494                 }
495             }
496             return largestID;
497         }
498 
499         public int getMostEndorsedAccount(){
500             int largestID = 0;
```

```
501         int largestCount = 0;
502         for(SocialAccount account : accountList){ //
    For every account in the list
503             int currentCount =
    getAccountEndorsements(account); //Get its total
    number of endorsements
504             if(currentCount > largestCount){ //If
    the currentCount is greater than the current leader
505                 largestID = account.getID(); //Set
    the id of the new leader
506                 largestCount = currentCount; //Set
    the count of the new leader
507             }
508         }
509         return largestID;
510     }
511 
512     /**
513     This method takes an account and counts the
    number of endorsements it has recieved on its posts
514 
515     @param account - the account to count the number
     of endorsements for
516     */
517     private int getAccountEndorsements(SocialAccount
     account){
518         int count = 0;
519         int accountID = account.getID(); //Get the
    accounts ID
520         for(SocialPost post : postList){ //For every
     post in the postList
521             if(post instanceof Endorsement){ //If
    the post is an endorsement
522                 Endorsement endorsement = (
    Endorsement)post; //Downcast to its endorsement
523                 int parentID = endorsement.
    getParentID(); //Find the parent of the endorsement
524                 SocialPost parentPost = postList.get
    (findPostIndex(parentID)); //Get the parent from the
     list
525                 if(accountID == parentPost.
    getHandleID()){ //Check if the endorsed post was
    created by the account
526                     count++;
```

```
527                 }
528             }
529         }
530         return count;
531     }
532 
533     //--- END OF ANALYTIC METHODS ---
534 
535     //--- MANAGEMENT RELATED METHODS ---
536 
537     public void erasePlatform(){
538         postList.clear(); //Clear the account list
539         accountList.clear(); //Clear the post list
540         SocialAccount.reset(); //Reset the
    SocialAccount counter
541         SocialPost.reset(); //Reset the SocialPost
    counter
542     }
543 
544     public void savePlatform(String filename) throws
     IOException {
545         ObjectOutputStream out = null;
546         try{
547             out = new ObjectOutputStream(new
    FileOutputStream(filename)); //Create an
    ObjectOutputStream
548             out.writeObject(postList); //Write the
    postList
549             out.writeObject(accountList); //Write
    the accountList
550         } catch (IOException e) {
551         } finally {
552             if(out != null){
553                 out.close(); //Close the
    OutputStream if it is open
554             }
555         }
556     }
557 
558     public void loadPlatform(String filename) throws
     IOException, ClassNotFoundException {
559         ObjectInputStream in = null;
560         try{
561             in = new ObjectInputStream(new
```

```
561 FileInputStream(filename));
562             Object obj = in.readObject();
563             if(obj instanceof ArrayList){
564                 postList = (ArrayList)obj;
565             }
566             obj = in.readObject();
567             if(obj instanceof ArrayList){
568                 accountList = (ArrayList)obj;
569             }
570         } catch(IOException e) {
571         } finally {
572             if(in != null){
573                 in.close();
574             }
575         }
576     }
577 
578     //--- END OF MANAGEMENT RELATED METHODS ---
579 }
```

```java
package socialmedia;

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

public class SocialPost implements Serializable{
    protected static int postCount = 0;
    protected int id;
    protected String message;
    protected int handleID;
    protected ArrayList<Integer> childIDs = new ArrayList<Integer>();

    public SocialPost(int newHandleID, String message){
        this.handleID = newHandleID;
        this.message = message;
        this.id = postCount;
        postCount++;
    }

    public void addChildID(int newChildID){this.childIDs.add(newChildID);}
    public int getPostID(){return this.id;}
    public int getHandleID(){return this.handleID;}
    public String getMessage(){return this.message;}
    public ArrayList<Integer> getChildIDs(){return this.childIDs;}
    public void removeChildID(int id){
        for(int index = 0; index < childIDs.size(); index++){ //For every child ID in the list
            if(id == childIDs.get(index)){ //If the child ID is the same as the ID to be removed
                this.childIDs.remove(index);
            }
        }
    }
    public static void reset(){postCount = 0;}
}

class Comment extends SocialPost{
    private int parentID;

    public Comment(int newParent, int newHandleID,
```

```
38 String message){
39         super(newHandleID, message);
40         this.parentID = newParent;
41     }
42 
43     public void setOrphan(){this.parentID = -1;}
44     public int getParentID(){return this.parentID;}
45 }
46 
47 class Endorsement extends SocialPost{
48     private int parentID;
49 
50     public Endorsement(int newParent, int newHandleID
   , String message){
51         super(newHandleID, message);
52         this.parentID = newParent;
53     }
54 
55     public static String makeEndorsementString(
   SocialPost postToEndorse, String handle){ //CHANGE
   THIS SO HANDLE POINTS TO ENDORSED HANDLE
56         return "EP@" + handle + ": " + postToEndorse.
   getMessage();
57     }
58     public int getParentID(){return this.parentID;}
59 }
```

```java
package socialmedia;

import java.io.Serializable;

public class SocialAccount implements Serializable{
    private static int accCount = 0;
    private int id;
    private String accHandle;
    private String accDesc;

    public SocialAccount(String handle, String description){
        this.accHandle = handle;
        this.accDesc = description;
        this.id = accCount;
        accCount++;
    }

    public SocialAccount(String handle){
        this.accHandle = handle;
        this.id = accCount;
        accCount++;
    }

    public String getHandle(){return this.accHandle;}
    public int getID(){return this.id;}
    public String getDescription(){return this.accDesc;}
    public void setDescription(String description){this.accDesc = description;}
    public void setHandle(String handle){this.accHandle = handle;}
    public static void reset(){accCount = 0;}
}
```