DOCUMENT_HEADING

6 files

SocialMedia 2\src\socialmedia\SocialMedia.java
SocialMedia 2\src\socialmedia\Account.java
SocialMedia 2\src\socialmedia\Post.java
SocialMedia 2\src\socialmedia\OriginalPost.java
SocialMedia 2\src\socialmedia\CommentPost.java
SocialMedia 2\src\socialmedia\EndorsementPost.java

SocialMedia 2\src\socialmedia\SocialMedia.java

```java
package socialmedia;

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

public class SocialMedia implements SocialMediaPlatform {



    int universalAccountID = 0;
// variable used to store the next available account ID number
    int universalPostID = 0;
// variable used to store the next available account ID number
    private ArrayList<Account> accountList = new ArrayList<Account>();
// list used to store all of the currently exisitng accounts
    private ArrayList<Post> postList = new ArrayList<Post>();
// list used to store all of the currently exisitng posts



    @Override
    public int createAccount(String handle) throws IllegalHandleException,
InvalidHandleException {
        return createAccount(handle, "");
// calls the other create account function, setting the description field as empty
    }



    @Override
    public int createAccount(String handle, String description) throws
IllegalHandleException, InvalidHandleException {
        for (Account account : accountList) {
// iterates through all the accounts in the account list
            if (account.getUserHandle() == handle) {
// if the account handle matches the handle requested then
                throw new IllegalHandleException("Handle already in use!");
// throw an error saying the handle is already in use
            }
        }

        if (handle.length() > 30) {
// if the inputted handles length is more than 30 characters
            throw new InvalidHandleException("Handle can't be longer than 30
characters!");          // throw an error explaining so
        } else if (handle.contains(" ")) {
// if the inputted handle contains white spaces
            throw new InvalidHandleException("Handle must not contain whitespaces!");
// throw an error explaining so
```

```
        } else if (handle == "") {
// if the handle is empty
            throw new InvalidHandleException("Handle must have one or more characters!");
// throw an error explaining so
        } else {
// if no errors are thrown
            accountList.add(new Account(++universalAccountID, handle, description));
// create an account with the specified handle and description and add it to the list
        }
        return universalAccountID;
// finally, return the ID of the account created
    }


    @Override
    public void removeAccount(int id) throws AccountIDNotRecognisedException {
        boolean idFound = false;
// boolean used to store if the account has been found
        Iterator<Account> accountIterator = accountList.iterator();
// iterator used to remove accounts without error
        Iterator<Post> postIterator = postList.iterator();
// iterator used to remove accounts without error


        while (accountIterator.hasNext()) {
// iterates through all exisitng accounts
            Account account = (Account)accountIterator.next();
// assigns the account as the next account in the list
            if (account.getUserID() == id) {
// if the ID of the account matches the the inputted ID
                while (postIterator.hasNext()) {
// iterates through all exisitng posts
                    Post post = (Post)postIterator.next();
// assigns the post as the next post in the list
                    if (post.getAuthorID(accountList) == id) {
// if the posts author ID matches the current ID
                        postIterator.remove();
// remove the post
                    } else if (post instanceof CommentPost) {
// if the post is an comment
                        CommentPost comment = (CommentPost) post;
// create new variable for commeny
                        if (comment.getParentPostID() == id) {
// if the comment's parent id is that of given id
                            comment.setParentPostID(0);                                    //
set the comments parent to the empty post
                        }
                    } else if (post instanceof EndorsementPost) {
// if the post is an endorsement
                        EndorsementPost endorsement = (EndorsementPost) post;
// create new variable for endorsement
                        if (endorsement.getParentPostID() == id) {
// if the endorsement's parent id is that of given id
                            postIterator.remove();
// remove the endorsement from the list
                        }
                    }
                }
                accountIterator.remove();
// remove that account from the list
                idFound = true;
// set boolean to true as account was found
            }
        }

```

```
        if (!idFound) {
// if the inputted ID does not match an existing account
            throw new AccountIDNotRecognisedException("Account ID not recognised!");
// throw an error saying so
        }
    }



    @Override
    public void removeAccount(String handle) throws HandleNotRecognisedException {
        boolean handleFound = false;
// boolean used to store if the account has been found
        Iterator<Account> accountIterator = accountList.iterator();
// iterator used to remove accounts without error
        Iterator<Post> postIterator = postList.iterator();
// iterator used to remove accounts without error

        while (accountIterator.hasNext()) {
// iterates through all exisitng accounts
            Account account = (Account)accountIterator.next();
// assigns the account as the next account in the list
            if (account.getUserHandle() == handle) {
// if the handle of the account matches the the inputted handle
                while (postIterator.hasNext()) {
// iterates through all exisitng posts
                    Post post = (Post)postIterator.next();
// assigns the post as the next post in the list
                    if (post.getAuthor() == handle) {
// if the posts author ID matches the current ID
                        postIterator.remove();
// remove the post
                    } else if (post instanceof CommentPost) {
// if the post is an comment
                        CommentPost comment = (CommentPost) post;
// create new variable for commeny
                        if (comment.getParentPostID() == account.getUserID()) {
// if the comment's parent id is that of given id
                            comment.setParentPostID(0);                                    //
set the comments parent to the empty post
                        }
                    } else if (post instanceof EndorsementPost) {
// if the post is an endorsement
                        EndorsementPost endorsement = (EndorsementPost) post;
// create new variable for endorsement
                        if (endorsement.getParentPostID() == account.getUserID()) {
// if the endorsement's parent id is that of given id
                            postIterator.remove();
// remove the endorsement from the list
                        }
                    }
                }
                accountIterator.remove();
// remove that account from the list
                handleFound = true;
// set boolean to true as account was found
            }
        }

        if (!handleFound) {
// if the inputted handle does not match an existing account
            throw new HandleNotRecognisedException("Account handle not recognised!");
// throw an error saying so
        }
    }

```

```
117
118     @Override
119     public void changeAccountHandle(String oldHandle, String newHandle) throws
    HandleNotRecognisedException, IllegalHandleException, InvalidHandleException {
120         boolean handleFound = false;
    // boolean to track whether the account has been found
121
122         for (Account account : accountList) {
    // iterates through all the accounts in the account list
123             if (account.getUserHandle() == newHandle) {
    // if the account handle matches the handle requested then
124                 throw new IllegalHandleException("Handle already in use!");
    // throw an error saying the handle is already in use
125             }
126         }
127
128         for (Account account : accountList) {
    // iterates through all exisitng accounts
129             if (account.getUserHandle() == oldHandle) {
    // if the account handle matches the given handle
130                 handleFound = true;
    // set boolean to true as account has been found
131                 if (newHandle.length() > 30) {
    // if the inputted handles length is more than 30 characters
132                     throw new InvalidHandleException("Handle can't be longer than 30
    characters!");  // throw an error explaining so
133                 } else if (newHandle.contains(" ")) {
    // if the inputted handle contains white spaces
134                     throw new InvalidHandleException("Handle must not contain
    whitespaces!");     // throw an error explaining so
135                 } else if (newHandle == "") {
    // if the handle is empty
136                     throw new InvalidHandleException("Handle must have one or more
    characters!");  // throw an error explaining so
137                 } else {
    // if no errors are thrown
138                     account.setUserHandle(newHandle);
    // set the handle of the account to the new given one
139                 }
140             }
141         }
142
143         if (!handleFound) {
    // if the account with the given handle was not found
144             throw new HandleNotRecognisedException("Account handle not recognised!");
    // throw an error saying so
145         }
146     }
147
148
149     @Override
150     public void updateAccountDescription(String handle, String description) throws
    HandleNotRecognisedException {
151         boolean handleFound = false;
    // boolean to track if the account is found
152         for (Account account : accountList) {
    // iterates through all existing accounts
153             if (account.getUserHandle() == handle) {
    // if the account handle matches the given one
154                 handleFound = true;
    // account has been found so set boolean to true
155                 account.setUserDesc(description);
    // set the description of the matching account to the description given
156             }
157         }
```

```
158
159         if (!handleFound) {
// if the account with the given handle was not found
160             throw new HandleNotRecognisedException("Account handle not recognised!");
// throw an error explaining so
161         }
162     }
163
164
165     @Override
166     public String showAccount(String handle) throws HandleNotRecognisedException {
167         boolean handleFound = false;
// boolean to track if the account is found
168         String returnMessage = null;
// string to return
169         int postCount = 0;
// variable for tracking how many posts an account has made
170         int endorsementCount = 0;
// variable for tracking how many endorsments an account has
171
172         for (Account account : accountList) {
// searches through all accounts
173             if (account.getUserHandle() == handle){
// if account handle matches given handle
174                 handleFound = true;
// set handle found to true
175
176                 for (Post post : postList) {
// iterate through all exisitng posts
177                     if (post.getAuthorID(accountList) == account.getUserID()) {
// if the author ID of the post matches the account ID
178                         postCount++;
// increment post count
179                         if (post instanceof OriginalPost) {
// if the post is an original post or comment
180                             endorsementCount += ((OriginalPost)
post).getEndorsementCount(postList);        // add the endorsement count to the total
endorsements
181                         }
182                     }
183                 }
184
185                 returnMessage =
// set return message
186                 "ID: " + account.getUserID() +
// show account ID
187                 "\nHandle: " + account.getUserHandle() +
// show account handle
188                 "\nDescription: " + account.getUserDesc() +
// show account description
189                 "\nPost count: " + postCount +
// show amount of posts
190                 "\nEndorse count: " + endorsementCount;
// show amount of endorsements
191             }
192         }
193
194         if (!handleFound) {
// if the account with the given handle was not found
195             throw new HandleNotRecognisedException("Account handle not recognised!");
// throw an error explaining so
196         }
197         return returnMessage;
// return the account information if found, else return null
198     }
```

```
199
200
201     @Override
202     public int createPost(String handle, String message) throws
HandleNotRecognisedException, InvalidPostException {
203         boolean handleFound = false;
// boolean to track if the account is found
204
205         if (message.length() > 100) {
// if the message is longer than 100 characters
206             throw new InvalidPostException("Message must not be longer than 100
characters!");      // throw an error explaining so
207         } else if (message == "") {
// if the message is empty
208             throw new InvalidPostException("Message must not be empty");
// throw an error explaining so
209         }
210
211         for (Account account : accountList) {
// searches through all accounts
212             if (account.getUserHandle() == handle) {
// if the account handle matches given handke
213                 handleFound = true;
// set handle found to true
214                 postList.add(new OriginalPost(++universalPostID, handle, message));
// create post and add it to the list of all posts
215             }
216         }
217
218         if (!handleFound) {
// if the account with the given handle was not found
219             throw new HandleNotRecognisedException("Account handle not recognised!");
// throw an error explaining so
220         }
221         return universalPostID;
// returns post id
222     }
223
224
225     @Override
226     public int endorsePost(String handle, int id) throws HandleNotRecognisedException,
PostIDNotRecognisedException, NotActionablePostException {
227         boolean handleFound = false;
// boolean to track if the account is found
228         boolean postFound = false;
// boolean to track is the post is found
229         OriginalPost postToEndorse = null;
// variable to pass the endorsed post
230
231         for (Account account : accountList) {
// search all exisitng accounts
232             if (account.getUserHandle() == handle) {
// if the account handle matches given handle
233                 handleFound = true;
// set handle found to true
234             }
235         }
236
237         if (!handleFound) {
// if the account with the given handle was not found
238             throw new HandleNotRecognisedException("Account handle not recognised!");
// throw an error explaining so
239         }
240
```

```
241         for (Post post : postList) {
// searches through all posts
242             if (post.getID() == id) {
// if the post id matches the given id
243                 if (post instanceof OriginalPost) {
// if post is original or comment
244                     postFound = true;
// set post found to true
245                     postToEndorse = (OriginalPost) post;
// set the post to the post to endorse variable
246                 } else if (post instanceof EndorsementPost) {
// throw an error if post is an endorsement
247                     throw new NotActionablePostException("Cannot endorse an endorsement post!");
248                 }
249             }
250         }
251 
252         if (postFound) {
// if the post was found
253             postList.add(new EndorsementPost(++universalPostID, handle, postToEndorse));
// create endorsement and add it to the list of all posts
254         } else {
// else post was not found
255             throw new PostIDNotRecognisedException("Post ID not recognised!");
// throw an error explaining so
256         }
257         return universalPostID;
258     }
259 
260 
261     @Override
262     public int commentPost(String handle, int id, String message) throws HandleNotRecognisedException, PostIDNotRecognisedException, NotActionablePostException, InvalidPostException {
263         boolean handleFound = false;
// boolean to track if the account is found
264         boolean postFound = false;
// boolean to track is the post is found
265 
266         if (message.length() > 100) {
// if the message is longer than 100 characters
267             throw new InvalidPostException("Message must not be longer than 100 characters!");      // throw an error explaining so
268         } else if (message == ""){
// if the message is empty
269             throw new InvalidPostException("Message must not be empty");
// throw an error explaining so
270         }
271 
272         for (Account account : accountList) {
// search all exisitng accounts
273             if (account.getUserHandle() == handle) {
// if the account handle matches given handle
274                 handleFound = true;
// set handle found to true
275             }
276         }
277 
278         if (!handleFound) {
// if the account with the given handle was not found
279             throw new HandleNotRecognisedException("Account handle not Recognised!");
// throw an error explaining so
280         }
281 
```

```
282         for (Post post : postList) {
// searches through all posts
283             if (post.getID() == id) {
// if the post id matches the given id
284                 if (post instanceof OriginalPost) {
// if post is original or comment
285                     postFound = true;
// set post found to true
286                 } else if (post instanceof EndorsementPost) {
// throw an error if post is an endorsement
287                     throw new NotActionablePostException("Cannot endorse an endorsement
post!");
288                 }
289             }
290         }
291 
292         if (postFound) {
// if the post was found
293             postList.add(new CommentPost(++universalPostID, handle, message, id));
// create comment add it to the list of all posts
294         } else {
// else post isnt found
295             throw new PostIDNotRecognisedException("Post ID not recognised!");
// throw an error explaining so
296         }
297         return universalPostID;
298     }
299 
300 
301     @Override
302     public void deletePost(int id) throws PostIDNotRecognisedException {
303         boolean postFound = false;
// boolean used to store if the account has been found
304         Iterator<Post> postIterator = postList.iterator();
// iterator used to remove accounts without error
305 
306         while (postIterator.hasNext()) {
// iterates through all exisitng accounts
307             Post post = (Post)postIterator.next();
// assigns the account as the next account in the list
308             if (post.getID() == id) {
// if the ID of the account matches the the inputted ID
309                 postIterator.remove();
// remove that post from the list
310                 postFound = true;
// set boolean to true as account was found
311             } else if (post instanceof CommentPost) {
// if the post is an comment
312                 CommentPost comment = (CommentPost) post;
// create new variable for commeny
313                 if (comment.getParentPostID() == id) {
// if the comment's parent id is that of given id
314                     comment.setParentPostID(0);                                    //
set the comments parent to the empty post
315                 }
316             } else if (post instanceof EndorsementPost) {
// if the post is an endorsement
317                 EndorsementPost endorsement = (EndorsementPost) post;
// create new variable for endorsement
318                 if (endorsement.getParentPostID() == id) {
// if the endorsement's parent id is that of given id
319                     postIterator.remove();
// remove the endorsement from the list
320                 }
321             }
322         }
```

```
323
324         if (!postFound) {
// if the inputted ID does not match an existing post
325             throw new PostIDNotRecognisedException("Post ID not Recognised!");
// throw an error saying so
326         }
327     }
328
329
330     @Override
331     public String showIndividualPost(int id) throws PostIDNotRecognisedException {
332         boolean postFound = false;
// boolean used to store if the account has been found
333         StringBuilder message = new StringBuilder();
// string used to store message
334
335         for (Post post : postList) {
// searches all posts
336             if (post.getID() == id) {
// if the post id matches the given one
337                 message.append("ID: " + post.getID() + "\n");
// add the ID to the message
338                 message.append("Account: " + post.getAuthor() + "\n");
// add the handle to the message
339                 if (post instanceof OriginalPost) {
// if the post is an original post or a comment
340                     OriginalPost original = (OriginalPost) post;
// change to OriginalPost type
341                     message.append("No. endorsements: " +
original.getEndorsementCount(postList) + " | ");    // add endorsements to message
342                     message.append("No. comments: " + original.getCommentCount(postList) +
"\n");          // add comments to message
343                 }
344                 message.append(post.getMessage());
// add the actual message
345                 postFound = true;
// set boolean to true as account was found
346             }
347         }
348
349         if (!postFound) {
// if the inputted ID does not match an existing post
350             throw new PostIDNotRecognisedException("Post ID not recognised!");
// throw an error saying so
351         }
352         return message.toString();
353     }
354
355
356     @Override
357     public StringBuilder showPostChildrenDetails(int id) throws
PostIDNotRecognisedException, NotActionablePostException {
358         boolean postFound = false;
// boolean used to store if the account has been found
359         boolean postCanHaveChildren = false;
// boolean used to store if the account can have children
360         boolean firstChild = true;
// boolean used to format child posts correctly
361         StringBuilder message = new StringBuilder();
// string used to store message
362
363
364         for (Post post : postList) {
// searches all posts
```

```
365             if (post.getID() == id) {
// if the post id matches the given one
366                 postFound = true;
// set boolean to true as account was found
367                 if (post instanceof OriginalPost) {
// if the post is an original post or a comment
368                     postCanHaveChildren = true;
// set can have children to true
369                 }
370             }
371         }
372 
373         if (!postFound) {
// if the inputted ID does not match an existing post
374             throw new PostIDNotRecognisedException("Post ID not recognised!");
// throw an error saying so
375         } else if (!postCanHaveChildren) {
// if the post is an endorsement
376             throw new NotActionablePostException("Post cannot be an endorsement!");
// throw an error saying so
377         } else {
// else if the post exists and can have children
378             message.append(showIndividualPost(id));
// display the post with the given id
379             for (Post post : postList) {
// search through all exisiting posts
380                 if (post instanceof CommentPost) {
// if the post is a comment
381                     CommentPost childPost = (CommentPost) post;
// create variable for comment
382                     if (childPost.getParentPostID() == id) {
// if the comments parent id is that of the given id
383                         if (firstChild) {
// check if this is the first child of the given post
384                             message.append("\n|");
// if so, append a | to the message for formatting
385                             firstChild = false;
// then set first child to false
386                         }
387                         message.append("\n| > " +
(showPostChildrenDetails(childPost.getID()).toString()) // call this function to display
children of child post
388                         .replace("\n", "\n    "));                                    //
used to keep indentation correctly
389                     }
390                 }
391             }
392         }
393         return message;
394     }
395 
396 
397     @Override
398     public int getNumberOfAccounts() {
399         return accountList.size();
// returns the size off account list, showing number off accounts
400     }
401 
402 
403     @Override
404     public int getTotalOriginalPosts() {
405         int originalPostCount = 0;
// variable for keeping count of all original posts
406         for (Post post : postList) {
// searches all exisitng posts
```

```
407             if (post instanceof OriginalPost && !(post instanceof CommentPost)) {
// if the post is an instance of a original post but not a comment
408                 originalPostCount++;
// increment the counter
409             }
410         }
411         return originalPostCount;
// return the counter value
412     }
413
414
415     @Override
416     public int getTotalEndorsmentPosts() {
417         int endorsementPostCount = 0;
// variable for keeping count of all endorsements
418         for (Post post : postList) {
// searches all exisitng posts
419             if (post instanceof EndorsementPost) {
// if the post is an instance of an endorsement
420                 endorsementPostCount++;
// increment the counter
421             }
422         }
423         return endorsementPostCount;
// return the counter value
424     }
425
426
427     @Override
428     public int getTotalCommentPosts() {
429         int commentPostCount = 0;
// variable for keeping count of all comments
430         for (Post post : postList) {
// searches all exisitng posts
431             if (post instanceof CommentPost) {
// if the post is an instance of a comment
432                 commentPostCount++;
// increment the counter
433             }
434         }
435         return commentPostCount;
// return the counter value
436     }
437
438
439     @Override
440     public int getMostEndorsedPost() {
441         int maxEndorsements = 0;
442         int mostEndorsedPostId = 0;
443
444         for (Post post : postList) {
// iterate through all exisitng posts
445             if (post instanceof OriginalPost) {
// if the post is an original post or a comment
446                 OriginalPost originalPost = (OriginalPost) post;
// create variable for getting endorsements
447                 int endorsements = originalPost.getEndorsementCount(postList);
// get the endorsements of the post
448
449             if (endorsements > maxEndorsements) {
// if the posts endorsements are higher than the max current endorsements
450                 maxEndorsements = endorsements;
// set the max endorsements to the posts endorsement number
451                 mostEndorsedPostId = originalPost.getID();
```

```
// set the max endorsed post id to this posts id
452                 }
453             }
454         }
455         return mostEndorsedPostId;
456     }
457 
458 
459     @Override
460     public int getMostEndorsedAccount() {
461         int maxEndorsements = 0;
462         int mostEndorsedAccountId = 0;
463 
464         for (Account account : accountList) {
// iterate through all accounts
465             int endorsements = 0;
// set endorsements to 0
466 
467             for (Post post : postList) {
// iterate through all exisitng posts
468                 if (post.getAuthorID(accountList) == account.getUserID()) {
// if the author ID of the post matches the account ID
469                     if (post instanceof OriginalPost) {
// if the post is an original post or comment
470                         endorsements += ((OriginalPost)
post).getEndorsementCount(postList);                // add the endorsement count to the
total endorsements
471                     }
472                 }
473             }
474 
475             if (endorsements > maxEndorsements) {
// if the total endorsements is greater than the current max endorsements
476                 maxEndorsements = endorsements;
// set the max endorsements to the total endorsements
477                 mostEndorsedAccountId = account.getUserID();
// set the most endorsed account ID to the account ID
478             }
479         }
480         return mostEndorsedAccountId;
481     }
482 
483 
484     @Override
485     public void erasePlatform() {
486         accountList.clear();
// clear account list
487         postList.clear();
// clear post list
488         universalAccountID = 0;
// set the universal account ID to 0
489         universalPostID = 0;
// set the universal post ID to 0
490     }
491 
492 
493     @Override
494     public void savePlatform(String filename) throws IOException {
495         try (ObjectOutputStream out = new ObjectOutputStream(new
FileOutputStream(filename))) {              // create the output variable
496             out.writeInt(universalAccountID);
// save the account id number
497             out.writeInt(universalPostID);
// save the post id number
```

```
498             out.writeObject(accountList);
// save all the accounts
499             out.writeObject(postList);
// save all the posts
500             out.close();
// close the file
501         }
502     }
503 
504 
505     @Override
506     public void loadPlatform(String filename) throws IOException, ClassNotFoundException {
507         try (ObjectInputStream in = new ObjectInputStream(new FileInputStream(filename)))
{                   // create the input variable
508             erasePlatform();
// clear platform before loading
509 
510             universalAccountID = in.readInt();
// read in the universal account ID
511             universalPostID = in.readInt();
// read in the universal post ID
512 
513             List<?> rawLoadedAccounts = (List<?>) in.readObject();
// read in the list of accounts
514             List<Account> loadedAccounts = new ArrayList<>();
// create a new list of accounts
515             for (Object obj : rawLoadedAccounts) {
// iterate through the list of accounts
516                 loadedAccounts.add((Account) obj);
// add the accounts to the new list
517             }
518             accountList.addAll(loadedAccounts);
// add the new list of accounts to the current list
519 
520             List<?> rawLoadedPosts = (List<?>) in.readObject();
// read in the list of posts
521             List<Post> loadedPosts = new ArrayList<>();
// create a new list of posts
522             for (Object obj : rawLoadedPosts) {
// iterate through the list of posts
523                 loadedPosts.add((Post) obj);
// add the posts to the new list
524             }
525             postList.addAll(loadedPosts);
// add the new list of posts to the current list
526         }
527     }
528 }
529 
530 
```

SocialMedia 2\src\socialmedia\Account.java

```
1  package socialmedia;
2  
3  import java.io.Serializable;
4  
5  public class Account implements Serializable {
6      // Variable Definitions
7      private int userID;
8      private String userHandle;
9      private String userDesc;
```

```
10
11      public Account(int inputID,String inputHandle, String inputDesc){
12          userID = inputID;
13          userHandle = inputHandle;
14          userDesc = inputDesc;
15      }
16
17      public int getUserID(){
18          return userID;
19      }
20
21      public String getUserHandle(){
22          return userHandle;
23      }
24
25      public String getUserDesc(){
26          return userDesc;
27      }
28
29      public void setUserHandle(String inputHandle){
30          userHandle = inputHandle;
31      }
32
33      public void setUserDesc(String inputDesc){
34          userDesc = inputDesc;
35      }
36  }
```

SocialMedia 2\src\socialmedia\Post.java

```
1   package socialmedia;
2
3   import java.io.Serializable;
4   import java.util.ArrayList; // had to add this for getAuthorID
5
6   public class Post implements Serializable {
7       private int postID;
8       private String author;
9       private String message;
10
11      public int getID(){
12          return postID;
13      }
14
15      public String getAuthor(){
16          return author;
17      }
18
19      public String getMessage(){
20          return message;
21      }
22
23      public void setID(int inputID){
24          postID = inputID;
25      }
26
27      public void setAuthor(String inputAuthor){
28          author = inputAuthor;
```

```
    }

    public void setMessage(String inputMessage){
        message = inputMessage;
    }
    public int getAuthorID(ArrayList<Account> accountList) {
        int id = -1;

        for (Account account : accountList) {
            if (account.getUserHandle() == author) {
                id = account.getUserID();
            }
        }
        return id;
    }
}

```

SocialMedia 2\src\socialmedia\OriginalPost.java

```
package socialmedia;

import java.util.ArrayList;

public class OriginalPost extends Post {

    private int commentCount, endorsementCount;

    public OriginalPost(int postID, String author, String message){
        super.setID(postID);
        super.setAuthor(author);
        super.setMessage(message);
    }

    public int getCommentCount(ArrayList<Post> postList){
        commentCount = 0;
        for(Post post : postList){
            if(post instanceof CommentPost){
                CommentPost comment = (CommentPost) post;
                if(comment.getParentPostID() == super.getID()){
                    commentCount += 1;
                }
            }
        }
        return commentCount;
    }

    public int getEndorsementCount(ArrayList<Post> postList){
        endorsementCount = 0;
        for(Post post : postList){
            if(post instanceof EndorsementPost){
                EndorsementPost endorsement = (EndorsementPost) post;
                if(endorsement.getParentPostID() == super.getID()){
                    endorsementCount += 1;
                }
            }
        }
        return endorsementCount;
```

```
    }
}

```

SocialMedia 2\src\socialmedia\CommentPost.java

```
package socialmedia;

public class CommentPost extends OriginalPost {
    private int parentPostID;

    public CommentPost(int postID, String author, String message, int inputParentPostID){
        super(postID, author, message);
        parentPostID = inputParentPostID;
    }

    public int getParentPostID(){
        return parentPostID;
    }

    public void setParentPostID(int inputParentPostID){
        parentPostID = inputParentPostID;
    }
}
```

SocialMedia 2\src\socialmedia\EndorsementPost.java

```
package socialmedia;

public class EndorsementPost extends Post {
    private int parentPostID;

    public EndorsementPost(int postID, String author, Post post){
        parentPostID = post.getID();
        super.setID(postID);
        super.setAuthor(author);
        super.setMessage( "EP@ " + post.getAuthor() + ": " + post.getMessage());
    }
    public int getParentPostID(){
        return parentPostID;
    }
}

```