

# 1 Account.java

```
1 package socialmedia;
2
3 import java.io.Serializable;
4 import java.util.ArrayList;
5 import java.util.List;
6
7 /**
8  * This class stores information of each Account in the social media platform.
9  * Accounts can be created or removed and their description field can be updated.
10 * This class is a superclass of Posts and Comments classes.
11 * @author Aisha
12 * @author Saida
13 *
14 */
15 @SuppressWarnings("serial")
16 public class Account implements Serializable {
17
18     //declaring variables
19     private int ID;
20     private String handle;
21     private String description;
22     private static int lastAccountID = 0;
23     private static ArrayList<Account> accounts = new ArrayList<>();
24     private List<Post> postsList = new ArrayList<>();
25     private List<Endorsement> accountEndorsementsList = new ArrayList<>();
26     private List<Comment> accountCommentsList = new ArrayList<>();
27
28     /**
29      * constructor for objects of class Account.
30      * @param handle
31      * @param description
32      */
33     public Account(String handle, String description) {
34         this.handle = handle;
35         this.description = description;
36         this.ID = ++lastAccountID;
37     }
38
39     /**
40      * one-arg constructor for objects of class Account.
41      * @param handle
42      */
43     public Account(String handle) {
44         this.handle = handle;
45         this.ID = ++lastAccountID;
46     }
47
48     //setter for description
49     public void setDescription(String description) {
50         this.description = description;
51     }
52 }
```

```

53 //setter for handle
54 public void setHandle(String handle) {
55     this.handle = handle;
56 }
57
58 //getter for handle
59 public String getHandle() {
60     return handle;
61 }
62
63 //getter for the static list of accounts
64 public static ArrayList<Account> getAccountsList() {
65     return accounts;
66 }
67
68 //getter for description
69 public String getDescription() {
70     return description;
71 }
72
73 //getter for the list of posts of each account
74 public List<Post> getEachPostList() {
75     return postsList;
76 }
77
78 //getter for the list of endorsements of each account
79 public List<Endorsement> getAccountEndorsementsList() {
80     return accountEndorsementsList;
81 }
82
83 //getter for the list of comments of each account
84 public List<Comment> getAccountCommentList() {
85     return accountCommentsList;
86 }
87
88 /**
89  * This method adds an account into the static accounts list.
90  */
91 public static void addAccount(Account account){
92     accounts.add(account);
93 }
94
95 /**
96  * This method adds an endorsement to the endorsement list of each account.
97  */
98 public void addAccountEndorsement(Endorsement endorsement) {
99     accountEndorsementsList.add(endorsement);
100 }
101
102 /**
103  * This method returns the ID of the Account.
104  * @return ID
105  */
106 public int getID(){
107     return ID;

```

```

108     }
109
110     /**
111      * This method displays the Account.
112      * @return toString of Account
113      */
114     public String toString() {
115         return "Account[name = " + handle + ", description = " +
116             description + ", ID = " + ID + "]";
117     }
118 }

```

## 2 Comment.java

```

1  package socialmedia;
2  import java.util.ArrayList;
3  import java.util.List;
4
5  /**
6   * This class stores information on each Endorsement in the social media platform.
7   * Endorsements automatically replicate the endorsed message. The endorsed message cannot be endorsed again.
8   * This class is a subclass of Post class.
9   * @author Aisha
10  * @author Saida
11  */
12  @SuppressWarnings("serial")
13  public class Comment extends Post {
14
15      //declaring variables
16      private static ArrayList<Comment> comments = new ArrayList<>();
17      private String comment;
18      private int commentID;
19      private static int lastCommentID = Endorsement.getLastEndorsementID();
20      private List <Comment> commentsCommentsList = new ArrayList<>();
21
22      /**
23       * a constructor for objects in class Comment.
24       * @param handle
25       * @param id
26       * @param message
27       */
28      public Comment(String handle, int id, String message) {
29          super(handle, id);
30          this.comment = message;
31          this.commentID = ++lastCommentID;
32      }
33
34      //getter for the static list of comments
35      public static ArrayList<Comment> getCommentsList() {
36          return comments;
37      }
38
39      //getter for comment
40      public String getComment() {

```

```

41     return comment;
42 }
43
44 //getter for commentID
45 public int getCommentID() {
46     return commentID;
47 }
48
49 //getter for the list of replies of each comment.
50 public List<Comment> getCommentsCommentsList() {
51     return commentsCommentsList;
52 }
53
54 /**
55  * This method adds a comment to the static list of comments.
56  */
57 public static void addComment(Comment comment) {
58     comments.add(comment);
59 }
60
61 /**
62  * This method displays the Comment.
63  */
64 public String toString() {
65     return "@Comment " + getHandle() + ": " + comment;
66 }
67
68 }
69 }

```

### 3 Endorsement.java

```

1 package socialmedia;
2
3 import java.util.ArrayList;
4
5 /**
6  * This class stores information on each Endorsement in the social media platform.
7  * Endorsements automatically replicate the endorsed message. The endorsed message cannot be endorsed again.
8  * This class is a subclass of Post class.
9  * @author Aisha
10  * @author Saida
11  *
12  */
13 @SuppressWarnings("serial")
14 public class Endorsement extends Post {
15
16     //declaring variables
17     private String endorsedMessage;
18     private int endorsementID;
19     private static int lastEndorsementID = Post.getLastID();
20     private static ArrayList<Endorsement> endorsements = new ArrayList<>();
21
22     /**

```

```

23     * constructor for objects of class Endorsement.
24     * @param handle
25     * @param id
26     */
27     public Endorsement(String handle, int id) {
28         super(handle, id);
29         this.endorsementID = ++lastEndorsementID;
30         this.endorsedMessage = getMessage();
31     }
32
33     //setter for endorsedMessage
34     public void setEndorsedMessage(String endorsedMessage) {
35         this.endorsedMessage = endorsedMessage;
36     }
37
38     //getter for the static list of endorsements
39     public static ArrayList<Endorsement> getEndorsementsList() {
40         return endorsements;
41     }
42
43     //getter for endorsementID
44     public int getEndorsementID() {
45         return endorsementID;
46     }
47
48     //getter for lastEndorsementID()
49     public static int getLastEndorsementID() {
50         return lastEndorsementID;
51     }
52
53     /**
54      * This method adds an endorsement to the static list of endorsements.
55      */
56     public static void addEndorsement(Endorsement endorsement){
57         endorsements.add(endorsement);
58     }
59
60     /**
61      * This method displays the Endorsement.
62      */
63     public String toString() {
64         return "EP@ " + getHandle() + ": " + endorsedMessage;
65     }
66 }
67

```

## 4 Post.java

```

1 package socialmedia;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 /**

```

```

7  * This class stores information on each post of an individual Account in the social media platforms.
8  * Posts can be created, removed and have a caption up to 100 characters.
9  * The class is a subclass of Account class.
10 * @author Aisha
11 * @author Saida
12 *
13 */
14 @SuppressWarnings("serial")
15 public class Post extends Account {
16
17     //declaring variables
18     private static int lastID = 0;
19     private String message;
20     private int id;
21     private static ArrayList<Post> posts = new ArrayList<>();
22     private List<Endorsement> endorsementsList = new ArrayList<>();
23     private List<Comment> commentsList = new ArrayList<>();
24
25     /**
26      * constructor for objects of class Post.
27      * @param handle
28      * @param message
29      */
30     public Post(String handle, String message) {
31         super(handle);
32         this.id = ++lastID;
33         this.message = message;
34     }
35
36     /**
37      * 2nd constructor for objects of class Post.
38      * @param handle
39      * @param id
40      */
41     public Post(String handle, int id) {
42         super(handle);
43         this.id = id;
44     }
45
46     //setter for message
47     public void setMessage(String postCaption) {
48         this.message = postCaption;
49     }
50
51     //getter for message
52     public String getMessage() {
53         return message;
54     }
55
56     //getter for postID
57     public int getPostID() {
58         return id;
59     }
60
61     //getter for the list of endorsements of each post

```

```

62 public List<Endorsement> getEachPostEndorsementList() {
63     return endorsementsList;
64 }
65
66 //getter for the list of comments of each post
67 public List<Comment> getEachCommentList() {
68     return commentsList;
69 }
70
71 //getter for the static list of posts
72 public static ArrayList<Post> getPostsList() {
73     return posts;
74 }
75
76 /**
77  * This method adds a post into the static posts list.
78  */
79 public static void addPost(Post post){
80     posts.add(post);
81 }
82
83 /**
84  * This method gets the lastID of Post.
85  * @return lastID
86  */
87 public static int getLastID() {
88     return lastID;
89 }
90
91 /**
92  * This method adds an endorsement to the endorsement list of every post object created.
93  */
94 public void addEndorsementtoList(Endorsement endorsement) {
95     endorsementsList.add(endorsement);
96 }
97
98 /**
99  * This method returns the number of endorsements for each post object created.
100  * @return number of endorsements
101  */
102 public int getNumberOfEndorsements() {
103     return endorsementsList.size();
104 }
105
106 /**
107  * This method adds a comment to the comment list of every post object created.
108  */
109 public void addCommentToList(Comment comment) {
110     commentsList.add(comment);
111 }
112
113 /**
114  * This method returns the number of comments for each post object created.
115  * @return number of comments
116  */

```

```

117     public int getNumberOfComments() {
118         return commentsList.size();
119     }
120
121     /**
122      * This method displays the Post.
123      * @return toString of Post
124      */
125     public String toString() {
126         return "Post[name = " + getHandle() + ", message = " +
127             message + ", ID = " + id + "]";
128     }
129
130
131
132
133 }

```

## 5 SocialMedia.java

```

1  package socialmedia;
2
3  import java.util.ArrayList;
4  import java.io.FileInputStream;
5  import java.io.FileOutputStream;
6  import java.io.IOException;
7  import java.io.ObjectInputStream;
8  import java.io.ObjectOutputStream;
9
10 /**
11  * SocialMedia is a compiling implementor of
12  * the SocialMediaPlatform interface.
13  * @author Aisha
14  * @author Saida
15  * @version 1.0
16  */
17 @SuppressWarnings("serial")
18 public class SocialMedia implements SocialMediaPlatform {
19
20     public int createAccount(String handle, String description) throws IllegalHandleException,
21         InvalidHandleException {
22
23         if (handle.isEmpty() || handle.length() > 30 || handle.contains(" ")) {
24             throw new InvalidHandleException("Sorry, the handle you provided is empty, too long or contains
25                 whitespace.");
26         }
27
28         for (int i = 0; i < Account.getAccountsList().size(); i++) {
29             Account account = Account.getAccountsList().get(i);
30             if (account.getHandle().equals(handle)) {
31                 throw new IllegalHandleException("Sorry, the handle already exists.");
32             }
33         }
34     }
35 }

```



```

33     Account newAccount = new Account(handle, description);
34     Account.addAccount(newAccount);
35     return newAccount.getID();
36
37 }
38
39
40 @Override
41 public void removeAccount(String handle) throws HandleNotRecognisedException {
42
43     boolean checkHandle = false;
44
45     for (int i = 0; i < Account.getAccountsList().size(); i++) {
46         Account account = Account.getAccountsList().get(i);
47         if (account.getHandle().equals(handle)) {
48             Account.getAccountsList().remove(account);
49             Endorsement.getEndorsementsList().removeAll(account.getAccountEndorsementsList());
50             checkHandle = true;
51         }
52     }
53
54     for (int i = 0; i < Post.getPostsList().size(); i++) {
55         Post post = Post.getPostsList().get(i);
56         if (post.getHandle().equals(handle)) {
57             Post.getPostsList().remove(post);
58             post.getEachPostEndorsementList().clear();
59             checkHandle = true;
60         }
61     }
62
63     if (!checkHandle) {
64         throw new HandleNotRecognisedException("Sorry, the handle does not exist.");
65     }
66
67 }
68
69 @Override
70 public void updateAccountDescription(String handle, String description) throws
    HandleNotRecognisedException {
71
72     boolean checkHandle = false;
73
74     for (int i = 0; i < Account.getAccountsList().size(); i++) {
75         Account account = Account.getAccountsList().get(i);
76         if (account.getHandle().equals(handle)) {
77             account.setDescription(description);
78             checkHandle = true;
79         }
80
81     } if (!checkHandle) {
82         throw new HandleNotRecognisedException("Sorry, the handle does not exist.");
83     }
84
85 }
86

```

```

87
88 @Override
89 public int getNumberOfAccounts() {
90     return Account.getAccountsList().size();
91 }
92
93
94 @Override
95 public int getTotalOriginalPosts() {
96     return Post.getPostsList().size();
97 }
98
99
100 @Override
101 public int getTotalEndorsmentPosts() {
102     return Endorsement.getEndorsementsList().size();
103 }
104
105 @Override
106 public int getTotalCommentPosts() {
107     return Comment.getCommentsList().size();
108 }
109
110 @Override
111 public int createAccount(String handle) throws IllegalHandleException, InvalidHandleException {
112
113     if (handle.isEmpty() || handle.length() > 30 || handle.contains(" ")) {
114         throw new InvalidHandleException("Sorry, the handle you provided is empty, too long or contains
115             whitespace.");
116     }
117
118     for (int i = 0; i < Account.getAccountsList().size(); i++) {
119         Account account = Account.getAccountsList().get(i);
120         if (account.getHandle().equals(handle)) {
121             throw new IllegalHandleException("Sorry, the handle already exists.");
122         }
123     }
124
125     Account newAccount = new Account(handle);
126     Account.addAccount(newAccount);
127     return newAccount.getID();
128 }
129
130 @Override
131 public void removeAccount(int id) throws AccountIDNotRecognisedException {
132
133     boolean checkID = false;
134
135     for (int i = 0; i < Account.getAccountsList().size(); i++) {
136         Account account = Account.getAccountsList().get(i);
137         if (account.getID() == id) {
138             checkID = true;
139             Account.getAccountsList().remove(account);
140             Post.getPostsList().removeAll(account.getEachPostList());

```

```

141         Endorsement.getEndorsementsList().removeAll(account.getAccountEndorsementsList());
142     }
143 }
144
145 if (!checkID) {
146     throw new AccountIDNotRecognisedException("Sorry, the ID is not recognised.");
147 }
148
149 }
150
151 @Override
152 public void changeAccountHandle(String oldHandle, String newHandle) throws HandleNotRecognisedException,
    IllegalHandleException, InvalidHandleException {
153
154     boolean checkHandle = false;
155
156     for (int i = 0; i < Account.getAccountsList().size(); i++) {
157         Account account = Account.getAccountsList().get(i);
158         if (account.getHandle().equals(newHandle)) {
159             throw new IllegalHandleException("Sorry, this handle already exists.");
160         } else if (oldHandle.isEmpty() || oldHandle.length() > 30 || oldHandle.contains(" ")) {
161             throw new InvalidHandleException("Sorry, the handle you provided is empty, too long or
                contains whitespace.");
162         } else if (account.getHandle().equals(oldHandle)) {
163             account.setHandle(newHandle);
164             checkHandle = true;
165         }
166     }
167
168     if (!checkHandle) {
169         throw new HandleNotRecognisedException("Sorry, this handle does not exist.");
170     } else if (newHandle.isEmpty() || newHandle.length() > 30 || newHandle.contains(" ")) {
171         throw new InvalidHandleException("Sorry, the handle you provided is empty, too long or contains
            whitespace.");
172     }
173
174 }
175
176 @Override
177 public String showAccount(String handle) throws HandleNotRecognisedException {
178
179     boolean checkHandle = false;
180
181     for (int i = 0; i < Account.getAccountsList().size(); i++) {
182         Account account = Account.getAccountsList().get(i);
183         if (account.getHandle().equals(handle)) {
184             String formattedString = String.format("ID: " + account.getID() + "\nAccount: " +
                account.getHandle() + "\nDescription: "
185             + account.getDescription() + "\nPost Count: " + account.getEachPostList().size() +
                "\nEndorsement Count: " + account.getAccountEndorsementsList().size());
186             checkHandle = true;
187             return formattedString;
188         }
189     }
190
191     if (!checkHandle) {
192         throw new HandleNotRecognisedException("Sorry, the handle does not exist.");
193     }
194 }

```

```

191     }
192
193     return null;
194
195 }
196
197 @Override
198 public int createPost(String handle, String message) throws HandleNotRecognisedException,
199     InvalidPostException {
200
201     boolean checkHandle = false;
202
203     if (message.isEmpty() || message.length() > 100) {
204         throw new InvalidPostException("Sorry, your post is either empty or too long.");
205     }
206
207     for (int i = 0; i < Account.getAccountsList().size(); i++) {
208         Account account = Account.getAccountsList().get(i);
209         if (account.getHandle().equals(handle)) {
210             Post newPost = new Post(handle, message);
211             Post.addPost(newPost);
212             account.getEachPostList().add(newPost);
213             checkHandle = true;
214             return newPost.getPostID();
215         }
216     }
217
218     if (!checkHandle) {
219         throw new HandleNotRecognisedException("Sorry, the handle does not exist.");
220     }
221
222     return 0;
223 }
224
225 @Override
226 public int endorsePost(String handle, int id) throws HandleNotRecognisedException,
227     PostIDNotRecognisedException, NotActionablePostException {
228
229     boolean checkHandle = false;
230     boolean checkID = false;
231
232     for (int i = 0; i < Endorsement.getEndorsementsList().size(); i++) {
233         Endorsement endorsement = Endorsement.getEndorsementsList().get(i);
234         if (endorsement.getEndorsementID() == id) {
235             throw new NotActionablePostException("You cannot endorse an endorsement.");
236         }
237     }
238
239     for (int i = 0; i < Post.getPostsList().size(); i++) {
240         Post post = Post.getPostsList().get(i);
241         if (post.getPostID() == id) {
242             checkID = true;
243         }
244         for (Comment comment: post.getEachCommentList()) {

```

```

244         if (comment.getCommentID() == id) {
245             checkID = true;
246         }
247     }
248 }
249
250 if(!checkID) {
251     throw new PostIDNotRecognisedException("Sorry, the ID is not recognised.");
252 }
253
254 for (int i = 0; i < Account.getAccountsList().size(); i++) {
255     Account account = Account.getAccountsList().get(i);
256     if (account.getHandle().equals(handle)) {
257         checkHandle = true;
258     }
259 }
260
261 if(!checkHandle) {
262     throw new HandleNotRecognisedException("Sorry, the handle does not exist.");
263 }
264
265 for (int i = 0; i < Post.getPostsList().size(); i++) {
266     for (Account account: Account.getAccountsList()) {
267         Post post = Post.getPostsList().get(i);
268         if (post.getPostID() == id && account.getHandle().equals(post.getHandle())) {
269             Endorsement newEndorsement = new Endorsement(handle, id);
270             Endorsement.addEndorsement(newEndorsement);
271             post.addEndorsementtoList(newEndorsement);
272             account.addAccountEndorsement(newEndorsement);
273             return newEndorsement.getEndorsementID();
274         }
275     }
276 }
277
278 return 0;
279 }
280
281
282 @Override
283 public int commentPost(String handle, int id, String message) throws HandleNotRecognisedException,
284     PostIDNotRecognisedException, NotActionablePostException, InvalidPostException {
285
286     boolean checkHandle = false;
287     boolean checkID = false;
288
289     for (int i = 0; i < Endorsement.getEndorsementsList().size(); i++) {
290         Endorsement endorsement = Endorsement.getEndorsementsList().get(i);
291         if (endorsement.getEndorsementID() == id) {
292             throw new NotActionablePostException("You cannot endorse an endorsement.");
293         }
294     }
295
296     @SuppressWarnings("unused")
297     boolean checkComment = false;
298     for (int j = 0; j < Post.getPostsList().size(); j++) {

```

```

299         Post post = Post.getPostsList().get(j);
300         if (post.getPostID() == id) {
301             checkID = true;
302         } else {
303             for (Comment comment: post.getEachCommentList()) {
304                 if (comment.getCommentID() == id) {
305                     checkID = true;
306                     checkComment = true;
307                 }
308             }
309         }
310     }
311
312     if(!checkID){
313         throw new PostIDNotRecognisedException("Sorry, the ID is not recognised.");
314     }
315
316     for (int i = 0; i < Post.getPostsList().size();i++) {
317         Post post = Post.getPostsList().get(i);
318         if (post.getHandle().equals(handle)) {
319             checkHandle = true;
320         }
321     }
322
323     if(!checkHandle){
324         throw new HandleNotRecognisedException("Sorry, the handle does not exist.");
325     } else if (message.isEmpty() || message.length() > 100) {
326         throw new InvalidPostException("Sorry, your post is either empty or too long.");
327     }
328
329     for (int i = 0; i < Post.getPostsList().size();i++) {
330         Post post = Post.getPostsList().get(i);
331         if (post.getPostID() == id) {
332             Comment comment = new Comment(handle, id, message);
333             Comment.addComment(comment);
334             post.addCommentToList(comment);
335             comment.getCommentsCommentsList().add(comment);
336             return comment.getCommentID();
337         }
338     }
339
340     return 0;
341 }
342
343 @Override
344 public void deletePost(int id) throws PostIDNotRecognisedException {
345
346     boolean checkPostID = false;
347
348     for (int i = 0; i < Post.getPostsList().size(); i++) {
349         Post post = Post.getPostsList().get(i);
350         if (post.getPostID() == id) {
351             checkPostID = true;
352             Post.getPostsList().remove(post);
353             Endorsement.getEndorsementsList().removeAll(post.getEachPostEndorsementList());

```

```

354     }
355 }
356
357 if(!checkPostID){
358     throw new PostIDNotRecognisedException("Sorry, the ID is not recognised.");
359 }
360
361 }
362
363 @Override
364 public String showIndividualPost(int id) throws PostIDNotRecognisedException {
365
366     boolean checkPostID = false;
367
368     for (int i = 0; i < Post.getPostsList().size();i++) {
369         Post post = Post.getPostsList().get(i);
370         if (post.getPostID() == id) {
371             String formattedString = String.format("ID: " + post.getPostID() + "\nAccount: " +
372                 post.getHandle()
373                 + "\nNo. of Endorsements: " + post.getNumberOfEndorsements() +
374                 " || No. of Comments: " + post.getNumberOfComments() + "\n" + post.getMessage());
375             checkPostID = true;
376             return formattedString;
377         }
378
379         if (!checkPostID){
380             throw new PostIDNotRecognisedException("Sorry, the ID is not recognised.");
381         }
382
383         return null;
384     }
385 }
386
387 @Override
388 public StringBuilder showPostChildrenDetails(int id)
389     throws PostIDNotRecognisedException, NotActionablePostException {
390
391     boolean checkID = false;
392
393     for (int i = 0; i < Endorsement.getEndorsementsList().size(); i++) {
394         Endorsement endorsement = Endorsement.getEndorsementsList().get(i);
395         if (endorsement.getEndorsementID() == id) {
396             throw new NotActionablePostException("You cannot endorse an endorsement.");
397         }
398     }
399
400     StringBuilder postChildren = new StringBuilder();
401
402     for (int i = 0; i < Post.getPostsList().size(); i++) {
403         Post post = Post.getPostsList().get(i);
404         if (post.getPostID() == id) {
405             checkID = true;
406             postChildren.append(showIndividualPost(id));
407             for (int j = 0; j < post.getEachCommentList().size(); j++) {

```

```

408         Comment comment = post.getEachCommentList().get(j);
409         postChildren.append("\n| > ID: ").append(comment.getCommentID()).append("\n");
410         postChildren.append("| Account: ").append(comment.getHandle()).append("\n");
411         postChildren.append("| No.endorsements: ").append(post.getNumberOfEndorsements()).append(" |
         No. comments: ").append(post.getNumberOfComments());
412         postChildren.append("\n").append(comment.getComment());
413         for (int k = 0; k < comment.getCommentsCommentsList().size();k++) {
414             Comment commentOfcomment = comment.getCommentsCommentsList().get(k);
415             postChildren.append("\n | > ID: ").append(commentOfcomment.getCommentID()).append("\n");
416             postChildren.append(" | Account: ").append(commentOfcomment.getHandle()).append("\n");
417             postChildren.append(" | No.endorsements:
                 ").append(commentOfcomment.getNumberOfEndorsements()).append(" | No. comments:
                 ").append(commentOfcomment.getNumberOfComments());
418             postChildren.append("\n").append(commentOfcomment.getComment());
419         }
420     }
421 }
422 }
423
424     if(!checkID){
425         throw new PostIDNotRecognisedException("Sorry, the ID is not recognised.");
426     }
427
428     return postChildren;
429 }
430
431 @Override
432 public int getMostEndorsedPost() {
433
434     int max = 0;
435     Post mostEndorsedPost = null;
436
437     for (int i = 0; i < Post.getPostsList().size(); i++) {
438         Post post = Post.getPostsList().get(i);
439         int endorsementsSize = post.getEachPostEndorsementList().size();
440         if (endorsementsSize > max) {
441             max = endorsementsSize;
442             mostEndorsedPost = post;
443         }
444     }
445
446     System.out.println("The most endorsed post is: " + mostEndorsedPost);
447
448     return mostEndorsedPost.getPostID();
449 }
450
451 @Override
452 public int getMostEndorsedAccount() {
453
454     int max = 0;
455     Account mostEndorsedAccount = null;
456
457     for (int i = 0; i < Account.getAccountsList().size(); i++) {
458         Account account = Account.getAccountsList().get(i);

```



```

460         int endorsementsSize = account.getAccountEndorsementsList().size();
461         if (endorsementsSize > max) {
462             max = endorsementsSize;
463             mostEndorsedAccount = account;
464         }
465     }
466 }
467
468 System.out.println("The most endorsed account is: " + mostEndorsedAccount);
469
470
471 return mostEndorsedAccount.getID();
472 }
473
474 @Override
475 public void erasePlatform() {
476     Account.getAccountsList().clear();
477     Post.getPostsList().clear();
478     Endorsement.getEndorsementsList().clear();
479     Comment.getCommentsList().clear();
480
481 }
482
483
484 @Override
485 public void savePlatform(String filename) throws IOException {
486
487     try {
488
489         FileOutputStream fos = new FileOutputStream(filename);
490         ObjectOutputStream OOS = new ObjectOutputStream(fos);
491
492         ArrayList <Account> accounts = Account.getAccountsList();
493         ArrayList<Post> posts = Post.getPostsList();
494         ArrayList<Endorsement> endorsements = Endorsement.getEndorsementsList();
495         ArrayList<Comment> comments = Comment.getCommentsList();
496
497         OOS.writeObject(accounts);
498
499         OOS.writeObject(posts);
500
501         OOS.writeObject(endorsements);
502
503         OOS.writeObject(comments);
504
505         OOS.close();
506         fos.close();
507
508         System.out.println("Platform saved.");
509
510     } catch (IOException e) {
511         e.printStackTrace();
512     }
513 }
514

```

```

515 @Override
516 @SuppressWarnings("unchecked")
517 public void loadPlatform(String filename) throws IOException, ClassNotFoundException {
518     try {
519         FileInputStream fis = new FileInputStream(filename);
520         ObjectInputStream OIS = new ObjectInputStream(fis);
521         ArrayList<Account> accounts = (ArrayList<Account>) OIS.readObject();
522         ArrayList<Post> posts = (ArrayList<Post>) OIS.readObject();
523         ArrayList<Endorsement> endorsements = (ArrayList<Endorsement>) OIS.readObject();
524         ArrayList<Comment> comments = (ArrayList<Comment>) OIS.readObject();
525
526         for (Post post : posts) {
527             System.out.println(post);
528         }
529         for (Account account : accounts) {
530             System.out.println(account);
531         }
532         for (Comment comment : comments) {
533             System.out.println(comment);
534         }
535         for (Endorsement endorsement : endorsements) {
536             System.out.println(endorsement);
537         }
538         OIS.close();
539         fis.close();
540     } catch (IOException | ClassNotFoundException e) {
541         e.printStackTrace();
542     }
543 }
544
545 }
546
547 }
548
549 }
550
551 }
552
553 }
554
555 }
556
557 }
558
559 }
560
561 }
562 }

```