

```

1 package socialmedia;
2
3 import javax.security.auth.login.AccountNotFoundException;
4 import java.io.*;
5 import java.util.ArrayList;
6 import java.util.List;
7 import java.util.logging.Level;
8 import java.util.logging.Logger;
9
10 /**
11  * SocialMedia is an implementor of the SocialMediaPlatform interface.
12  */
13 public class SocialMedia implements SocialMediaPlatform {
14
15     private static final Logger LOGGER = Logger.getLogger( SocialMedia
16     .class.getName());
17     private List<Account> accounts = new ArrayList<>();
18     private ArrayList<Post> posts = new ArrayList<>();
19     private StringBuilder childrenPostContent = new StringBuilder();
20     private Account DELETED_USER = new Account();
21
22     @Override
23     public int createAccount(String handle) throws
24     IllegalHandleException, InvalidHandleException {
25         Account account = new Account(handle);
26         accounts.add(account);
27         return account.getId();
28     }
29
30     @Override
31     public int createAccount(String handle, String description) throws
32     IllegalHandleException, InvalidHandleException {
33         try {
34             Account account = new Account(handle, description);
35             accounts.add(account);
36             return account.getId();
37         } catch (IllegalHandleException ihe){
38             LOGGER.log( Level.SEVERE, ihe.toString(), ihe );
39             return -1;
40         } catch (InvalidHandleException ine){
41             LOGGER.log( Level.SEVERE, ine.toString(), ine );
42             return -1;
43         }
44     }
45
46     @Override
47     public void removeAccount(int id) throws
48     AccountIDNotRecognisedException {
49         Account a = getAccountById(id);
50         a.removeAccount();
51         accounts.remove(a);
52     }
53
54     @Override
55     public void removeAccount(String handle) throws

```

```

53 HandleNotRecognisedException {
54     Account a = getAccountByHandle(handle);
55     a.removeAccount();
56     // TODO: remove corresponding posts and likes too
57     accounts.remove(a);
58 }
59
60 @Override
61 public void changeAccountHandle(String oldHandle, String
newHandle)
62     throws HandleNotRecognisedException,
IllegalHandleException, InvalidHandleException {
63     Account a = getAccountByHandle(oldHandle);
64     a.changeAccountHandle(newHandle);
65 }
66
67 /**
68  * Finds account with a given handle.
69  * @param handle handle to identify the account.
70  * @return account with a given handle.
71  * @throws HandleNotRecognisedException if there is no matching
account.
72  */
73 public Account getAccountByHandle(String handle) throws
HandleNotRecognisedException {
74     boolean found = false;
75     int foundIndex = -1;
76     for (int i = 0; i < accounts.size() && !found; i++) {
77         if (accounts.get(i).getHandle() == handle){
78             found = true;
79             foundIndex = i;
80         }
81     }
82     if (found){
83         return accounts.get(foundIndex);
84     }
85     else {
86         throw new HandleNotRecognisedException("Account with
handle: " + handle + " not found in the system.");
87     }
88 }
89
90 /**
91  * Finds account with a given id.
92  * @param id id to identify the account.
93  * @return account with a given id.
94  * @throws AccountIDNotRecognisedException
95  */
96 public Account getAccountById(int id) throws
AccountIDNotRecognisedException {
97     boolean found = false;
98     int foundIndex = -1;
99     for (int i = 0; i < accounts.size() && !found; i++) {
100         if (accounts.get(i).getId() == id){
101             found = true;
102             foundIndex = i;
103         }

```

```

104     }
105     if (found){
106         return accounts.get(foundIndex);
107     }
108     else {
109         throw new AccountIDNotRecognisedException("Account with
110 id: " + id + " not found in the system.");
111     }
112
113     @Override
114     public void updateAccountDescription(String handle, String
115 description) throws HandleNotRecognisedException {
116         Account a = getAccountByHandle(handle);
117         a.setDescriptionField(description);
118     }
119
120     @Override
121     public String showAccount(String handle) throws
122 HandleNotRecognisedException {
123         Account a = getAccountByHandle(handle);
124         return a.toString();
125     }
126
127     @Override
128     public int createPost(String handle, String message) throws
129 HandleNotRecognisedException, InvalidPostException {
130         Account a = getAccountByHandle(handle);
131         Post p = new Post(message, a);
132         posts.add(p);
133         return p.getId();
134     }
135
136     @Override
137     public int endorsePost(String handle, int id)
138 throws HandleNotRecognisedException,
139 PostIDNotRecognisedException, NotActionablePostException {
140         Post p = getPostById(id);
141         Account a = getAccountByHandle(handle);
142         if (p instanceof Endorsement){
143             throw new NotActionablePostException("Post with id: '" +
144 id + "' is an endorsement " +
145 "so it cannot be endorsed.");
146         }
147         Endorsement endorsement = new Endorsement(p,a);
148         posts.add(endorsement);
149         return endorsement.getId();
150     }
151
152     public Post getPostById(int id) throws
153 PostIDNotRecognisedException{
154         for (int i = 0; i < posts.size(); i++) {
155             if (posts.get(i).getId() == id){
156                 return posts.get(i);
157             }
158         }
159     }

```

```

154         throw new PostIDNotRecognisedException("Post with id: '" + id
155         + "' not found.");
156     }
157     @Override
158     public int commentPost(String handle, int id, String message)
159     throws HandleNotRecognisedException,
160         PostIDNotRecognisedException, NotActionablePostException
161     , InvalidPostException {
162         Account a = getAccountByHandle(handle);
163         Post p = getPostById(id);
164         if (p instanceof Endorsement){
165             throw new NotActionablePostException("Post with id: '" +
166             id + "' is an endorsement " +
167             "so it cannot be commented.");
168         }
169         Comment comment = new Comment(a,p,message);
170         posts.add(comment);
171         return comment.getId();
172     }
173     @Override
174     public void deletePost(int id) throws
175     PostIDNotRecognisedException {
176         Post targetPost = new Post();
177         for(Post post: posts){
178             if(post.getId() == id){
179                 targetPost = post;
180                 break;
181             }
182         }
183         targetPost.setPostContent("<The initial content was deleted
184         from the system, therefore it is not available anymore.>");
185         targetPost.setDeleted(true);
186         for(Post Endorsement: targetPost.getEndorsements()){
187             for(Post post: posts){
188                 if(post.getId() == Endorsement.getId()){
189                     posts.remove(post);
190                     break;
191                 }
192             }
193             Endorsement.clearAll();
194         }
195     }
196     @Override
197     public String showIndividualPost(int id) throws
198     PostIDNotRecognisedException {
199         boolean postIDRecognised = false;
200         Post targetPost = new Post();
201         String individualPost;
202         try {
203             for (Post post : posts) {
204                 if (post.getId() == id) {
205                     postIDRecognised = true;
206                     targetPost = post;
207                     break;

```

```

204         }
205     }
206     if (!postIDRecognised) {
207         throw new PostIDNotRecognisedException("Post Id not
recognised exception: Please try again entering a valid Id.");
208     }
209     individualPost = "Id: " + targetPost.getId() + "\nAccount
: " + targetPost.getAccount().getHandle()
210         + "\nNo. endorsements: " + targetPost.
getEndorsementCount() + " | No. comments: " + targetPost.
getCommentCount() + "\n" + targetPost.getPostContent() + " \n";
211     return individualPost;
212 } catch (PostIDNotRecognisedException e){
213     String message = "Post Id not recognised exception:
Please try again entering a valid Id.";
214     return message;
215 }
216 }
217 public void clearStringBuilder(){
218     childrenPostContent.setLength(0);
219 }
220 public void FormatStringBuilder(Post post) throws
PostIDNotRecognisedException {
221     try{
222         if(post != null){
223             String individualPost;
224             if(post.isComment()){
225                 childrenPostContent.append((" ").repeat(Math.
max(0,post.getDepth()) - 1)).append("| >");
226                 individualPost = showIndividualPost(post.getId
()).replace("\n","\n" + (" ").repeat(Math.max(0,post.getDepth())));
227             }else{
228                 individualPost = showIndividualPost(post.getId
());
229             }
230             childrenPostContent.append(individualPost).append("\n");
231             for(Post child: post.getPostChildrenList()){
232                 FormatStringBuilder(child);
233             }
234         }
235     } catch (PostIDNotRecognisedException e){
236         System.out.println("PROBLEM WITH showingIndividualPosts
()");
237     }
238 }
239
240
241 @Override
242 public StringBuilder showPostChildrenDetails(int id)
243     throws PostIDNotRecognisedException,
NotActionablePostException {
244     // TODO Auto-generated method stub
245     return null;
246 }
247
248 @Override

```

```

249     public int getNumberOfAccounts() {
250         return accounts.size();
251     }
252
253     @Override
254     public int getTotalOriginalPosts() {
255         int count = 0;
256         for(Post post: posts){
257             if(post.isComment() || post.isEndorsement()){
258                 continue;
259             }
260             count++;
261         }
262         return count;
263     }
264
265     @Override
266     public int getTotalEndorsmentPosts() {
267         int count = 0;
268         for(Post post: posts){
269             if(!post.isEndorsement()){
270                 continue;
271             }
272             count++;
273         }
274         return count;
275     }
276
277     @Override
278     public int getTotalCommentPosts() {
279         int count = 0;
280         for(Post post: posts){
281             if(!post.isComment()){
282                 continue;
283             }
284             count++;
285         }
286         return count;
287     }
288
289     @Override
290     public int getMostEndorsedPost() {
291         Post MostEndorsed = new Post();
292         for(Post post : posts){
293             if(post.isEndorsement()){
294                 continue;
295             }
296             if(MostEndorsed.getAccount() == null){
297                 MostEndorsed = post;
298             }
299             else if(MostEndorsed.getEndorsementCount() < post.
300 getEndorsementCount()){
301                 MostEndorsed = post;
302             }
303         }
304         return MostEndorsed.getId();

```

```

305
306     }
307
308     @Override
309     public int getMostEndorsedAccount() {
310         Account MostEndorsed = new Account();
311         for(Account user : accounts){
312             if(MostEndorsed.getHandle() == null){
313                 MostEndorsed = user;
314             }
315             else if(MostEndorsed.getUserEndorsements() < user.
316 getUserEndorsements()){
317                 MostEndorsed = user;
318             }
319         }
320         return MostEndorsed.getId();
321     }
322
323     @Override
324     public void erasePlatform() {
325         for(Account account: accounts){
326             account.getPosts().clear();
327         }
328         accounts.clear();
329         for(Post posts: posts){
330             posts.getPostChildrenList().clear();
331             posts.getEndorsements().clear();
332             posts.clearAccount();
333         }
334         posts.clear();
335         DELETED_USER.getPosts().clear();
336     }
337
338
339     @Override
340     public void savePlatform(String filename) throws IOException {
341         try{
342             FileOutputStream fileStore = new FileOutputStream(
343 filename + ".ser");
344             ObjectOutputStream objectStore = new ObjectOutputStream(
345 fileStore);
346             objectStore.writeObject(accounts);
347             objectStore.writeObject(posts);
348             objectStore.writeObject(DELETED_USER);
349             objectStore.close();
350             fileStore.close(); }
351         catch(IOException e){
352             System.out.println("Unfortunately, not able to find the
353 file. Please try again.");
354         }
355     }
356
357     @Override
358     public void loadPlatform(String filename) throws IOException,
359 ClassNotFoundException {
360         try{

```

```
357         ObjectInputStream ois = new ObjectInputStream(new
        FileInputStream(filename + ".ser"));
358         accounts = (ArrayList<Account>) ois.readObject();
359         posts = (ArrayList<Post>) ois.readObject();
360         DELETED_USER = (Account) ois.readObject();
361         ois.close();}
362     catch(IOException | ClassNotFoundException e) {
363         System.out.println("Unfortunately, the class or the file
        were not found. Please try again.");
364     }
365
366 }
367
368 public List<Account> getAllAccounts(){
369     return accounts;
370 }
371
372 public List<Post> getPosts() {return posts; }
373 }
374
```



```

1 package socialmedia;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 /**
7  * A class representing user's post.
8  */
9 public class Post {
10     private Account account;
11     private int numberOfEndorsements = 0;
12     private int numberOfComments = 0;
13     private String postContent;
14     private int commentCount = 0;
15     private int endorsementCount = 0;
16     private boolean isDeleted = false;
17     private boolean isComment = false;
18     private ArrayList<Post> postChildrenList = new ArrayList<>();
19     private boolean isEndorsement = false;
20     private int depth = 0;
21     protected int id;
22     // will increment with each new post. Will make sure that they all
    have an unique numerical id
23     private static int counter = 0;
24     protected String message;
25     Account author;
26     List<Endorsement> endorsements = new ArrayList<>();
27     List<Comment> comments = new ArrayList<>();
28     Post(){
29     }
30     Post(Account account, int id){
31         this.account = account;
32         this.id = id;
33     }
34     Post(Account account, String postContent, int postID) {
35         this.account = account;
36         this.postContent = postContent;
37         this.id = id;
38     }
39
40     public Post(String message, Account author) throws
InvalidPostException {
41         setId();
42         setMessage(message);
43         this.author = author;
44     }
45
46     public void clearAccount(){
47         this.account = null;
48     }
49
50     /**
51     * Constructor used by child classes (does not throw exception)
52     * It also does not set the message.
53     *
54     * @param author author Account
55     */

```

```

56     public Post(Account author) {
57         setId();
58         this.author = author;
59     }
60
61     protected void setId() {
62         this.id = counter;
63         counter += 1;
64     }
65
66     /**
67      * Sets post's message
68      *
69      * @param message message to be set
70      * @throws InvalidPostException if message is null, just white
characters, empty or longer than 100 characters
71      */
72     public void setMessage(String message) throws
InvalidPostException {
73         if (message != null && message.trim() != ""
74             && message.length() <= 100) {
75             this.message = message;
76         } else
77             throw new InvalidPostException("Message '" + message +
"' is either null, just whitespaces " +
78             "or longer than 100 chars. Post cannot be created
.");
79     }
80
81     public int getId() {
82         return id;
83     }
84
85     public String getMessage() {
86         return message;
87     }
88
89     public Account getAuthor() {
90         return author;
91     }
92
93     public List<Endorsement> getEndorsements() {
94         return endorsements;
95     }
96
97     public List<Comment> getComments() {
98         return comments;
99     }
100
101     /**
102      * increments number of endorsements for the post and adds the
endorsement.
103      */
104     public void endorse(Endorsement endorsement) {
105         endorsements.add(endorsement);
106         this.numberOfEndorsements += 1;
107     }

```

```

108
109     /**
110      * increments comments count and adds the comment.
111      */
112     public void comment(Comment comment) {
113         comments.add(comment);
114         this.numberOfComments += 1;
115     }
116
117     public void clearAll() {
118         account = null;
119         postContent = "<The original content was deleted from the
system and is not available anymore.>";
120         commentCount = -1;
121         endorsementCount = -1;
122     }
123
124     @Override
125     public String toString() {
126         return "Post{" +
127             "id=" + id +
128             ", message='" + message + '\'' +
129             ", author=" + author +
130             ", endorsements=" + endorsements +
131             ", comments=" + comments +
132             '}';
133     }
134
135
136     public void setPostContent(String postContent) {
137         this.postContent = postContent;
138     }
139     public boolean isDeleted() {
140         return isDeleted;
141     }
142     public void setDeleted(boolean deleted) {
143         isDeleted = deleted;
144     }
145     public Account getAccount() {
146         return account;
147     }
148     public int getEndorsementCount() {
149         return endorsementCount;
150     }
151     public int getCommentCount() {
152         return commentCount;
153     }
154     public String getPostContent() {
155         return postContent;
156     }
157     public void setComment(boolean comment) {
158         isComment = comment;
159     }
160     public boolean isComment() {
161         return isComment;
162     }
163     public void addDepth(){

```

```
164         depth++;
165     }
166     public int getDepth() {
167         return depth;
168     }
169     public void setDepth(int depth){
170         this.depth = depth;
171     }
172     public ArrayList<Post> getPostChildrenList() {
173         return postChildrenList;
174     }
175     public void setEndorsement(boolean endorsement) {
176         isEndorsement = endorsement;
177     }
178
179     public boolean isEndorsement() {
180         return isEndorsement;
181     }
182 }
```

```
1 package socialmedia;
2
3 public class Comment extends Post{
4     private Post commentedPost;
5
6     public Comment(Account author, Post commentedPost, String
commentText) throws InvalidPostException {
7         super(author);
8         super.setMessage(message);
9         this.commentedPost = commentedPost;
10        commentedPost.comment(this);
11        commentedPost.author.comment();
12    }
13
14    @Override
15    public String toString() {
16        return "Comment{" +
17            "commentedPost=" + commentedPost +
18            ", id=" + id +
19            ", message='" + message + '\'' +
20            ", author=" + author +
21            '}';
22    }
23 }
24
```

```
1 package socialmedia;
2
3 /**
4  * A class representing other post's endorsement.
5  */
6 public class Endorsement extends Post{
7     private Post endorsedPost;
8
9     public Endorsement(Post endorsedPost, Account account){
10         super(account);
11         this.message = endorsedPost.message;
12         this.endorsedPost = endorsedPost;
13         endorsedPost.endorse(this);
14         endorsedPost.author.endorse();
15     }
16
17     @Override
18     public String toString() {
19         return "Endorsement{" +
20             "id=" + id +
21             ", message='" + message + '\'' +
22             ", author=" + author +
23             '}';
24     }
25 }
26
```

```

1 package socialmedia;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 /**
7  * A class representing user's account
8  */
9 public class Account {
10     private int id;
11     private int numberOfEndorsements = 0;
12     private int numberOfComments = 0;
13     // will increment with each new account. Will make sure that they
    all have an unique numerical id
14     private static int counter = 0;
15     private String descriptionField;
16     private String handle;
17     private ArrayList<Post> posts = new ArrayList<>();
18     private static List<String> accountHandles = new ArrayList<>();
19     Account(){
20
21     }
22
23     /**
24      * Creates an instance of Account class
25      * @param descriptionField String containing description field
26      * @param handle handle to identify the account
27      * @throws IllegalArgumentException if the handle already exists in
    the platform.
28      * @throws InvalidHandleException if the new handle is empty, has
    more than 30 characters, or has white spaces.
29      */
30     public Account( String handle, String descriptionField) throws
    IllegalArgumentException, InvalidHandleException {
31         setId();
32         setHandle(handle);
33         setDescriptionField(descriptionField);
34     }
35
36     /**
37      * Creates an instance of Account class with empty description
    field
38      * @param handle handle to identify the account
39      * @throws IllegalArgumentException if the handle already exists in
    the platform.
40      * @throws InvalidHandleException if the new handle is empty, has
    more than 30 characters, or has white spaces.
41      */
42     public Account(String handle) throws IllegalArgumentException,
    InvalidHandleException {
43         setId();
44         setHandle(handle);
45         setDescriptionField("");
46     }
47
48
49     public int getId() {

```

```

50         return id;
51     }
52
53     public void setId() {
54         this.id = counter;
55         counter += 1;
56     }
57
58     public String getDescriptionField() {
59         return descriptionField;
60     }
61
62     public void setDescriptionField(String descriptionField) {
63         this.descriptionField = descriptionField;
64     }
65
66     public String getHandle() {
67         return handle;
68     }
69
70     /**
71      * Sets a handle for account instance.
72      * @param handle handle to identify the account.
73      * @throws IllegalHandleException if the handle already exists in
74      the platform.
75      * @throws InvalidHandleException if the new handle is empty, has
76      more than 30 characters, or has white spaces.
77      */
78     private void setHandle(String handle) throws
79     IllegalHandleException, InvalidHandleException{
80         if (handle != null && handle != ""
81             && handle.length() <= 30 &&
82             !containsWhitespace(handle)) {
83             if (!accountHandles.contains(handle)){
84                 this.handle = handle;
85                 accountHandles.add(handle);
86             }else
87                 throw new IllegalHandleException("Handle '" + handle
88 + "' already exists in the system.");
89         }else
90             throw new InvalidHandleException("Handle '"+ handle +"
91 is empty, has more than 30 characters," +
92             " or has white spaces");
93     }
94
95     public void changeAccountHandle(String newHandle)throws
96     IllegalHandleException, InvalidHandleException{
97         accountHandles.remove(this.handle);
98         setHandle(newHandle);
99     }
100
101     /**
102      * Checks a string for whitespaces
103      * @param str string checked for whitespaces
104      * @return true if given string contains any whitespace, else
105      false
106     */

```



```

100     private boolean containsWhitespace(String str){
101         // true if contains onw or more instances of any whitespace
        character
102         return str.matches(".*\\s.*");
103     }
104
105     public void removeAccount(){
106         accountHandles.remove(this.handle);
107     }
108
109     /**
110      * increments number of endorsements for the account.
111      */
112     public void endorse(){
113         this.numberOfEndorsements += 1;
114     }
115     public void comment(){
116         this.numberOfComments +=1;
117     }
118
119     @Override
120     public String toString() {
121         return "Account{" +
122             "id=" + id +
123             ", descriptionField='" + descriptionField + '\'' +
124             ", handle='" + handle + '\'' +
125             '}';
126     }
127     public int getUserEndorsements() {
128         return numberOfEndorsements;
129     }
130
131     public ArrayList<Post> getPosts() {
132         return posts;
133     }
134 }
135

```