```java
package socialmedia;

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

import socialmedia.exceptions.*;
import socialmedia.models.Account;
import socialmedia.models.Comment;
import socialmedia.models.EndorsementPost;
import socialmedia.models.Post;

import java.util.stream.Collectors;

public class SocialMedia implements SocialMediaPlatform, Serializable {
    private static final int MAX_COMMENT_LENGTH = 100;

    private ArrayList<Account> accounts = new ArrayList<>();

    private ArrayList<Post> posts = new ArrayList<>();

    /**
     * Indicates what the ID of the next new post should be.
     */
    private int postId = 0;

    /**
     * Create an account.
     *
     * @param handle the accounts handle as input
     * @return it returns the accountID
     * @throws IllegalHandleException if the handle has a space
     * @throws InvalidHandleException if the handle exists already
     */
    @Override
    public int createAccount(String handle) throws IllegalHandleException, InvalidHandleException {
```

```
36
37          //IllegalHandleException
38          for (int j = 0; j < this.accounts.size(); j
    ++) {
39              if (handle.equals(accounts.get(j).
    getHandle())) {
40                  throw new IllegalHandleException();
41              }
42          }
43
44          //InvalidHandleException
45          for (int i = 0; i < this.accounts.size(); i
    ++) {
46              if (handle.contains(" ")) {
47                  throw new InvalidHandleException();
48              }
49              if (handle != null && !handle.isEmpty
    ()) {
50                  throw new InvalidHandleException();
51              }
52          }
53
54          //Create account code:
55          int arrayLength = this.accounts.size();
56          int accountID = arrayLength + 1;
57          this.accounts.add(new Account(accountID,
    handle, "empty"));
58          return accountID;
59      }
60
61      /**
62       * Create an account with the given handle
63       *
64       * @param handle      account's handle.
65       * @param description account's description.
66       * @return returns the accountID
67       * @throws IllegalHandleException if the handle
     is illegal
68       * @throws InvalidHandleException If the handle
     is inavlid handle
69       */
70      @Override
71      public int createAccount(String handle, String
```

```
 71 description) throws IllegalHandleException,
    InvalidHandleException {
 72         //IllegalHandleException
 73         for (int j = 0; j < this.accounts.size();
    j++) {
 74             if (handle.equals(accounts.get(j).
    getHandle())) {
 75                 throw new IllegalHandleException
    ();
 76             }
 77         }
 78 
 79         //InvalidHandleException
 80         for (int i = 0; i < this.accounts.size();
    i++) {
 81             if (handle.contains(" ")) {
 82                 throw new InvalidHandleException
    ();
 83             }
 84             if (handle != null && !handle.isEmpty
    ()) {
 85                 throw new InvalidHandleException
    ();
 86             }
 87         }
 88 
 89         //Create new account code
 90         int arrayLength = this.accounts.size();
 91         int accountID = arrayLength + 1;
 92         this.accounts.add(new Account(accountID,
    handle, description));
 93 
 94         return accountID;
 95     }
 96 
 97     /**
 98      * Remove the account
 99      *
100      * @param id ID of the account.
101      * @throws AccountIDNotRecognisedException
    account id not recognized excpetion
102      */
103     @Override
```

```
104     public void removeAccount(int id) throws
    AccountIDNotRecognisedException {
105         //AccountIDNotRecognisedException
106         for (int i = 0; i < accounts.size(); i
    ++) {
107             if (!accounts.contains(accounts.get(i
    ).getId())) {
108                 throw new
    AccountIDNotRecognisedException();
109             }
110 
111         }
112 
113         // Loops through and checks if there is
    the same id
114         for (int i = 0; i < this.accounts.size();
    i++) {
115             if (id == this.accounts.get(i).getId
    ()) {
116                 this.accounts.remove(i);
117             }
118         }
119     }
120 
121     /**
122      * Remove the account
123      *
124      * @param handle account's handle.
125      * @throws HandleNotRecognisedException handle
     not recongized excpetion thrown
126      */
127     @Override
128     public void removeAccount(String handle)
    throws HandleNotRecognisedException {
129         //HandleNotRecognisedException
130         for (int i = 0; i < accounts.size(); i
    ++) {
131             if (!handle.equals(this.accounts.get(i
    ).getHandle())) {
132                 throw new
    HandleNotRecognisedException();
133             }
134 
```

```
135         }
136         //Loops through and removes account
137         for (int i = 0; i < this.accounts.size();
    i++) {
138             if (handle.equals(this.accounts.get(i
    ).getHandle())) {
139                 accounts.remove(i);
140             }
141         }
142     }
143 
144     /**
145      * Change the handle of the account.
146      *
147      * @param oldHandle account's old handle.
148      * @param newHandle account's new handle.
149      * @throws HandleNotRecognisedException handle
     not recongized excpetion
150      * @throws IllegalHandleException
    illegal handle expection thrown
151      * @throws InvalidHandleException
    illegal handle exception
152      */
153     @Override
154     public void changeAccountHandle(String
    oldHandle, String newHandle) throws
    HandleNotRecognisedException,
    IllegalHandleException, InvalidHandleException {
155         // Loops through and checks the handles,
    if true then set handle with new handle
156         for (int i = 0; i < this.accounts.size();
    i++) {
157             if (this.accounts.get(i).getHandle().
    equals(oldHandle)) {
158                 this.accounts.get(i).setHandle(
    newHandle);
159             }
160         }
161     }
162 
163     /**
164      * Update the description of the given account
165      *
```

```
166      * @param handle      handle to identify the
    account.
167      * @param description new text for description
    .
168      * @throws HandleNotRecognisedException handle
     not recognised expection
169      */
170     @Override
171     public void updateAccountDescription(String
    handle, String description) throws
    HandleNotRecognisedException {
172         //HandleNotRecognisedException
173         for (int i = 0; i < accounts.size(); i
    ++) {
174             if (!handle.equals(this.accounts.get(i
    ).getHandle())) {
175                 throw new
    HandleNotRecognisedException();
176             }
177         }
178
179         for (int i = 0; i < accounts.size(); i
    ++) {
180             //Loops through and checks if the
    handle is the same, if it is update the
    description.
181             if (accounts.get(i).getHandle().equals
    (handle)) {
182                 accounts.get(i).setDescription(
    description);
183             }
184         }
185     }
186
187     /**
188      * Show an account
189      *
190      * @param handle handle to identify the
    account.
191      * @return returns a string
192      * @throws HandleNotRecognisedException Handle
     Not Recognised Exception
193      */
```

```
194     @Override
195     public String showAccount(String handle)
    throws HandleNotRecognisedException {
196         final var account = findAccountByHandle(
    handle);
197 
198         if (account == null)
199             // no account was found with the given
     handle
200             throw new HandleNotRecognisedException
    ();
201 
202         return account.toString();
203     }
204 
205     /**
206      * Create a new post
207      *
208      * @param handle  handle to identify the
    account.
209      * @param message post message.
210      * @return The ID of the created post.
211      * @throws HandleNotRecognisedException Handle
     Not Recognised Exception
212      * @throws InvalidPostException         Thrown
     when the post is invalid
213      */
214     @Override
215     public int createPost(String handle, String
    message) throws HandleNotRecognisedException,
    InvalidPostException {
216         final var account = findAccountByHandle(
    handle);
217 
218         //HandleNotRecognisedException
219         if (account == null)
220             throw new HandleNotRecognisedException
    ();
221 
222         //Creates new post object and appends it
223         final var newPost = new Post(postId++,
    message, account);
224         posts.add(newPost);
```

```
225
226         return newPost.getId();
227     }
228
229     /**
230      * Endorse a post
231      *
232      * @param handle of the account endorsing a
    post.
233      * @param id     of the post being endorsed.
234      * @return The ID of the created endorsement
    post.
235      * @throws HandleNotRecognisedException Handle
     Not Recognized
236      * @throws PostIDNotRecognisedException Post
    ID Not Recognized
237      * @throws NotActionablePostException   The
    post is not actionable.
238      */
239     @Override
240     public int endorsePost(String handle, int id)
    throws HandleNotRecognisedException,
    PostIDNotRecognisedException,
    NotActionablePostException {
241         final var originalPost = findPostById(id);
242         final var endorser = findAccountByHandle(
    handle);
243
244         //PostIDNotRecognisedException
245         if (originalPost == null)
246             // no post was found with the given id
247             throw new PostIDNotRecognisedException
    ();
248
249         //PostIDNotRecognisedException
250         if (endorser == null)
251             // no account was found with the given
     handle
252             throw new HandleNotRecognisedException
    ();
253
254         //PostIDNotRecognisedException
255         if (originalPost instanceof
```

```
255 EndorsementPost)
256             // the post found is an endorsement
    post
257             throw new NotActionablePostException
    ();
258 
259         //Creates new endorsementPost and appends
    it
260         final var endorsementPost = new
    EndorsementPost(originalPost, postId++, endorser);
261 
262         posts.add(endorsementPost);
263 
264         return endorsementPost.getId();
265     }
266 
267     /**
268      * Comment on the given post.
269      *
270      * @param handle  of the account commenting a
    post.
271      * @param id      of the post being commented.
272      * @param message the comment post message.
273      * @return The ID of the created comment.
274      * @throws HandleNotRecognisedException Handle
     is not recognized
275      * @throws PostIDNotRecognisedException Post
    id is not recognized
276      * @throws NotActionablePostException   The
    post is not actionable
277      * @throws InvalidPostException         The
    post is invalid
278      */
279     @Override
280     public int commentPost(String handle, int id,
    String message) throws
    HandleNotRecognisedException,
    PostIDNotRecognisedException,
    NotActionablePostException, InvalidPostException {
281         final var parent = findPostById(id);
282         final var commenter = findAccountByHandle(
    handle);
283 
```

```
284
285         //PostIDNotRecognisedException
286         if (parent == null)
287             // no post was found with the given id
288             throw new PostIDNotRecognisedException
    ();
289
290         //HandleNotRecognisedException
291         if (commenter == null)
292             // no account was found with the given
     handle
293             throw new HandleNotRecognisedException
    ();
294
295         //NotActionablePostException
296         if (parent instanceof EndorsementPost)
297             // the post found is an endorsement
    post
298             throw new NotActionablePostException
    ();
299
300         //InvalidPostException
301         if (message.isEmpty() || message.length
    () > MAX_COMMENT_LENGTH)
302             // the given message is empty or the
    message is too long
303             throw new InvalidPostException();
304
305         //Creates new comment object and appends
    the comment
306         final var comment = new Comment(parent,
    postId++, commenter, message);
307
308         posts.add(comment);
309         parent.addComment(comment);
310
311         return comment.getId();
312     }
313
314     /**
315      * Delete the post with the given ID
316      *
317      * @param id ID of post to be removed.
```

```
318      * @throws PostIDNotRecognisedException The
    given ID is not recognized.
319      */
320     @Override
321     public void deletePost(int id) throws
    PostIDNotRecognisedException {
322         final var post = findPostById(id);
323 
324         //PostIDNotRecognisedException
325         if (post == null)
326             // no post was found with the given id
327             throw new PostIDNotRecognisedException
    ();
328 
329         posts.remove(post);
330     }
331 
332     /**
333      * Show details of a post with the given ID.
334      *
335      * @param id of the post to be shown.
336      * @return The details of the post in string.
337      * @throws PostIDNotRecognisedException
338      */
339     @Override
340     public String showIndividualPost(int id)
    throws PostIDNotRecognisedException {
341         final var post = findPostById(id);
342 
343         //PostIDNotRecognisedException
344         if (post == null)
345             // no post was found with the given id
346             throw new PostIDNotRecognisedException
    ();
347 
348         return post.toString();
349     }
350 
351     /**
352      * Show details of a post and its childre
    nposts.
353      *
354      * @param id of the post to be shown.
```

```
355         * @return The details in string.
356         * @throws PostIDNotRecognisedException
357         * @throws NotActionablePostException
358         */
359        @Override
360        public StringBuilder showPostChildrenDetails(
    int id) throws PostIDNotRecognisedException,
    NotActionablePostException {
361            final var post = findPostById(id);
362
363            //PostIDNotRecognisedException
364            if (post == null)
365                // no post was found with the given id
366                throw new PostIDNotRecognisedException
    ();
367
368            //NotActionablePostException
369            if (post instanceof EndorsementPost)
370                // the post found is an endorsement
    post
371                throw new NotActionablePostException
    ();
372
373            //Creates post children details using
    string builder
374            final var stringBuilder = new
    StringBuilder();
375
376            showPostChildrenDetails(post, 0,
    stringBuilder);
377
378            return stringBuilder;
379        }
380
381        /**
382         * Recursive entry of showPostChildrenDetails
    . params are used to track states of recursion.
383         *
384         * @param parent  The current parent {@link
    Post} of children.
385         * @param level   How deep the recursion is.
    Used for calculating indentation.
386         * @param builder The instance of {@link
```

```
386 StringBuilder} for storing the string
    representation of post-children details.
387      * @return A {@link StringBuilder} with (
    partial) post children details.
388      */
389     private StringBuilder showPostChildrenDetails(
    Post parent, int level, StringBuilder builder) {
390         final var indentation = " ".repeat(4 *
    level);
391 
392         //Checks the level of the object and
    creates a string
393         if (level == 0) {
394             builder.append(parent.toString()).
    append("\n");
395         } else {
396             final var lines = parent.toString().
    split("\n");
397             builder.append(lines[0]).append("\n");
398             for (int i = 1; i < 4; i++) {
399                 builder.append(indentation).append
    (lines[i]).append("\n");
400             }
401         }
402 
403         // only do a recursive call if this parent
     has comments aka children posts.
404         if (parent.hasComments()) {
405             builder.append(indentation).append("|\
    n");
406             for (final var comment : parent.
    getComments()) {
407                 // for each comment, print out
    their details (they will be treated as a parent)
408                 builder.append(indentation).append
    ("| > ");
409                 showPostChildrenDetails(comment,
    level + 1, builder);
410             }
411         }
412 
413         return builder;
414     }
```

```
415
416     /**
417      * @return the most endorsed post in social
    media.
418      */
419     @Override
420     public int getMostEndorsedPost() {
421         //Returns the most endorsed post
422         // the last most endorsed post found.
423         Post mostEndorsedPost = null;
424         for (final var post : posts) {
425             // if the endorsement count of this
    post is larger than the one we found previously
426             // replace that with this post for
    future comparison
427             if (mostEndorsedPost == null ||
    mostEndorsedPost.getEndorsements() < post.
    getEndorsements()) {
428                 mostEndorsedPost = post;
429             }
430         }
431         return mostEndorsedPost.getId();
432     }
433
434     /**
435      * @return the most endorsed account in social
     media.
436      */
437     @Override
438     public int getMostEndorsedAccount() {
439         // the last most endorsed account found.
440         Account mostEndorsedAccount = null;
441         for (final var account : accounts) {
442             // if the endorsement count of this
    account is larger than the one we found previously
443             // replace that with this account for
    future comparison
444             if (mostEndorsedAccount == null ||
    mostEndorsedAccount.getEndorsementCount() <
    account.getEndorsementCount()) {
445                 mostEndorsedAccount = account;
446             }
447         }
```

```
448             return mostEndorsedAccount.getId();
449         }
450 
451         /**
452          * Clears all data in social media.
453          */
454         @Override
455         public void erasePlatform() {
456             //Clears the arrays
457             accounts.clear();
458             posts.clear();
459             postId = 0;
460         }
461 
462         /**
463          * Save the data of social media.
464          *
465          * @param filename location of the file to be
    saved
466          * @throws IOException An error occurred when
    saving data.
467          */
468         @Override
469         public void savePlatform(String filename)
    throws IOException {
470             //Saves the platform to the specified name
471             final var stream = new ObjectOutputStream(
    new FileOutputStream(filename));
472             stream.writeObject(this);
473         }
474 
475         /**
476          * Loads a platform from a file.
477          *
478          * @param filename location of the file to be
    loaded
479          * @throws IOException             An error
    occurred when reading the file.
480          * @throws ClassNotFoundException
481          */
482         @Override
483         public void loadPlatform(String filename)
    throws IOException, ClassNotFoundException {
```

```
484         //Loads the specified file name that has
    been serialized
485         final var stream = new ObjectInputStream(
    new FileInputStream(filename));
486         final var obj = stream.readObject();
487 
488         if (obj instanceof SocialMedia) {
489             accounts = ((SocialMedia) obj).
    accounts;
490             posts = ((SocialMedia) obj).posts;
491         }
492     }
493 
494     /**
495      * @return The number of account registered on
     the social media.
496      */
497     @Override
498     public int getNumberOfAccounts() {
499         int totalAccounts = this.accounts.size();
500         return totalAccounts;
501     }
502 
503     /**
504      * @return The number of posts made on the
    social media.
505      */
506     @Override
507     public int getTotalOriginalPosts() {
508         //Filters through and sorts all of the
    original posts
509         return posts.stream()
510                 .filter(post -> !(post instanceof
    EndorsementPost || post instanceof Comment))
511                 .collect(Collectors.toCollection(
    ArrayList::new))
512                 .size();
513     }
514 
515     /**
516      * @return number of endorsement posts on
    social media.
517      */
```

```
518     @Override
519     public int getTotalEndorsmentPosts() {
520         return posts.stream()
521                 .filter(post -> post instanceof
    EndorsementPost)
522                 .collect(Collectors.toCollection(
    ArrayList::new))
523                 .size();
524     }
525 
526     /**
527      * @return number of comments on social media.
528      */
529     @Override
530     public int getTotalCommentPosts() {
531         return posts.stream()
532                 .filter(post -> post instanceof
    Comment)
533                 .collect(Collectors.toCollection(
    ArrayList::new))
534                 .size();
535     }
536 
537 
538     /**
539      * Finds the account with the given handle.
540      *
541      * @param handle The handle of the account
542      * @return the account with the given handle,
    or null if none is found.
543      */
544     private Account findAccountByHandle(String
    handle) {
545         return accounts.stream()
546                 .filter(account -> account.
    getHandle().equals(handle))
547                 .findFirst()
548                 .orElse(null);
549     }
550 
551     /**
552      * Finds a post on social media with the given
     id.
```

```
553      *
554      * @param postId The ID of the post
555      * @return The post found, or null if none is
    found.
556      */
557     private Post findPostById(int postId) {
558         return posts.stream()
559                 .filter(post -> post.getId() ==
    postId)
560                 .findFirst()
561                 .orElse(null);
562     }
563 }
564
```

```java
package socialmedia.models;

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

public class Post implements Serializable {
    private int id;

    private String message;

    private Account author;

    private ArrayList<Post> comments;

    private int endorsements;

    public Post(int id, String message, Account author) {
        this.id = id;
        this.message = message;
        this.author = author;
        this.endorsements = 0;
        comments = new ArrayList<>();
    }

    public int getId() {
        return id;
    }

    public String getMessage() {
        return message;
    }

    public Account getAuthor() {
        return author;
    }

    public boolean hasComments() {
        return comments.size() > 0;
    }

    public ArrayList<Post> getComments() {
        return comments;
```

```
43      }
44
45      public void addComment(Comment comment) {
46          comments.add(comment);
47      }
48
49      public int getEndorsements() {
50          return endorsements;
51      }
52
53      protected void endorse() {
54          endorsements++;
55      }
56
57      @Override
58      public String toString() {
59          return "ID: " + id + "\n" +
60                  "Account: " + author.getHandle() +
   "\n" +
61                  "No. endorsements: " + endorsements
    + " | No. comments: " + comments.size() + "\n" +
62                  message;
63      }
64  }
65
```

```java
package socialmedia.models;

import java.io.Serializable;

public class Account implements Serializable {
    private int id;
    private String handle;
    private String description;

    /**
     * The number of endorsements that this {@link Account} has received.
     */
    private int endorsementCount;

    /**
     * The number of posts that this {@link Account} has posted.
     */
    private int postCount;

    public Account(int id, String handle, String description) {
        this.id = id;
        this.handle = handle;
        this.description = description;
        endorsementCount = 0;
    }

    public void setDescription(String newDesc) {
        this.description = newDesc;
    }

    public void setHandle(String newHandle) {
        this.handle = newHandle;
    }

    public int getId() {
        return id;
    }

    public String getHandle() {
        return handle;
```

```
41      }
42
43      public String getDescription() {
44          return description;
45      }
46
47      public int getEndorsementCount() {
48          return endorsementCount;
49      }
50
51      /**
52       * Endorse this {@link Account}
53       */
54      public void endorse() {
55          endorsementCount++;
56      }
57
58      public void increasePostCount() {
59          postCount++;
60      }
61
62      public void decreasePostCount() {
63          postCount--;
64      }
65
66      @Override
67      public String toString() {
68          return "\n"
69                  + "\n"
70                  + "ID: " + id + "\n"
71                  + "Handle: " + handle + "\n"
72                  + "Description: " + description + "
    \n"
73                  + "Post count: " + postCount + "\n"
74                  + "Endorse count: " +
    endorsementCount + "\n"
75                  + "\n";
76      }
77  }
78
```

```java
package socialmedia.models;

public class Comment extends Post {
    private Post parent;

    public Comment(Post parent, int id, Account commenter, String comment) {
        super(id, comment, commenter);
        this.parent = parent;
    }
}

```

```java
package socialmedia.models;

public class EndorsementPost extends Post {
    private Post originalPost;

    public EndorsementPost(Post originalPost, int id, Account endorser) {
        super(id, "EP@" + originalPost.getAuthor().getHandle() + ": " + originalPost.getMessage(), endorser);
        this.originalPost = originalPost;
        originalPost.endorse();
        originalPost.getAuthor().endorse();
    }
}

```