1 Account.java

```java
package socialmedia;

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

/**
 * This class represents an account on a social media platform, contains the posts in the account
 */
public class Account implements Serializable {

    // private instance variables
    private String handle;
    private String description;
    private int accountId;
    private static int accountIdCounter = 0;
    // Arraylist contains all posts
    private ArrayList<Post> posts = new ArrayList<>();

    /**
     * Constructor of a new Account object with the given handle and description.
     *
     * @param handle    the handle for the account
     * @param description the description for the account
     */
    public Account(String handle){
        this.handle = handle;
        this.accountId = accountIdCounter++;
    }

    // public getters and setter for private instance
    public String getHandle(){
        return handle;
    }
    public String getDescription(){
        return description;
    }
    public int getAccountId(){
        return accountId;
    }


    public void setHandle(String handle){
        this.handle = handle;
    }
    public void setDescription(String description){
        this.description = description;
    }
    public void setAccountId(int accountId){
        this.accountId = accountId;
    }

```

```java
    /**
     * Returns a list of all the posts made by the account.
     *
     * @return a list of all the posts made by the account
     */
    public ArrayList<Post> getPosts() {
        return posts;
    }

    /**
     * Adds a new post to the account's list of posts.
     *
     * @param post the post to add
     */
    public void addPost(Post post) {
        posts.add(post);
    }

    /**
     * Removes a post from the account's list of posts.
     *
     * @param post the post to remove
     */
    public void removePost(Post post) {
        posts.remove(post);
    }

    /**
     * Returns a summary of the account, including its ID, handle, description, number of posts, and number
         of endorsements.
     * @return a summary of the account
     */
    public String getSummary(){
        int postCount = 0;
        int endorseCount = 0;
        for (Post post: posts){
            postCount++;
            endorseCount += post.getEndorsements().size();
        }
        String summary = String.format("ID: %s\nHandle: %s\nDescription: %s\nPost count: %d\nEndorse count:
             %d",
        accountId, handle, description, postCount, endorseCount);

        return summary;
    }

    /**
     * Returns the current account ID counter.
     * @return the current account ID counter
     */
    public static int getCurrentId() {
        return accountIdCounter;
    }

    /**
```

```
106      * Sets the current account ID counter.
107      * @param currentId the new account ID counter
108      */
109     public static void setCurrentId(int currentId) {
110         accountIdCounter = currentId;
111     }
112 
113 }
```

2 Post.java

```
1  package socialmedia;
2  
3  import java.util.ArrayList;
4  import java.io.Serializable;
5  
6  /**
7   * Post class represents each post in the system, contains endorsements and comments
8   * in each post and methods to deal with them
9   */
10 public class Post implements Serializable {
11 
12     //private attributes
13     private int postId;
14     private String message;
15     private boolean endorsement;
16     private boolean commentPost;
17     private static int postIdCounter = 0;
18     private int originalPostId;
19     private ArrayList<Post> endorsements = new ArrayList<Post>();
20     private ArrayList<Post> comments = new ArrayList<Post>();
21 
22     /** Constructor for race class
23     *
24     * @param message content of new post
25     */
26     public Post(String message) {
27         this.postId = postIdCounter++;
28         this.message = message;
29         this.endorsement = false;
30         this.commentPost = false;
31     }
32 
33     //getter methods
34     public int getPostId() {
35         return postId;
36     }
37     public String getMessage() {
38         return message;
39     }
40 
41     public boolean isEndorsement(){
42         return endorsement;
43     }
```

```
44
45      public boolean isComment(){
46          return commentPost;
47      }
48
49      public ArrayList<Post> getEndorsements() {
50          return endorsements;
51      }
52
53      public ArrayList<Post> getComments() {
54          return comments;
55      }
56
57      //setter methods
58      public void setPostId(int postId) {
59          this.postId = postId;
60      }
61      public void setMessage(String message) {
62          this.message = message;
63      }
64
65      public void setIsEndorsement(boolean endorsement){
66          this.endorsement = endorsement;
67      }
68
69      public void setIsCommentPost(boolean commentPost){
70          this.commentPost = commentPost;
71      }
72
73      /** Adds a new endorsement to the post
74      *
75      * @param endorsement Endorsement to be added to the list of endorsements
76      */
77      public void addEndorsement(Post endorsement) {
78          this.endorsements.add(endorsement);
79      }
80
81      /** Remove a endorsement from the post
82      *
83      * @param endorsement Endorsement to be removed from the list of endorsements
84      */
85      public void removeEndorsement(Post endorsement) {
86          this.endorsements.remove(endorsement);
87      }
88
89      /** Adds a new comment to the post
90      *
91      * @param comment comment to be added to the list of comments
92      */
93      public void addComment(Post comment){
94          this.comments.add(comment);
95      }
96
97      /** Removes a comment from the post
98      *
```

```
99      * @param comment comment to be removed from the list of comments
100     */
101     public void removeComment(Post comment){
102         this.comments.remove(comment);
103     }
104
105     /** get the post id for the post being commented on
106      *
107      * @return the id of the post being commented on
108      */
109     public int getOriginalPostId() {
110         return originalPostId;
111     }
112
113     /** set the post id for the post being commented on
114      *
115      *
116      */
117     public void setOriginalPostId(int originalPostId) {
118         this.originalPostId = originalPostId;
119     }
120
121     /**
122      * Returns the current post ID counter.
123      * @return the current post ID counter
124      */
125     public static int getCurrentId() {
126         return postIdCounter;
127     }
128
129     /**
130      * Sets the current post ID counter.
131      * @param currentId the new post ID counter
132      */
133     public static void setCurrentId(int currentId) {
134         postIdCounter = currentId;
135     }
136
137
138
139
140
141 }
```

3 SocialMedia.java

```
1  package socialmedia;
2
3  import java.io.IOException;
4  import java.util.ArrayList;
5  import java.util.List;
6  import java.io.*;
7  import java.util.Comparator;
8
```

```java
/**
 * SocialMedia is a minimally compiling, but non-functioning implementor
 * of the MiniSocialMediaPlatform interface.
 *
 * @author Chit Hin Ernest Cheong
 * @version 1.0
 */

public class SocialMedia implements MiniSocialMediaPlatform {

   // attributes which hold the accounts in the social media
   private ArrayList<Account> accounts = new ArrayList<>();

   @Override
   public int createAccount(String handle) throws IllegalHandleException, InvalidHandleException {
      // check if the handle is illegal or duplicate
      if (handle.isEmpty() || handle.length() > 30 || handle.indexOf(' ') >= 0) {
         throw new InvalidHandleException();
      }
      for (Account account : accounts) {
         if (account.getHandle().equals(handle)) {
            throw new IllegalHandleException();
         }
      }
      // create a new account and add it to the list of accounts
      Account newAccount = new Account(handle);
      accounts.add(newAccount);
      return newAccount.getAccountId();
   }

   @Override
   public void removeAccount(int id) throws AccountIDNotRecognisedException {
      // use the size() method to get the size of the accounts list and loop over it using a standard for
          loop
      for (int i = 0; i < accounts.size(); i++) {
         Account account = accounts.get(i);
         if (account.getAccountId() == id) {
            // remove account from the list of accounts
            accounts.remove(i);
            return;
         }
      }
      throw new AccountIDNotRecognisedException();

   }

   @Override
   public void changeAccountHandle(String oldHandle, String newHandle)
         throws HandleNotRecognisedException, IllegalHandleException, InvalidHandleException {
       //check if the new handle is valid
      if (newHandle.isEmpty() || newHandle.length() > 30 || newHandle.indexOf(' ') >= 0) {
         throw new InvalidHandleException();
      }
      // Check if the old handle is recognised
      boolean accountFound = false;
```

```
63          for (Account account : accounts) {
64              if (account.getHandle().equals(oldHandle)) {
65                  accountFound = true;
66                  break;
67              }
68              if (account.getHandle().equals(newHandle)) {
69                  throw new IllegalHandleException();
70              }
71          }
72          if (!accountFound) {
73              throw new HandleNotRecognisedException();
74          }
75          // search the account and change the handle
76          for (Account account : accounts) {
77              if (account.getHandle().equals(oldHandle)) {
78                  account.setHandle(newHandle);
79                  return;
80              }
81          }
82
83      }
84
85      @Override
86      public String showAccount(String handle) throws HandleNotRecognisedException {
87          // Declare a null account variable to store the matched account
88          Account account = null;
89          // Iterate through the accounts to find the one that matches the handle
90          for (Account a : accounts) {
91              if (a.getHandle().equals(handle)) {
92                  // Store the matched account in the account variable
93                  account = a;
94                  break;
95              }
96          }
97          // If the account variable is still null, that means no account matched the handle, so throw an
                exception
98          if (account == null) {
99              throw new HandleNotRecognisedException();
100         }
101         return account.getSummary();
102
103     }
104
105     @Override
106     public int createPost(String handle, String message) throws HandleNotRecognisedException,
            InvalidPostException {
107         // Declare a null account variable to store the matched account
108         Account account = null;
109         for (Account a : accounts) {
110             if (a.getHandle().equals(handle)) {
111                 account = a;
112                 break;
113             }
114         }
115         // If the account variable is still null, that means no account matched the handle, so throw an
```

```
            exception
116         if (account == null) {
117             throw new HandleNotRecognisedException();
118         }
119
120           // Check if the message is empty or has more than 100 characters
121           if (message == null || message.isEmpty() || message.length() > 100) {
122               throw new InvalidPostException();
123           }
124
125           // Create a new post for the account
126           Post post = new Post(message);
127           account.addPost(post);
128
129           // Return the sequential ID of the created post
130           return post.getPostId();
131       }
132
133       @Override
134       public int endorsePost(String handle, int id)
135               throws HandleNotRecognisedException, PostIDNotRecognisedException, NotActionablePostException {
136           // Declare a null account variable to store the matched account
137           Account account = null;
138           for (Account a : accounts) {
139               if (a.getHandle().equals(handle)) {
140                   account = a;
141                   break;
142               }
143           }
144           // If the account variable is still null, that means no account matched the handle, so throw an
                exception
145           if (account == null) {
146               throw new HandleNotRecognisedException();
147           }
148
149           // Check if the post exists and is not already an endorsement
150           Post post = null;
151           for (Post p : account.getPosts()) {
152               if (p.getPostId() == id) {
153                   post = p;
154                   break;
155               }
156           }
157           if (post == null) {
158               throw new PostIDNotRecognisedException();
159           } else if (post.isEndorsement()) {
160               throw new NotActionablePostException();
161           }
162
163           // Create the endorsement post
164           String message = "EP@" + account.getHandle() + ": " + post.getMessage();
165           Post endorsementPost = new Post(message);
166           endorsementPost.setIsEndorsement(true);
167
168           // Add the endorsement post to the endorsements array in the post class
```

```
169         post.addEndorsement(endorsementPost);
170
171         // Add the endorsement post to the account
172           account.addPost(endorsementPost);
173
174           // Return the ID of the endorsement post
175           return endorsementPost.getPostId();
176     }
177
178     @Override
179     public int commentPost(String handle, int id, String message) throws HandleNotRecognisedException,
180           PostIDNotRecognisedException, NotActionablePostException, InvalidPostException {
181         // Declare a null account variable to store the matched account
182         Account account = null;
183           for (Account a : accounts) {
184              if (a.getHandle().equals(handle)) {
185               account = a;
186               break;
187             }
188           }
189         // If the account variable is still null, that means no account matched the handle, so throw an
              exception
190         if (account == null) {
191            throw new HandleNotRecognisedException();
192         }
193         // Check if the post exists, vaild and is not already an endorsement
194         Post post = null;
195         for (Post p : account.getPosts()) {
196            if (p.getPostId() == id) {
197               post = p;
198               break;
199            }
200         }
201           if (post == null) {
202               throw new PostIDNotRecognisedException();
203           } else if (post.isEndorsement()) {
204               throw new NotActionablePostException();
205           } else if (message == null || message.isEmpty() || message.length() > 100) {
206            throw new InvalidPostException();
207         }
208         // Creates the comment post
209         Post comment = new Post(message);
210         comment.setIsCommentPost(true);
211         comment.setOriginalPostId(id);
212
213         // Add the comment post to the account
214           account.addPost(comment);
215
216         // Add the comment post to the endorsements array in the post class
217         post.addComment(comment);
218
219         // Return the ID of the post
220         return comment.getPostId();
221     }
222
```

```
223     @Override
224     public void deletePost(int id) throws PostIDNotRecognisedException {
225        // Initialize a boolean flag to track if the post with the given ID was found
226        boolean found = false;
227        // Loop through all accounts and posts of each account
228        for (Account account: accounts){
229           for (Post post: account.getPosts()) {
230               if (id == post.getPostId()){
231                 found = true;
232                 // If the post is an endorsement, remove it from all accounts that endorsed it
233                 if (post.isEndorsement()){
234                         for (Post endorsements : post.getEndorsements()) {
235                       endorsements.setIsEndorsement(false);
236                       // Remove the endorsement post from its account
237                       account.removePost(endorsements);
238                    }
239                 }
240                 // If the post is a comment, set its message to a generic message indicating the original
                        content was removed
241                 if (post.isComment()){
242                    for (Post reply : account.getPosts()) {
243                             if (reply.getOriginalPostId() == id) {
244                                 reply.setMessage("The original content was removed from the system and is no
                                        longer available.");
245                             }
246                         }
247                 }
248                 // Remove the post from the account it belongs to
249                  account.removePost(post);
250                  break;
251           }
252        }
253     }
254     // If no post with the given ID was found, throw an exception
255     if (!found){
256        throw new PostIDNotRecognisedException();
257     }
258
259  }
260
261  @Override
262  public String showIndividualPost(int id) throws PostIDNotRecognisedException {
263     // Initialize a boolean flag to track if the post with the given ID was found
264       boolean found = false;
265       // Loop through all accounts
266       for (Account account: accounts){
267        // Loop through all posts of each account
268        for (Post post: account.getPosts()) {
269           // If the post ID matches the given ID, mark it as found
270            if (id == post.getPostId()){
271                  found = true;
272                  // Get the number of endorsements and comments for this post
273                  int endorsementCount = post.getEndorsements().size();
274                  int commentCount = post.getComments().size();
275                  // Return a formatted string with post information
```

```
                return String.format("ID: %d\nAccount: %s\nNo. endorsements: %d | No. comments: %d\n%s",
                post.getPostId(), account.getHandle(), endorsementCount, commentCount, post.getMessage());
            }
        }
    }
    // If no post with the given ID was found, throw an exception
    if (!found){
     throw new PostIDNotRecognisedException();
   }
   // This line will never be reached, since the method either returns a string or throws an exception
   return null;
 }

@Override
public StringBuilder showPostChildrenDetails(int id)
     throws PostIDNotRecognisedException, NotActionablePostException {
    // Declare a null post variable to store the matched post
    Post postFound = null;
    // Loop through all accounts
    for (Account account: accounts){
     // Loop through all posts of each account
     for (Post post: account.getPosts()) {
           if (post.getPostId() == id) {
           // Assign the matched post to the postFound variable
               postFound = post;
               break;
           }
       }
    }
    // If no post with the given ID is found, throw a PostIDNotRecognisedException
    if (postFound == null) {
        throw new PostIDNotRecognisedException();
    }
    // If the matched post is an endorsement or not a comment, throw a NotActionablePostException
    else if (postFound.isEndorsement() || !postFound.isComment()) {
        throw new NotActionablePostException();
    }
    // If the matched post is a comment, proceed to get its children details
    else {
        // Create a StringBuilder to store the post details and its children's details
        StringBuilder sb = new StringBuilder();
        // Append the details of the root level post to the StringBuilder
        sb.append(showIndividualPost(id)).append("\n");
        // Get the comments of the post and add them to an ArrayList of replies
        ArrayList<Post> replies = new ArrayList<>();
        replies.addAll(postFound.getComments());
        // Sort the replies by their post IDs and append the details of each reply to the StringBuilder
        replies.sort(new Comparator<Post>() {
        public int compare(Post o1, Post o2) {
           return Integer.compare(o1.getPostId(), o2.getPostId());
        }
     });
        for (Post reply : replies) {
            sb.append("| >").append(showIndividualPost(reply.getPostId())).append("\n");
        }
```

```
            return sb;
        }
    }

    @Override
    public int getMostEndorsedPost() {
        // Loop through the account and their posts to find the most endorsed post
        int mostEndorsements = 0;
        int mostEndorsedPostId = 0;
          for (Account account: accounts){
           for (Post post: account.getPosts()) {
              // Get the number of endorsements for the post
                  int endorsementCount = post.getEndorsements().size();

              // Check if the current post has more endorsements than the current most endorsed post
                  if (endorsementCount > mostEndorsements) {
                      mostEndorsements = endorsementCount;
                      mostEndorsedPostId = post.getPostId();
                  }
              }
          }
        return mostEndorsedPostId;
    }

    @Override
    public int getMostEndorsedAccount() {
        int mostEndorsements = 0;
        int mostEndorsedAccountId = 0;
        // Loop throught the account and their post
          for (Account account : accounts) {
              int endorsementCount = 0;
              for (Post post : account.getPosts()) {
              // Get the number of endorsements for each post of the account
                  endorsementCount += post.getEndorsements().size();
              }
              // Check if the current account has more endorsements than the current most endorsed account
              if (endorsementCount > mostEndorsements) {
                  mostEndorsements = endorsementCount;
                  mostEndorsedAccountId = account.getAccountId();
              }
          }

          return mostEndorsedAccountId;
    }

    @Override
    public void erasePlatform() {
        // Reset counters
        post.setCurrentId(0);
        account.setCurrentId(0);

        // Loop through all accounts and remove posts
        for (Account account : accounts) {
            account.getPosts().clear();
        }
```

```
386
387         // Clear accounts list
388         accounts.clear();
389
390         assert(accounts.size() == 0);
391
392
393     }
394
395     @Override
396     public void savePlatform(String filename) throws IOException {
397         try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(filename))) {
398             // Save the list of accounts
399             out.writeObject(accounts);
400
401             // Save the current ID counts
402             out.writeInt(Account.getCurrentId());
403             out.writeInt(Post.getCurrentId());
404             out.close();
405         }
406
407     }
408
409     @Override
410     public void loadPlatform(String filename) throws IOException, ClassNotFoundException {
411         try (ObjectInputStream in = new ObjectInputStream(new FileInputStream(filename))) {
412             // Load the list of accounts
413             accounts = (ArrayList<Account>) in.readObject();
414
415             // Load the current ID counts
416             int accountIdCounter = in.readInt();
417             int postIdCounter = in.readInt();
418             Account.setCurrentId(accountIdCounter);
419             Post.setCurrentId(postIdCounter);
420         }
421
422     }
423
424 }
```