```java
package socialmedia;

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

public class SocialMedia implements SocialMediaPlatform{
    //stores the account in a hashmap to quickly find a
    //account by it's handle
    private HashMap<String,Account> accountsByHandle;

    //stores the account in a hashmap to quickly find a
    //account by it's ID
    private HashMap<Integer,Account> accountsByID;

    //stores posts against its ID;
    private HashMap<Integer,Post> posts;
    private HashMap<Integer,Comment> comments;
    private HashMap<Integer,EndorsementPost> endorsementPosts;



    //used to assign a unique ID to a account
    private int accountIDTracker;

    //used to assign unique ID to a post
    private int postIdTracker;

    //empty post
    private Post emptyPost;

    private int childComments;

    //constructor to initialize the variables
    public SocialMedia() {
        accountsByHandle=new HashMap<>();
        accountsByID =new HashMap<>();
        postIdTracker=accountIDTracker=0;
        endorsementPosts = new HashMap<>();
        posts = new HashMap<>();
        comments = new HashMap<>();
        // generic empty post template
        emptyPost=new Post(-1,null,"The original content was removed" +
                " from the system and is no longer available.");

    }

    @Override
    public int createAccount(String handle, String description) throws
IllegalHandleException, InvalidHandleException {
        //checking if the handle exists already
        if(accountsByHandle.containsKey(handle))throw new IllegalHandleException();

        //check if the handle is valid
        if(handle.isEmpty() || handle.length()>30 || handle.matches("\\s"))throw new
InvalidHandleException();

        Account newAccount=new Account(handle,description,accountIDTracker);
        //save the account
        accountsByHandle.put(handle,newAccount);
        accountsByID.put(accountIDTracker,newAccount);

        //update the tracker to assign in the next create call
        accountIDTracker++;

        return accountIDTracker-1;

    }

    @Override
    public void removeAccount(String handle) throws HandleNotRecognisedException {
        //checking if the handle does not exists
```

```
 70         if(!accountsByHandle.containsKey(handle)) throw new HandleNotRecognisedException
    ();
 71
 72         Account removedAC=accountsByHandle.get(handle);
 73         //delete all its posts first
 74         ArrayList<Integer> toRemove = new ArrayList<>(removedAC.getPosts());
 75         toRemove.addAll(removedAC.getComments());
 76         toRemove.addAll(removedAC.getEndorsementPosts());
 77         for(int id:toRemove) {
 78             try {
 79                 deletePost(id);
 80             } catch (PostIDNotRecognisedException e) {
 81                 e.printStackTrace();
 82             }
 83         }
 84
 85
 86         //remove the account
 87         accountsByHandle.remove(removedAC.getHandle());
 88         accountsByID.remove(removedAC.getID());
 89
 90
 91     }
 92
 93     @Override
 94     public void updateAccountDescription(String handle, String description) throws
    HandleNotRecognisedException {
 95         //checking if the handle does not exists
 96         if(!accountsByHandle.containsKey(handle))throw  new HandleNotRecognisedException
    ();
 97
 98         //update
 99         accountsByHandle.get(handle).setDescription(description);
100
101     }
102
103     @Override
104     public int getNumberOfAccounts() {
105         return accountsByHandle.size();
106     }
107
108     @Override
109     public int getTotalOriginalPosts() {
110         return posts.size();
111     }
112
113     @Override
114     public int getTotalEndorsmentPosts() {
115         return endorsementPosts.size();
116     }
117
118     @Override
119     public int getTotalCommentPosts() { return comments.size(); }
120
121     @Override
122     public int createAccount(String handle) throws IllegalHandleException,
    InvalidHandleException {
123         //checking if the handle exists already
124         if(accountsByHandle.containsKey(handle))throw  new IllegalHandleException();
125
126         //check if the handle is valid
127         if(handle.isEmpty() || handle.length()>30 || handle.matches("\\s"))throw new
    InvalidHandleException();
128
129         Account newAccount=new Account(handle,"",accountIDTracker);
130         //save the account
131         accountsByHandle.put(handle,newAccount);
132         accountsByID.put(accountIDTracker,newAccount);
133
134         //update the tracker to assign in the next create call
135         accountIDTracker++;
```

```
136
137         return accountIDTracker-1;
138
139     }
140
141     @Override
142     public void removeAccount(int id) throws AccountIDNotRecognisedException {
143         //checking if the handle does not exists
144         if(!accountsByID.containsKey(id))throw  new AccountIDNotRecognisedException();
145
146         //remove the account
147         Account removedAC=accountsByID.remove(id);
148         accountsByHandle.remove(removedAC.getHandle());
149
150     }
151
152     @Override
153     public void changeAccountHandle(String oldHandle, String newHandle) throws
    HandleNotRecognisedException, IllegalHandleException, InvalidHandleException {
154         // checking if the handle does not exist
155         if(!accountsByHandle.containsKey(oldHandle))throw  new
    HandleNotRecognisedException();
156
157         // check if new handle already exists
158         if(accountsByHandle.containsKey(newHandle))throw new IllegalHandleException();
159
160         // check if the handle is valid
161         if(newHandle.isEmpty() || newHandle.length()>30 || newHandle.matches("\\s"))
    throw new InvalidHandleException();
162
163         // update the maps and account
164         Account account=accountsByHandle.remove(oldHandle);
165         account.setHandle(newHandle);
166
167         accountsByHandle.put(newHandle,account);
168
169     }
170
171     @Override
172     public String showAccount(String handle) throws HandleNotRecognisedException {
173         if(!accountsByHandle.containsKey(handle))throw  new HandleNotRecognisedException
    ();
174
175         Account account=accountsByHandle.get(handle);
176         StringBuilder stringBuilder=new StringBuilder();
177         stringBuilder.append("ID: ").append(account.getID()).append("\n")
178                 .append("Handle: ").append(handle).append("\n")
179                 .append("Description: ").append(account.getDescription()).append("\n")
180                 .append("Post count: ").append(account.getPosts().size()+account.
    getComments().size()+account.getEndorsementPosts().size())
181                 .append("\n")
182                 .append("Endorse count: ");
183
184         int endorsementCount=0;
185         for(int id:account.getPosts()){
186             endorsementCount+=posts.get(id).getEndorsementPosts().size();
187         }
188         for(int id:account.getComments()){
189             endorsementCount+=comments.get(id).getEndorsementPosts().size();
190         }
191
192         stringBuilder.append(endorsementCount).append("\n");
193
194         return stringBuilder.toString();
195     }
196
197     @Override
198     public int createPost(String handle, String message) throws
    HandleNotRecognisedException, InvalidPostException {
199         //checking if the handle does not exist
200         if(!accountsByHandle.containsKey(handle))throw  new HandleNotRecognisedException
```

```
200 ();
201
202         if(message.isEmpty() || message.length()>100)throw  new InvalidPostException();
203
204         Post post=new Post(postIdTracker,accountsByHandle.get(handle),message);
205         //store this post
206         posts.put(postIdTracker,post);
207
208         //add a reference to this post to its owner account
209         post.getAccount().getPosts().add(postIdTracker);
210
211         postIdTracker++;
212         return post.getID();
213
214
215     }
216
217     @Override
218     public int endorsePost(String handle, int id) throws HandleNotRecognisedException,
    PostIDNotRecognisedException, NotActionablePostException {
219         // checking if the handle does not exist
220         if(!accountsByHandle.containsKey(handle))throw  new HandleNotRecognisedException
    ();
221
222         // checking if the post is available as post or comments or endorsementPosts or
    not
223         if(!(posts.containsKey(id) || comments.containsKey(id) || endorsementPosts.
    containsKey(id))){
224             throw new PostIDNotRecognisedException();
225         }
226         // endorsable posts are not endorsable.
227         if(endorsementPosts.containsKey(id))throw  new NotActionablePostException();
228
229         EndorsementPost post=new EndorsementPost(null,postIdTracker,accountsByHandle.get
    (handle));
230
231         // store this post
232         endorsementPosts.put(postIdTracker,post);
233
234         // add this post to its owner
235         post.getAccount().getEndorsementPosts().add(postIdTracker);
236
237         if(posts.containsKey(id)){
238
239             // the main post should refer to this post
240             posts.get(id).getEndorsementPosts().add(postIdTracker);
241
242             // add a reference to the main post
243             post.setParentPost(posts.get(id));
244         }
245         else if(comments.containsKey(id)){
246
247             //the main post should refer to this post
248             comments.get(id).getEndorsementPosts().add(postIdTracker);
249
250             //add a reference to the main post
251             post.setParentPost(comments.get(id));
252         }
253
254         //create the message of the endorsementPost
255         post.createMessage();
256
257         postIdTracker++;
258
259         return post.getID();
260
261     }
262
263     @Override
264     public int commentPost(String handle, int id, String message) throws
    HandleNotRecognisedException, PostIDNotRecognisedException, NotActionablePostException,
```

```
264 InvalidPostException {
265         // checking if the handle does not exist
266         if(!accountsByHandle.containsKey(handle))throw  new HandleNotRecognisedException
    ();
267 
268         // checking if the post is available as post or comments or endorsementPosts or
    not
269         if(!(posts.containsKey(id) || comments.containsKey(id) || endorsementPosts.
    containsKey(id))){
270             throw new PostIDNotRecognisedException();
271         }
272         if(endorsementPosts.containsKey(id))throw new NotActionablePostException();
273 
274         if(message.isEmpty() || message.length()>100)throw new InvalidPostException();
275 
276 
277         Comment comment=new Comment(postIdTracker,accountsByHandle.get(handle),message,
    null);
278 
279         //store this post
280         comments.put(postIdTracker,comment);
281 
282         //add this post to its owner
283         comment.getAccount().getComments().add(postIdTracker);
284 
285         if(posts.containsKey(id)){
286             // the main post should refer to this post
287             posts.get(id).getComments().add(postIdTracker);
288 
289             // add a reference to the main post
290             comment.setParentPost(posts.get(id));
291         }
292         else if(comments.containsKey(id)){
293             // the main post should refer to this post
294             comments.get(id).getComments().add(postIdTracker);
295 
296             // add a reference to the main post
297             comment.setParentPost(comments.get(id));
298         }
299 
300         postIdTracker++;
301         return comment.getID();
302 
303     }
304 
305     @Override
306     public void deletePost(int id) throws PostIDNotRecognisedException {
307         if(!(posts.containsKey(id) || comments.containsKey(id) || endorsementPosts.
    containsKey(id))){
308             throw new PostIDNotRecognisedException();
309         }
310 
311         if(posts.containsKey(id)){
312             Post post=posts.remove(id);
313 
314             // remove all its endorsement
315             for(int endorsements:post.getEndorsementPosts()){
316                 EndorsementPost endorsementPost=endorsementPosts.remove(endorsements);
317                 // remove the endorsement post from its owner account
318                 endorsementPost.getAccount().getEndorsementPosts().remove(
    endorsementPost.getID());
319             }
320             // all its comments should refer to the generic empty post
321             for(int commentsID:post.getComments()){
322                 comments.get(commentsID).setParentPost(emptyPost);
323             }
324             // finally remove the post from its owner account
325             post.getAccount().getPosts().remove(post.getID());
326         }
327         else if(comments.containsKey(id)){
328             Comment post=comments.remove(id);
```

```
329
330             //remove all its endorsement
331             for(int endorsements:post.getEndorsementPosts()){
332                 EndorsementPost endorsementPost=endorsementPosts.remove(endorsements);
333                 //remove the endorsement post from its owner account
334                 endorsementPost.getAccount().getEndorsementPosts().remove(
    endorsementPost.getID());
335             }
336             //all its comments should refer to the generic empty post
337             for(int commentsID:post.getComments()){
338                 comments.get(commentsID).setParentPost(emptyPost);
339             }
340
341             //remove the post from its owner account
342             post.getAccount().getComments().remove(post.getID());
343
344             //finally remove the reference from the comments parentPost
345             post.getParentPost().getComments().remove(post.getID());
346
347         }
348         else if(endorsementPosts.containsKey(id)){
349             EndorsementPost post=endorsementPosts.remove(id);
350
351             //remove the post from its owner account
352             post.getAccount().getEndorsementPosts().remove(post.getID());
353
354             //finally remove the reference from its parentPost
355             post.getParentPost().getEndorsementPosts().remove(post.getID());
356         }
357
358
359     }
360
361     @Override
362     public String showIndividualPost(int id) throws PostIDNotRecognisedException {
363         if(!(posts.containsKey(id) || comments.containsKey(id) || endorsementPosts.
    containsKey(id))){
364             throw new PostIDNotRecognisedException();
365         }
366
367         if(!endorsementPosts.containsKey(id)){
368             Post post_fake;
369             if(posts.containsKey(id))post_fake=posts.get(id);
370             else post_fake=comments.get(id);
371
372             ArrayList<Integer> commentsID_fake=new ArrayList<>(post_fake.getComments());
373             commentsID_fake.sort(Integer::compareTo);
374
375             StringBuilder fake=new StringBuilder();
376             if(commentsID_fake.size()>0)fake.append("|\n");
377             for(Integer cmID_Fake:commentsID_fake){
378                 buildPostTree(comments.get(cmID_Fake),fake,1);
379             }
380         }
381
382         String msg = "ID: "+
383                 id +
384                 "\n";
385
386         if (posts.containsKey(id)) {
387             Post post = posts.get(id);
388
389             msg += "Account: " +
390                     post.getAccount().getHandle() +
391                     "\n" +
392                     "No. endorsements: " +
393                     post.getEndorsementPosts().size() +
394                     " | No. comments: " +
395                     (post.getComments().size() + childComments) +
396                     "\n" +
397                     post.getMessage() +
```

```
398                     "\n";
399             childComments=0;
400         }
401         else if (comments.containsKey(id)) {
402             Comment post = comments.get(id);
403             msg += "Account: " +
404                     post.getAccount().getHandle() +
405                     "\n" +
406                     "No. endorsements: " +
407                     post.getEndorsementPosts().size() +
408                     " | No. comments: " +
409                     post.getComments().size() +
410                     "\n" +
411                     post.getMessage() +
412                     "\n";
413         }
414         else if(endorsementPosts.containsKey(id)) {
415             EndorsementPost post = endorsementPosts.get(id);
416             msg += "Account: " +
417                     post.getAccount().getHandle() +
418                     "\n" +
419                     "No. endorsements: 0 | No. comments: 0\n" +
420                     post.getMessage() +
421                     "\n";
422         }
423         return  msg;
424     }
425
426     @Override
427     public StringBuilder showPostChildrenDetails(int id) throws
    PostIDNotRecognisedException, NotActionablePostException {
428
429         // checking if the post is available as post or comments or endorsementPosts or
    not
430         if(!(posts.containsKey(id) || comments.containsKey(id) || endorsementPosts.
    containsKey(id))){
431             throw new PostIDNotRecognisedException();
432         }
433         if(endorsementPosts.containsKey(id)){
434             throw  new NotActionablePostException();
435         }
436
437         StringBuilder stringBuilder=new StringBuilder();
438         Post post;
439         if(posts.containsKey(id))post=posts.get(id);
440         else post=comments.get(id);
441
442         stringBuilder.append(showIndividualPost(id));
443
444         ArrayList<Integer> commentsID=new ArrayList<>(post.getComments());
445         commentsID.sort(Integer::compareTo);
446
447         if(commentsID.size()>0)stringBuilder.append("|\n");
448         for(Integer cmID:commentsID){
449             buildPostTree(comments.get(cmID),stringBuilder,1);
450         }
451         childComments=0;
452         return stringBuilder;
453
454     }
455
456     private void buildPostTree(Comment post,StringBuilder stringBuilder,int tabSize){
457
458         StringBuilder initialLine= new StringBuilder();
459         StringBuilder tabs= new StringBuilder();
460
461         for (int i = 0; i <tabSize-1 ; i++) {
462             initialLine.append('\t');
463         }
464         initialLine.append("| > ");
465
```

```
466         for (int i = 0; i <tabSize ; i++) {
467             tabs.append('\t');
468         }
469 
470 
471         String msg=initialLine+"ID: "+post.getID()+"\n"+
472                 tabs+"Account: " + post.getAccount().getHandle() + "\n" +
473                 tabs+"No. endorsements: " + post.getEndorsementPosts().size() + " | No.
    comments: " + post.getComments().size() + "\n" +
474                 tabs+post.getMessage() + "\n";
475 
476         stringBuilder.append(msg);
477 
478 
479         ArrayList<Integer> commentsID=new ArrayList<>(post.getComments());
480         commentsID.sort(Integer::compareTo);
481         if(commentsID.size()>0) stringBuilder.append(tabs).append("|\n");
482         for(Integer cmID:commentsID){
483             childComments++;
484             buildPostTree(comments.get(cmID),stringBuilder,tabSize+1);
485         }
486 
487     }
488 
489     @Override
490     public int getMostEndorsedPost() {
491         HashMap<Integer,Integer> frequency = new HashMap<>();
492 
493         // traversing through all endorsed posts and counting their parent id's
    frequency
494         for(Map.Entry<Integer,EndorsementPost> entry:endorsementPosts.entrySet()){
495             Integer parentPostID=entry.getValue().getParentPost().getID();
496             int cnt = frequency.getOrDefault(parentPostID,0);
497             frequency.put(parentPostID,cnt+1);
498         }
499         // traversing through the frequency map
500         Integer mostEndorsedCount=-1;
501         Integer ID=0;
502         for (Map.Entry<Integer,Integer> entry : frequency.entrySet()){
503             //
504             if(entry.getValue().compareTo(mostEndorsedCount)>0){
505                 mostEndorsedCount = entry.getValue();
506                 ID = entry.getKey();
507             }
508         }
509 
510         return ID;
511     }
512 
513     @Override
514     public int getMostEndorsedAccount() {
515         HashMap<Integer,Integer>map = new HashMap<>();
516 
517         for(Map.Entry<Integer,EndorsementPost> entry : endorsementPosts.entrySet()) {
518             Integer parentPostAccountID = entry.getValue().getParentPost().getAccount().
    getID();
519             int cnt = map.getOrDefault(parentPostAccountID,0);
520             map.put(parentPostAccountID,cnt+1);
521         }
522 
523         Integer mostEndorsedCount=-1;
524         Integer ID=0;
525         for (Map.Entry<Integer,Integer> entry:map.entrySet()){
526 
527             if(entry.getValue().compareTo(mostEndorsedCount)>0){
528                 mostEndorsedCount = entry.getValue();
529                 ID = entry.getKey();
530             }
531         }
532 
533         return ID;
```

```
534
535     }
536
537     @Override
538     public void erasePlatform() {
539
540         accountsByHandle.clear();
541         accountsByID.clear();
542         postIdTracker=accountIDTracker=0;
543         endorsementPosts.clear();
544         posts.clear();
545         comments.clear();
546         //add a generic empty post
547         emptyPost=new Post(-1,null,"The original content was removed" +
548                 " from the system and is no longer available.");
549
550     }
551
552     @Override
553     public void savePlatform(String filename) throws IOException {
554         FileOutputStream fos = new FileOutputStream(filename);
555         ObjectOutputStream out = new ObjectOutputStream(fos);
556
557         out.writeObject(this);
558
559         out.close();
560     }
561
562     @Override
563     public void loadPlatform(String filename) throws IOException, ClassNotFoundException
    {
564         FileInputStream fis = new FileInputStream(filename);
565         ObjectInputStream in = new ObjectInputStream(fis);
566         SocialMedia md = (SocialMedia)in.readObject();
567
568         accountsByHandle = md.accountsByHandle;
569         accountsByID = md.accountsByID;
570         postIdTracker = md.postIdTracker;
571         accountIDTracker = md.accountIDTracker;
572         endorsementPosts = md.endorsementPosts;
573         posts = md.posts;
574         comments = md.comments;
575         //add a generic empty post
576         emptyPost = md.emptyPost;
577     }
578 }
579
```

File - C:\Users\haris\IdeaProjects\Year1\ECM1410\CA3\src\socialmedia\Account.java

```java
package socialmedia;

import java.io.Serializable;
import java.util.HashSet;

/**
 * Represents an account
 */
public class Account implements Serializable {
    //Account handle and description
    private String handle;
    private String description;
    private int ID;
    // posts and comments owned by the account
    private HashSet<Integer> posts;
    private HashSet<Integer> comments;
    private HashSet<Integer> endorsementPosts;

    /**
     * Account constructor
     * @param handle the handle of the account
     * @param description the description of the account
     * @param ID the id of the account
     */
    public Account(String handle, String description, int ID) {
        this.handle = handle;
        this.description = description;
        this.ID = ID;
        posts = new HashSet<>();
        comments = new HashSet<>();
        endorsementPosts = new HashSet<>();
    }

    //getters and setters
    public HashSet<Integer> getPosts() {
        return posts;
    }

    public void setPosts(HashSet<Integer> posts) {
        this.posts = posts;
    }

    public HashSet<Integer> getComments() {
        return comments;
    }

    public void setComments(HashSet<Integer> comments) {
        this.comments = comments;
    }

    public HashSet<Integer> getEndorsementPosts() {
        return endorsementPosts;
    }

    public void setEndorsementPosts(HashSet<Integer> endorsementPosts) {
        this.endorsementPosts = endorsementPosts;
    }

    public int getID() {
        return ID;
    }

    public void setID(int ID) {
        this.ID = ID;
    }

    public String getHandle() {
        return handle;
    }

    public void setHandle(String handle) {
```

```
72          this.handle = handle;
73      }
74
75      public String getDescription() {
76          return description;
77      }
78
79      public void setDescription(String description) {
80          this.description = description;
81      }
82  }
83
```

File - C:\Users\haris\IdeaProjects\Year1\ECM1410\CA3\src\socialmedia\Post.java

```java
package socialmedia;

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

/**
 * Represents a post
 */
public class Post implements Serializable {
    //ID of the post
    protected int ID;
    //account that this post belongs to
    protected Account account;
    //post's message
    protected String message;

    //stores ID of the comments of this post
    protected HashSet<Integer> comments;
    //stores ID of the posts that endorse this post
    protected HashSet<Integer> endorsementPosts;

    /**
     * Post constructor
     * @param ID the id of the post
     * @param account the account posting
     * @param message the message of the post
     */
    public Post(int ID, Account account, String message) {
        this.ID = ID;
        this.account = account;
        this.message = message;
        comments=new HashSet<>();
        endorsementPosts=new HashSet<>();
    }

    //getters and setters
    public int getID() {
        return ID;
    }

    public void setID(int ID) {
        this.ID = ID;
    }

    public Account getAccount() {
        return account;
    }

    public void setAccount(Account account) {
        this.account = account;
    }

    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }

    public HashSet<Integer> getComments() {
        return comments;
    }

    public void setComments(HashSet<Integer> comments) {
        this.comments = comments;
    }

    public HashSet<Integer> getEndorsementPosts() {
        return endorsementPosts;
```

```
72      }
73
74      public void setEndorsementPosts(HashSet<Integer> endorsementPosts) {
75          this.endorsementPosts = endorsementPosts;
76      }
77  }
78
```

```java
package socialmedia;

import java.util.Objects;

/**
 * Represents a comment
 */
public class Comment extends Post {

    private Post parentPost;

    /**
     * Comment constructor
     * @param ID the id of the comment
     * @param account the account commenting
     * @param message the message being commented
     * @param parentPost the post that is being commented on
     */
    public Comment(int ID, Account account, String message,Post parentPost) {
        super(ID, account, message);
        this.parentPost=parentPost;
    }

    public Post getParentPost() {
        return parentPost;
    }

    public void setParentPost(Post parentPost) {
        this.parentPost = parentPost;
    }

    // for using hashset we need to provide a method so that
    // two object can be compared
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Comment)) return false;
        Comment comment = (Comment) o;
        return getID() == comment.getID();
    }
    // also used for comparison in hashset
    @Override
    public int hashCode() {
        return Objects.hash(getID());
    }
}

```

```java
package socialmedia;

import java.io.Serializable;
import java.util.Objects;

/**
 * Represents an endorsement
 */
public class EndorsementPost implements Serializable {
    private Post parentPost;
    private int ID;
    private Account account;
    private String message;

    /**
     * Endorsement post constructor
     * @param parentPost the post to be endorsed
     * @param ID the id of the endorsement post
     * @param account the account endorsing the post
     */
    public EndorsementPost(Post parentPost, int ID, Account account) {
        this.parentPost= parentPost;
        this.ID = ID;
        this.account = account;
    }

    // getters and setters
    public Post getParentPost() {
        return parentPost;
    }

    public void setParentPost(Post parentPost) {
        this.parentPost = parentPost;

    }

    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }

    public int getID() {
        return ID;
    }

    public void setID(int ID) {
        this.ID = ID;
    }

    public Account getAccount() {
        return account;
    }

    public void setAccount(Account account) {
        this.account = account;
    }

    // for using hashset we need to provide a method so that
    // two object can be compared
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof EndorsementPost)) return false;
        EndorsementPost post = (EndorsementPost) o;
        return ID == post.ID;
    }

    @Override
```

File - C:\Users\haris\IdeaProjects\Year1\ECM1410\CA3\src\socialmedia\EndorsementPost.java

```
72      public int hashCode() {
73          return Objects.hash(ID);
74      }
75
76      public void createMessage() {
77          message = "EP@" +
78                  parentPost.getAccount().getHandle() +
79                  ": " +
80                  parentPost.getMessage();
81      }
82  }
83
```