```java
package socialmedia;

/**
 * Thrown when attempting to use an account ID that does not exit in the system.
 *
 * @author Diogo Pacheco
 * @version 1.0
 *
 */
public class AccountIDNotRecognisedException extends Exception {

  /**
   * Constructs an instance of the exception with no message
   */
  public AccountIDNotRecognisedException() {
    // do nothing
  }

  /**
   * Constructs an instance of the exception containing the message argument
   *
   * @param message message containing details regarding the exception cause
   */
  public AccountIDNotRecognisedException(String message) {
    super(message);
  }

}
package socialmedia;

/**
 * Thrown when attempting to use an account handle that does not exit in the
 * system.
 *
 * @author Diogo Pacheco
 * @version 1.0
 *
 */
public class HandleNotRecognisedException extends Exception {

  /**
   * Constructs an instance of the exception with no message
   */
  public HandleNotRecognisedException() {
    // do nothing
  }

  /**
   * Constructs an instance of the exception containing the message argument
   *
   * @param message message containing details regarding the exception cause
   */
  public HandleNotRecognisedException(String message) {
    super(message);
  }

}
package socialmedia;

/**
```

```
 * Thrown when attempting to assign an account handle already in use in the
 * system.
 *
 * @author Diogo Pacheco
 * @version 1.0
 *
 */
public class IllegalHandleException extends Exception {

  /**
   * Constructs an instance of the exception with no message
   */
  public IllegalHandleException() {
    // do nothing
  }

  /**
   * Constructs an instance of the exception containing the message argument
   *
   * @param message message containing details regarding the exception cause
   */
  public IllegalHandleException(String message) {
    super(message);
  }

}
package socialmedia;

/**
 * Thrown when attempting to assign an account handle empty or having more than
 * the system limit of characters. A handle must be a single word, i.e., no
 * white spaces allowed.
 *
 * @author Diogo Pacheco
 * @version 1.0
 *
 */
public class InvalidHandleException extends Exception {

  /**
   * Constructs an instance of the exception with no message
   */
  public InvalidHandleException() {
    // do nothing
  }

  /**
   * Constructs an instance of the exception containing the message argument
   *
   * @param message message containing details regarding the exception cause
   */
  public InvalidHandleException(String message) {
    super(message);
  }

}
package socialmedia;

/**
 * Thrown when attempting to create a post which the message is empty or has
```

```
 * more characters than the system's limit.
 *
 * @author Diogo Pacheco
 * @version 1.0
 *
 */
public class InvalidPostException extends Exception {

  /**
   * Constructs an instance of the exception with no message
   */
  public InvalidPostException() {
    // do nothing
  }

  /**
   * Constructs an instance of the exception containing the message argument
   *
   * @param message message containing details regarding the exception cause
   */
  public InvalidPostException(String message) {
    super(message);
  }

}
package socialmedia;

/**
 * Thrown when attempting to act upon an not-actionable post.
 *
 * @author Diogo Pacheco
 * @version 1.0
 *
 */
public class NotActionablePostException extends Exception {

  /**
   * Constructs an instance of the exception with no message
   */
  public NotActionablePostException() {
    // do nothing
  }

  /**
   * Constructs an instance of the exception containing the message argument
   *
   * @param message message containing details regarding the exception cause
   */
  public NotActionablePostException(String message) {
    super(message);
  }

}
package socialmedia;

/**
 * Thrown when attempting to use a post ID that does not exit in the system.
 *
 * @author Diogo Pacheco
 * @version 1.0
```

```
 *
 */
public class PostIDNotRecognisedException extends Exception {

  /**
   * Constructs an instance of the exception with no message
   */
  public PostIDNotRecognisedException() {
    // do nothing
  }

  /**
   * Constructs an instance of the exception containing the message argument
   *
   * @param message message containing details regarding the exception cause
   */
  public PostIDNotRecognisedException(String message) {
    super(message);
  }

}
package socialmedia;

import java.io.Serializable;

/**
 * Account is a class that encapsulates user functionality for the
 * social media platform. Each instance of account has a unique id
 * and handle attributes, which is enforced within the social media
 * class's functionality.
 * Account extends serializable so that it can be serialised allowing
 * for the social media platform to be saved, loaded, etc.
 *
 * @author Graham Faiola
 * @author Robert Campbell
 */
public class Account implements Serializable {

    private int uID;

    private String handle;

    private String description;

    private int numEndorsements;

    private int numPosts;

    /**
     * Creates an instance of the Account class using passed Arguments
     *
     * @param uID ID of the user to create
     * @param handle hadnle of the user to create
     * @param description
     */
    public Account(int uID, String handle, String description){
        assert (description.length() < 100) : "Description is too long";

        this.uID = uID;
        this.handle = handle;
```

```
241         this.description = description;
242     }
243 
244     /**
245      * Creates blank instance of Account class
246      */
247     public Account(){}
248 
249     public int getUID() {
250         return this.uID;
251     }
252 
253     public void setUID(int uID) {
254         this.uID = uID;
255     }
256 
257     public String getHandle() {
258         return this.handle;
259     }
260 
261     public void setHandle(String handle) {
262         this.handle = handle;
263     }
264 
265     public String getDescription() {
266         return this.description;
267     }
268 
269     public void setDescription(String description) {
270         this.description = description;
271     }
272 
273     public int getNumEndorsements() {
274         return numEndorsements;
275     }
276 
277     public void addNumEndorsements() {
278         this.numEndorsements += 1;
279     }
280 
281     public void removeNumEndorsements() {
282         this.numEndorsements -= 1;
283     }
284     public void setNumEndorsements(int numEndorsements) {
285         this.numEndorsements = numEndorsements;
286     }
287 
288     /**
289      * Increments the number of posts assigned to this instance
290      * of Account
291      */
292     public void addNumPosts() {
293         this.numPosts += 1;
294     }
295 
296     /**
297      * Decrements the number of posts assigned to this instance
298      * of Account
299      */
300     public void removeNumPosts() {
```

```
        this.numPosts -= 1;
    }

    public int getNumPosts() {
        return this.numPosts;
    }

}
package socialmedia;

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

/**
 * Post is a class that encapsulates post functionality. An instance of the Post
 * class can be commented on, Endorsed. Both of which are available in the comments
 * and endorsements ArrayLists within the class.
 * The Post stores the ID of the Account that posted it to allow for the Account
 * records, e.g. numEndorsements, numCounts etc., can be kept consistent.
 *
 * Post also implements serializable which allows for the Platform to be saved and
 * loaded within the SocialMedia class.
 *
 * @author Graham Faiola
 * @author Robert Campbell
 */
public class Post implements Serializable {

    protected int id;

    protected int uID;

    protected String content;

    protected boolean exists;

    protected ArrayList<Comment> comments = new ArrayList<>();

    protected ArrayList<Endorsement> endorsements = new ArrayList<>();

    protected int numEndorsements;

    protected int numComments;

    protected int parentID;

    protected boolean isEnd = false;

    protected boolean isOrphan;

    protected Post genericEmptyPost;

    /**
     * Creates instance of the Post class using passed arguments.
     *
     * @param id the id of the post
     * @param uID the id of the account responsible for the post
     * @param content the content of Post
     */
    public Post(int id, int uID, String content) {
```

```
361         this.id = id;
362         this.uID = uID;
363         this.content = content;
364     }
365 
366     /**
367      * Creates blank instance of the Post class
368      */
369     public Post() {
370     }
371 
372     public Post(int id, String content) {
373         this.id = id;
374         this.content = content;
375     }
376 
377     public Post(String content) {
378         this.content = content;
379     }
380 
381     public void setid(int id) {
382         this.id = id;
383     }
384 
385     public void setuID(int uID) {
386         this.uID = uID;
387     }
388 
389     public void setContent(String content) {
390         this.content = content;
391     }
392 
393     public void setExists(boolean exists) {
394         this.exists = exists;
395     }
396 
397     public int uID() {
398         return this.uID;
399     }
400 
401     public int id() {
402         return this.id;
403     }
404 
405     public String content() {
406         return this.content;
407     }
408 
409     public boolean exists() {
410         return this.exists;
411     }
412 
413     public void delete() {
414         this.exists = false;
415     }
416 
417     /**
418      * Adds a comment to the ArrayList comments parameter of Post class
419      *
420      * @param newcomment Comment object to add to list
```

```
421      */
422     public void addcomment(Comment newcomment) {
423         this.comments.add(newcomment);
424         int incrementcomment = this.getNumComments() + 1;
425         this.setNumComments(incrementcomment);
426     }
427 
428     /**
429      * Removes a comment from the ArrayList comment parameter of Post class
430      *
431      * @param oldcomment Comment object to remove from list of Comments
432      */
433     public void removecomment(Comment oldcomment) {
434         this.comments.remove(oldcomment);
435         int incrementcomment = this.getNumComments() - 1;
436         this.setNumComments(incrementcomment);
437     }
438 
439     /**
440      * Adds a specified endorsement to the endorsements ArrayList of Post class
441      *
442      * @param newendorsement Endorsement object to append to the list of Endorsements
443      */
444     public void addendorsement(Endorsement newendorsement) {
445         this.endorsements.add(newendorsement);
446         int incrementendorsement = this.getNumEndorsements() + 1;
447         this.setNumEndorsements(incrementendorsement);
448     }
449 
450     /**
451      * Removes a specified endorsement to the endorsements ArrayList of Post class
452      *
453      * @param oldendorsement Endorsement object to remove from the list of
    Endorsements
454      */
455     public void removeendorsement(Endorsement oldendorsement) {
456         this.endorsements.remove(oldendorsement);
457         int incrementendorsement = this.getNumEndorsements() -1;
458         this.setNumEndorsements(incrementendorsement);
459     }
460 
461     public ArrayList<Comment> getComments() {
462         return this.comments;
463     }
464 
465     public void setComments(ArrayList<Comment> comments) {
466         this.comments = comments;
467     }
468 
469     public ArrayList<Endorsement> getEndorsements() {
470         return this.endorsements;
471     }
472 
473     public int getNumEndorsements() {
474         return this.numEndorsements;
475     }
476 
477     public void setNumEndorsements(int numEndorsements) {
478         this.numEndorsements = numEndorsements;
479     }
```

```
480
481     public int getNumComments() {
482         return this.numComments;
483     }
484
485     public void setNumComments(int numComments) {
486         this.numComments = numComments;
487     }
488
489
490     public int getParentID() {
491         return parentID;
492     }
493
494     public void setParentID(int parentID) {
495         this.parentID = parentID;
496     }
497
498     /**
499      * Checks if the Post object is an Endorsement. As this means the post
500      * would not be endorseable or commentable.
501      *
502      * @return if this post is an Endorsement
503      */
504     public boolean isEnd() {
505         return isEnd;
506     }
507
508     public void setEnd(boolean end) {
509         isEnd = end;
510     }
511
512     public boolean isOrphan() {
513         return isOrphan;
514     }
515
516     public void setOrphan(boolean orphan) {
517         isOrphan = orphan;
518     }
519
520     public Post getGenericEmptyPost() {
521         return genericEmptyPost;
522     }
523
524     public void setGenericEmptyPost(Post genericEmptyPost) {
525         this.genericEmptyPost = genericEmptyPost;
526     }
527 }
528 package socialmedia;
529
530 /**
531  * The Comment class extends the superclass Post, this allows for a distinction
532  * between comments, original posts and endorsements when programming using these
533  * classes.
534  *
535  * @author Graham Faiola
536  * @author Robert Campbell
537  */
538 public class Comment extends Post {
539
```

```
    /**
     * Creates an instance of the comment class
     */
    public Comment() {}

}
package socialmedia;

/**
 * The Endorsement class extends the superclass Post, this allows for a distinction
 * between endorsements, original posts and comments when programming using these
 * classes.
 *
 * @author Graham Faiola
 * @author Robert Campbell
 */
public class Endorsement extends Post{

    /**
     * Constructs an instance of the Endorsement class and uses the setEnd method
     * of it's superclass to define itself as an Endorsement
     */
    public Endorsement() {
        this.setEnd(true);
    }

}
package socialmedia;

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

/**
 * Platform is the serializable class that is used by the SocialMedia class
 * to save and load the social media platform.
 *
 * @author Graham Faiola
 * @author Robert Campbell
 */
public class Platform implements Serializable {

    private ArrayList<Account> accounts = new ArrayList<>();
    private ArrayList<Post> posts = new ArrayList<>();
    private int nextid = 0;
    private int nextuid = 0;
    private Post genericEmptyPost;

    /**
     * Creates blank instance of the Platform class
     */
    public Platform() {
    }


    public ArrayList<Account> getAccounts() {
        return accounts;
    }

    public void setAccounts(ArrayList<Account> accounts) {
        this.accounts = accounts;
```

```
600     }
601
602     public ArrayList<Post> getPosts() {
603         return posts;
604     }
605
606     public void setPosts(ArrayList<Post> posts) {
607         this.posts = posts;
608     }
609
610     public int getNextid() {
611         return nextid;
612     }
613
614     public void setNextid(int nextpid) {
615         this.nextid = nextpid;
616     }
617
618     public int getNextuid() {
619         return nextuid;
620     }
621
622     public void setNextuid(int nextuid) {
623         this.nextuid = nextuid;
624     }
625
626     public Post getGenericEmptyPost() {
627         return genericEmptyPost;
628     }
629
630     public void setGenericEmptyPost(Post genericEmptyPost) {
631         this.genericEmptyPost = genericEmptyPost;
632     }
633 }
634 package socialmedia;
635
636 import java.io.*;
637 import java.util.ArrayList;
638 import java.util.Scanner;
639
640 /**
641  * BadSocialMedia is a compiling, and fully-functioning implementor of
642  * the SocialMediaPlatform interface.
643  *
644  * @author Graham Faiola
645  * @author Robert Campbell
646  * @version 1.0
647  */
648 public class SocialMedia implements SocialMediaPlatform {
649
650     private ArrayList<Account> accounts = new ArrayList<>();
651     private ArrayList<Post> posts = new ArrayList<>();
652     private int nextID = 1;
653     private int nextuid = 0;
654     private String orphanMessage = "The original content was removed from the system
    and is no longer available.";
655     private Post genericEmptyPost = new Post(orphanMessage);
656
657     /**
658      * Returns the id of an account matching the specified handle parameter
```

```
659      * If no matching account can be found the function returns -1
660      *
661      * @param handle handle of user
662      * @return id of user matching handle parameter
663      */
664     public int finduid(String handle) {
665         int id = -1;
666         for (Account a : accounts) {
667             if (a != null && a.getHandle().equals(handle)) {
668                 id = a.getUID();
669                 break;
670             }
671         }
672         return id;
673     }
674 
675     /**
676      * Searches through the accounts ArrayList to find an account with ID
677      * matching the passed id argument. If a matching account cannot be
678      * found then null is returned.
679      *
680      * @param id id of the account to be searched for
681      * @return the account matching the id paramter
682      */
683     public Account findAccountFromID(int id) {
684         Account foundAccount = null;
685         for (Account user : accounts) {
686             if (user != null && user.getUID() == id) {
687                 foundAccount = user;
688                 break;
689             }
690         }
691         return foundAccount;
692     }
693 
694     /**
695      * Checks if the handle specified is already used in the accounts ArrayList.
696      * If it is already used then a new account cannot be created with this
697      * handle.
698      *
699      * @param handle
700      * @return true if handle is already used, otherwise false
701      */
702     public boolean uniqueHandle(String handle) {
703         boolean unique = true;
704         for (Account user : accounts) {
705             if (user.getHandle() == handle) {
706                 unique = false;
707                 break;
708             }
709         }
710         return unique;
711     }
712 
713     /**
714      * Called when a comments parent post is deleted and the comment is made
715      * an orphan. The comment's parent id is set to null
716      *
717      * @param thisComment the comment object that's parent has been deleted
718      */
```

```
719     public void orphanComment(Comment thisComment) {
720         for (Comment comment : thisComment.getComments()) {
721             comment.setOrphan(true);
722             comment.setGenericEmptyPost(genericEmptyPost);
723             comment.setParentID(0);
724         }
725     }
726 
727     /**
728      * Deletes an endorsement matching the specified id.
729      * This function makes sure the account of the post that was endorsed
730      * also has the number of recieved endorsements decremented.
731      *
732      * @param id the id of the endorsement to delete.
733      */
734     public void deleteEndFromID(int id) {
735         Endorsement thisEnd = (Endorsement) findFromID(id);
736         posts.remove(thisEnd);
737         for (Post post : posts) {
738             if (post.id() == thisEnd.getParentID()) {
739                 post.removeendorsement(thisEnd);
740                 findAccountFromID(post.uID()).removeNumEndorsements();
741             }
742         }
743     }
744 
745     /**
746      * Finds a post from a specified id within the posts ArrayList. If one
747      * is not found, then null is returned
748      *
749      * @param id the id of the post to be searched for
750      * @return the Post object, or null if it cannot be found
751      */
752     public Post findFromID(int id) {
753         for (Post post : posts) {
754             if (post.id() == id) {
755                 return post;
756             }
757         }
758         return null;
759     }
760 
761     @Override
762     public int createAccount(String handle) throws IllegalHandleException,
    InvalidHandleException {
763         if (handle.length() == 0 || handle.length() > 30 || handle.contains(" ")) {
764             throw new InvalidHandleException("Handle does not match criteria");
765         } else if (!uniqueHandle(handle)) {
766             throw new IllegalHandleException("Handle is already in use");
767         } else {
768             Account newAccount = new Account();
769             newAccount.setUID(nextuid);
770             newAccount.setHandle(handle);
771 
772             nextuid += 1;
773             accounts.add(newAccount);
774             return newAccount.getUID();
775         }
776     }
777 
```

```
778     @Override
779     public int createAccount(String handle, String description) throws
    IllegalHandleException, InvalidHandleException {
780         // TODO Auto-generated method stub
781         if (handle.length() == 0 || handle.length() > 30 || handle.contains(" ")) {
782             throw new InvalidHandleException("Handle does not match criteria");
783         } else if (!uniqueHandle(handle)) {
784             throw new IllegalHandleException("Handle is already in use");
785         } else {
786             Account newAccount = new Account();
787             newAccount.setUID(nextuid);
788             newAccount.setHandle(handle);
789             newAccount.setDescription(description);
790 
791             nextuid += 1;
792             accounts.add(newAccount);
793             return newAccount.getUID();
794         }
795     }
796 
797     @Override
798     public void removeAccount(int id) throws AccountIDNotRecognisedException {
799         // TODO Auto-generated method stub
800         Account oldAccount = null;
801         for (Account user : accounts) {
802             if (user != null && user.getUID() == id) {
803                 oldAccount = user;
804                 break;
805             }
806         }
807         if (oldAccount == null) {
808             throw new AccountIDNotRecognisedException("Account ID not recognised");
809         } else {
810             accounts.remove(oldAccount);
811             for (Post post : posts) {
812                 if (id == post.uID()) {
813                     try {
814                         deletePost(post.id());
815                     } catch (PostIDNotRecognisedException e) {
816                         e.printStackTrace();
817                     }
818                 }
819             }
820         }
821     }
822 
823     @Override
824     public void removeAccount(String handle) throws HandleNotRecognisedException {
825         // TODO Auto-generated method stub
826         int uid = finduid(handle);
827         if (uid == -1) {
828             throw new HandleNotRecognisedException("Handle not recognised");
829             // Raises HandleNotRecognisedException
830         } else {
831             Account oldAccount = null;
832             for (Account user : accounts) {
833                 if (user != null && user.getHandle() == handle) {
834                     oldAccount = user;
835                     break;
836                 }
```

```
837             }
838             accounts.remove(oldAccount);
839         }
840     }
841 
842     @Override
843     public void changeAccountHandle(String oldHandle, String newHandle)
844             throws HandleNotRecognisedException, IllegalHandleException,
    InvalidHandleException {
845         int uid = finduid(oldHandle);
846         if (uid == -1) {
847             throw new HandleNotRecognisedException("Handle not recognised");
848         } else if (newHandle.length() == 0 || newHandle.length() > 30 ||
    newHandle.contains(" ")) {
849             throw new InvalidHandleException("Handle does not match criteria");
850         } else if (!uniqueHandle(newHandle)) {
851             throw new IllegalHandleException("Handle is already in use");
852         } else {
853             Account userToAlter = findAccountFromID(uid);
854             userToAlter.setHandle(newHandle);
855         }
856     }
857 
858     @Override
859     public void updateAccountDescription(String handle, String description) throws
    HandleNotRecognisedException {
860         int uid = finduid(handle);
861         if (uid == -1) {
862             throw new HandleNotRecognisedException("Handle not recognised");
863         } else {
864             Account userToAlter = findAccountFromID(uid);
865             userToAlter.setDescription(description);
866         }
867     }
868 
869     @Override
870     public String showAccount(String handle) throws HandleNotRecognisedException {
871         // TODO Auto-generated method stub
872         int thisUID = finduid(handle);
873         if (thisUID == -1) {
874             throw new HandleNotRecognisedException("Handle not recognised");
875         } else {
876             Account thisaccount = new Account();
877             for (Account account : accounts) {
878                 if (thisUID == account.getUID()) {
879                     thisaccount = account;
880                     break;
881                 }
882             }
883             String stringFormat = "<pre>\nID: " + thisaccount.getUID() + "\nHandle: "
    + thisaccount.getHandle()
884                     + "\nDescription: " + thisaccount.getDescription() + "\nPost
    counts: " + thisaccount.getNumPosts()
885                     + "\nEndorse count: " + thisaccount.getNumEndorsements() +
    "\n</pre>";
886             return stringFormat;
887         }
888     }
889 
890     @Override
```

```
891     public int createPost(String handle, String message) throws
    HandleNotRecognisedException, InvalidPostException {
892         // maybe add error msg and repeat later if needed
893         int newpostuid = finduid(handle);
894         if (message.length() > 100) {
895             throw new InvalidPostException("This post is more than 100 characters
    long");
896         } else if (newpostuid == -1) {
897             throw new HandleNotRecognisedException("Handle not recognised");
898         } else {
899             Post newpost = new Post();
900             int newpostid = nextID;
901             nextID += 1;
902             newpost.setid(newpostid);
903             newpost.setuID(newpostuid);
904             findAccountFromID(newpostuid).addNumPosts();
905             newpost.setContent(message);
906             posts.add(newpost);
907             return newpostid;
908         }
909     }
910
911     @Override
912     public int endorsePost(String handle, int id)
913             throws HandleNotRecognisedException, PostIDNotRecognisedException,
    NotActionablePostException {
914         int newpostuid = finduid(handle);
915         Post post = findFromID(id);
916         Account endorsedAccount = findAccountFromID(post.uID());
917
918         if (newpostuid == -1) {
919             throw new HandleNotRecognisedException("Handle is not recognised");
920         } else if (post == null) {
921             throw new PostIDNotRecognisedException("Post ID is not recognised");
922         } else if (post.isEnd()) {
923             throw new NotActionablePostException("This post is an endorsement, not a
    comment or original post");
924         } else {
925             int newID = nextID;
926             nextID += 1;
927
928             Endorsement newEnd = new Endorsement();
929             newEnd.setid(newID);
930             newEnd.setParentID(id);
931             newEnd.setuID(newpostuid);
932             newEnd.setEnd(true);
933             newEnd.setContent(post.content());
934             post.addendorsement(newEnd);
935             posts.add(newEnd);
936
937             endorsedAccount.addNumEndorsements();
938             return newID;
939         }
940     }
941
942     @Override
943     public int commentPost(String handle, int id, String message) throws
    HandleNotRecognisedException,
944             PostIDNotRecognisedException, NotActionablePostException,
    InvalidPostException {
```

```
945         // TODO Auto-generated method stub
946         // should add comment to list within post object, as well as total post list.
947         int newpostuid = finduid(handle);
948         Post post = findFromID(id);
949         if (newpostuid == -1) {
950             throw new HandleNotRecognisedException("Handle is not recognised");
951         } else if (post == null) {
952             throw new PostIDNotRecognisedException("Post ID is not recognised");
953         } else if (post.isEnd()) {
954             throw new NotActionablePostException("This post cannot be commented on");
955         } else if (message.length() > 100) {
956             throw new InvalidPostException("Message length is greater than 100
    characters");
957         } else {
958             int newID = nextID;
959             nextID += 1;
960 
961             Comment newcomment = new Comment();
962             newcomment.setParentID(id);
963             newcomment.setid(newID);
964             newcomment.setContent(message);
965             newcomment.setuID(newpostuid);
966             post.addcomment(newcomment);
967             posts.add(newcomment);
968 
969             findAccountFromID(newpostuid).addNumPosts();
970             return newID;
971         }
972     }
973 
974     @Override
975     public void deletePost(int id) throws PostIDNotRecognisedException {
976         // TODO Auto-generated method stub
977         Post thispost = findFromID(id);
978         if (thispost == null) {
979             throw new PostIDNotRecognisedException("Post ID is not recognised");
980         } else {
981             if (!thispost.isEnd()) {
982                 for (Endorsement end : thispost.getEndorsements()) {
983                     deleteEndFromID(end.id());
984                 }
985                 posts.remove(thispost);
986                 findAccountFromID(thispost.uID()).removeNumPosts();
987                 orphanComment((Comment) thispost);
988                 for (Post post : posts) {
989                     if (post.id() == thispost.getParentID() && thispost.getParentID()
    != 0) {
990                         post.removecomment((Comment) thispost);
991                     }
992                 }
993             } else if (thispost.isEnd) {
994                 deleteEndFromID(thispost.id());
995             }
996         }
997     }
998 
999     @Override
1000     public String showIndividualPost(int id) throws PostIDNotRecognisedException {
1001         // TODO Auto-generated method stub
1002         Post indivPost = findFromID(id);
```

```
1003        if (indivPost == null) {
1004            throw new PostIDNotRecognisedException("Post ID is not recognised");
1005        } else {
1006            Account postAccount = findAccountFromID(indivPost.uID());
1007            String PostDetails = "\n<pre>\nID: " + indivPost.id() + "\nAccount: " +
    postAccount.getHandle()
1008                    + "\nNo. endorsements: " + indivPost.getNumEndorsements() + " |
    No. comments: "
1009                    + indivPost.getNumComments() + "\n" + indivPost.content() +
    "\n</pre>";
1010            return PostDetails;
1011        }
1012    }
1013
1014    @Override
1015    public StringBuilder showPostChildrenDetails(int id)
1016            throws PostIDNotRecognisedException, NotActionablePostException {
1017        // TODO Auto-generated method stub
1018        Post parentPost = findFromID(id);
1019        if (parentPost == null) {
1020            throw new PostIDNotRecognisedException("Post ID is not recognised");
1021        } else if (parentPost.isEnd()) {
1022            throw new NotActionablePostException("This post is an endorsement, not a
    comment or original post");
1023        } else {
1024            String parentPostDetails = showIndividualPost(id).replaceAll("\n</pre>$",
    "");
1025            StringBuilder postChildrenDetails = new StringBuilder();
1026            postChildrenDetails.append(parentPostDetails);
1027            if (parentPost.getNumComments() != 0) {
1028                String strPostChildren = recursivePostChildren(1,
    parentPost.getComments()).toString();
1029                postChildrenDetails.append(strPostChildren);
1030            }
1031            postChildrenDetails.append("\n</pre>\n");
1032            return postChildrenDetails;
1033        }
1034    }
1035
1036    public StringBuilder recursivePostChildren(int indentSize, ArrayList<Comment>
    postChildren)
1037            throws PostIDNotRecognisedException {
1038        StringBuilder indentString = new StringBuilder("\n");
1039        int indentIncrement = 0;
1040        while (indentIncrement < indentSize) {
1041            indentIncrement++;
1042            indentString.append("\t");
1043        }
1044        String strIndents = indentString.toString();
1045        String strFirstLineFormat = strIndents.replaceAll("\t$", "|");
1046        String strSecondLineFormat = strIndents.replaceAll("\t$", "| > ");
1047        String strFirstLinesFormat = strFirstLineFormat + strSecondLineFormat;
1048        StringBuilder postChildrenDetails = new StringBuilder();
1049        for (Comment comment : postChildren) {
1050            String parentPostDetails =
    showIndividualPost(comment.id()).replaceAll("\n</pre>$", "");
1051            parentPostDetails = parentPostDetails.replaceFirst("\n<pre>\n",
    strFirstLinesFormat);
1052            Scanner linescanner = new Scanner(parentPostDetails);
1053            int scannerlinetracker = 0;
```

```
1054            while (linescanner.hasNextLine()) {
1055                String thisline = linescanner.nextLine();
1056                if (scannerlinetracker <= 1) {
1057                    scannerlinetracker += 1;
1058                    linescanner.close();
1059                    continue;
1060                }
1061                scannerlinetracker += 1;
1062                String indentedthisline = strIndents.replaceAll("\n", "") + thisline;
1063                parentPostDetails = parentPostDetails.replaceFirst(thisline,
    indentedthisline);
1064            }
1065            postChildrenDetails.append(parentPostDetails);
1066            if (comment.getNumComments() != 0) {
1067                ArrayList<Comment> thisCommentChildren = comment.getComments();
1068                int newindentSize = indentSize + 1;
1069                postChildrenDetails.append(recursivePostChildren(newindentSize,
    thisCommentChildren));
1070            }
1071        }
1072        return postChildrenDetails;
1073    }
1074
1075    @Override
1076    public int getNumberOfAccounts() {
1077        // TODO Auto-generated method stub
1078        int accountNo = 0;
1079        for (Account user : accounts) {
1080            if (user != null) {
1081                accountNo += 1;
1082            }
1083        }
1084        return accountNo;
1085    }
1086
1087    @Override
1088    public int getTotalOriginalPosts() {
1089        // TODO Auto-generated method stub
1090        int postNo = 0;
1091        for (Post post : posts) {
1092            if (post.getParentID() == 0 && !post.isOrphan()) {
1093                postNo += 1;
1094            }
1095        }
1096        return postNo;
1097    }
1098
1099    @Override
1100    public int getTotalEndorsmentPosts() {
1101        // TODO Auto-generated method stub
1102        int endorsementNo = 0;
1103
1104        for (Account account : accounts) {
1105            endorsementNo += account.getNumEndorsements();
1106        }
1107        return endorsementNo;
1108    }
1109
1110    @Override
1111    public int getTotalCommentPosts() {
```

```
1112        // TODO Auto-generated method stub
1113        int commentNo = 0;
1114
1115        for (Post post : posts) {
1116            commentNo += post.getNumComments();
1117            if (post.isOrphan()) {
1118                commentNo += 1;
1119            }
1120        }
1121        return commentNo;
1122    }
1123
1124    @Override
1125    public int getMostEndorsedPost() {
1126        int highestEnd = 0;
1127        Post mostEnd = new Post();
1128        for (Post post : posts) {
1129            if (post.getNumEndorsements() > highestEnd) {
1130                mostEnd = post;
1131                highestEnd = post.getNumEndorsements();
1132            }
1133        }
1134        return mostEnd.id();
1135        // TODO Auto-generated method stub
1136    }
1137
1138    @Override
1139    public int getMostEndorsedAccount() {
1140        // TODO Auto-generated method stub
1141        int highestEnd = 0;
1142        Account mostEnd = new Account();
1143        for (Account account : accounts) {
1144            if (account.getNumEndorsements() > highestEnd) {
1145                mostEnd = account;
1146                highestEnd = account.getNumEndorsements();
1147            }
1148        }
1149        return mostEnd.getUID();
1150    }
1151
1152    @Override
1153    public void erasePlatform() {
1154        // TODO Auto-generated method stub
1155        posts = new ArrayList<>();
1156        accounts = new ArrayList<>();
1157        nextuid = 0;
1158        nextID = 0;
1159
1160    }
1161
1162    @Override
1163    public void savePlatform(String filename) throws IOException {
1164        // TODO Auto-generated method stub
1165        Platform platform = new Platform();
1166        platform.setAccounts(accounts);
1167        platform.setPosts(posts);
1168        platform.setNextid(nextID);
1169        platform.setNextuid(nextuid);
1170        platform.setGenericEmptyPost(genericEmptyPost);
1171        // creating file for the platform
```

```
1172         String platformfilename = filename + ".ser";
1173         FileOutputStream storeplatform = new FileOutputStream(platformfilename);
1174         // creating the file writer
1175         ObjectOutputStream serialiseObject = new ObjectOutputStream(storeplatform);
1176         // writing to file to store the platform
1177         serialiseObject.writeObject(platform);
1178         // close file
1179         serialiseObject.close();
1180         storeplatform.close();
1181     }
1182 
1183     @Override
1184     public void loadPlatform(String filename) throws IOException,
     ClassNotFoundException {
1185         // TODO Auto-generated method stub
1186         // accessing the file
1187         String filenameser = filename + ".ser";
1188         FileInputStream getPlatform = new FileInputStream(filenameser);
1189         // creating file reader
1190         ObjectInputStream readPlatform = new ObjectInputStream(getPlatform);
1191         // reading object from file
1192         Platform loadedPlatform = (Platform) readPlatform.readObject();
1193         posts.addAll(loadedPlatform.getPosts());
1194         accounts.addAll(loadedPlatform.getAccounts());
1195         nextID = loadedPlatform.getNextid();
1196         nextuid = loadedPlatform.getNextuid();
1197         genericEmptyPost = loadedPlatform.getGenericEmptyPost();
1198     }
1199 
1200 }
1201 package socialmedia;
1202 
1203 import java.io.IOException;
1204 import java.io.Serializable;
1205 
1206 /**
1207  * MiniSocialMediaPlatform interface. The no-argument constructor of a class
1208  * implementing this interface should initialise the MiniSocialMediaPlatform as
1209  * an empty platform with no initial accounts nor posts within it. For Solo
1210  * submissions ONLY.
1211  *
1212  * @author Diogo Pacheco
1213  * @version 1.0
1214  *
1215  */
1216 public interface MiniSocialMediaPlatform extends Serializable {
1217 
1218   // Account-related methods ************************************
1219 
1220   /**
1221    * The method creates an account in the platform with the given handle.
1222    * <p>
1223    * The state of this SocialMediaPlatform must be be unchanged if any exceptions
1224    * are thrown.
1225    *
1226    * @param handle account's handle.
1227    * @throws IllegalHandleException if the handle already exists in the platform.
1228    * @throws InvalidHandleException if the new handle is empty, has more than 30
1229    *                                characters, or has white spaces.
1230    * @return the ID of the created account.
```

```
1231    *
1232    */
1233   int createAccount(String handle) throws IllegalHandleException,
     InvalidHandleException;
1234
1235   /**
1236    * The method removes the account with the corresponding ID from the platform.
1237    * When an account is removed, all of their posts and likes should also be
1238    * removed.
1239    * <p>
1240    * The state of this SocialMediaPlatform must be be unchanged if any exceptions
1241    * are thrown.
1242    *
1243    * @param id ID of the account.
1244    * @throws AccountIDNotRecognisedException if the ID does not match to any
1245    *                                         account in the system.
1246    */
1247   void removeAccount(int id) throws AccountIDNotRecognisedException;
1248
1249   /**
1250    * The method replaces the oldHandle of an account by the newHandle.
1251    * <p>
1252    * The state of this SocialMediaPlatform must be be unchanged if any exceptions
1253    * are thrown.
1254    *
1255    * @param oldHandle account's old handle.
1256    * @param newHandle account's new handle.
1257    * @throws HandleNotRecognisedException if the old handle does not match to any
1258    *                                      account in the system.
1259    * @throws IllegalHandleException       if the new handle already exists in the
1260    *                                      platform.
1261    * @throws InvalidHandleException       if the new handle is empty, has more
1262    *                                      than 30 characters, or has white spaces.
1263    */
1264   void changeAccountHandle(String oldHandle, String newHandle)
1265       throws HandleNotRecognisedException, IllegalHandleException,
     InvalidHandleException;
1266
1267   /**
1268    * The method creates a formatted string summarising the stats of the account
1269    * identified by the given handle. The template should be:
1270    *
1271    * <pre>
1272    * ID: [account ID]
1273    * Handle: [account handle]
1274    * Description: [account description]
1275    * Post count: [total number of posts, including endorsements and replies]
1276    * Endorse count: [sum of endorsements received by each post of this account]
1277    * </pre>
1278    *
1279    * @param handle handle to identify the account.
1280    * @return the account formatted summary.
1281    * @throws HandleNotRecognisedException if the handle does not match to any
1282    *                                      account in the system.
1283    */
1284   String showAccount(String handle) throws HandleNotRecognisedException;
1285
1286   // End Account-related methods ************************************
1287
1288   // Post-related methods ************************************
```

```
1289
1290   /**
1291    * The method creates a post for the account identified by the given handle with
1292    * the following message.
1293    * <p>
1294    * The state of this SocialMediaPlatform must be be unchanged if any exceptions
1295    * are thrown.
1296    *
1297    * @param handle  handle to identify the account.
1298    * @param message post message.
1299    * @throws HandleNotRecognisedException if the handle does not match to any
1300    *                                      account in the system.
1301    * @throws InvalidPostException         if the message is empty or has more than
1302    *                                      100 characters.
1303    * @return the sequential ID of the created post.
1304    */
1305   int createPost(String handle, String message) throws HandleNotRecognisedException,
     InvalidPostException;
1306
1307   /**
1308    * The method creates an endorsement post of an existing post, similar to a
1309    * retweet on Twitter. An endorsement post is a special post. It contains a
1310    * reference to the endorsed post and its message is formatted as:
1311    * <p>
1312    * <code>"EP@" + [endorsed account handle] + ": " + [endorsed message]</code>
1313    * <p>
1314    * The state of this SocialMediaPlatform must be be unchanged if any exceptions
1315    * are thrown.
1316    *
1317    * @param handle of the account endorsing a post.
1318    * @param id     of the post being endorsed.
1319    * @return the sequential ID of the created post.
1320    * @throws HandleNotRecognisedException if the handle does not match to any
1321    *                                      account in the system.
1322    * @throws PostIDNotRecognisedException if the ID does not match to any post in
1323    *                                      the system.
1324    * @throws NotActionablePostException   if the ID refers to a endorsement post.
1325    *                                      Endorsement posts are not endorsable.
1326    *                                      Endorsements are not transitive. For
1327    *                                      instance, if post A is endorsed by post
1328    *                                      B, and an account wants to endorse B, in
1329    *                                      fact, the endorsement must refers to A.
1330    */
1331   int endorsePost(String handle, int id)
1332       throws HandleNotRecognisedException, PostIDNotRecognisedException,
     NotActionablePostException;
1333
1334   /**
1335    * The method creates a comment post referring to an existing post, similarly to
1336    * a reply on Twitter. A comment post is a special post. It contains a reference
1337    * to the post being commented upon.
1338    * <p>
1339    * The state of this SocialMediaPlatform must be be unchanged if any exceptions
1340    * are thrown.
1341    *
1342    * @param handle  of the account commenting a post.
1343    * @param id      of the post being commented.
1344    * @param message the comment post message.
1345    * @return the sequential ID of the created post.
1346    * @throws HandleNotRecognisedException if the handle does not match to any
```

```
1347    *                                       account in the system.
1348    * @throws PostIDNotRecognisedException if the ID does not match to any post in
1349    *                                       the system.
1350    * @throws NotActionablePostException   if the ID refers to a endorsement post.
1351    *                                       Endorsement posts are not endorsable.
1352    *                                       Endorsements cannot be commented. For
1353    *                                       instance, if post A is endorsed by post
1354    *                                       B, and an account wants to comment B, in
1355    *                                       fact, the comment must refers to A.
1356    * @throws InvalidPostException         if the comment message is empty or has
1357    *                                       more than 100 characters.
1358    */
1359   int commentPost(String handle, int id, String message) throws
     HandleNotRecognisedException,
1360       PostIDNotRecognisedException, NotActionablePostException, InvalidPostException;
1361
1362   /**
1363    * The method removes the post from the platform. When a post is removed, all
1364    * its endorsements should be removed as well. All replies to this post should
1365    * be updated by replacing the reference to this post by a generic empty post.
1366    * <p>
1367    * The generic empty post message should be "The original content was removed
1368    * from the system and is no longer available.". This empty post is just a
1369    * replacement placeholder for the post which a reply refers to. Empty posts
1370    * should not be linked to any account and cannot be acted upon, i.e., it cannot
1371    * be available for endorsements or replies.
1372    * <p>
1373    * The state of this SocialMediaPlatform must be be unchanged if any exceptions
1374    * are thrown.
1375    *
1376    * @param id ID of post to be removed.
1377    * @throws PostIDNotRecognisedException if the ID does not match to any post in
1378    *                                       the system.
1379    */
1380   void deletePost(int id) throws PostIDNotRecognisedException;
1381
1382   /**
1383    * The method generates a formated string containing the details of a single
1384    * post. The format is as follows:
1385    *
1386    * <pre>
1387    * ID: [post ID]
1388    * Account: [account handle]
1389    * No. endorsements: [number of endorsements received by the post] | No. comments:
     [number of comments received by the post]
1390    * [post message]
1391    * </pre>
1392    *
1393    * @param id of the post to be shown.
1394    * @return a formatted string containing post's details.
1395    * @throws PostIDNotRecognisedException if the ID does not match to any post in
1396    *                                       the system.
1397    */
1398   String showIndividualPost(int id) throws PostIDNotRecognisedException;
1399
1400   /**
1401    * The method builds a StringBuilder showing the details of the current post and
1402    * all its children posts. The format is as follows:
1403    *
1404    * <pre>
```

```
1405    * {@link #showIndividualPost(int) showIndividualPost(id)}
1406    * |
1407    * [for reply: replies to the post sorted by ID]
1408    * |  > {@link #showIndividualPost(int) showIndividualPost(reply)}
1409    * </pre>
1410    *
1411    * See an example:
1412    *
1413    * <pre>
1414    * ID: 1
1415    * Account: user1
1416    * No. endorsements: 2 | No. comments: 3
1417    * I like examples.
1418    * |
1419    * | > ID: 3
1420    *     Account: user2
1421    *     No. endorsements: 0 | No. comments: 1
1422    *     No more than me...
1423    *     |
1424    *     | > ID: 5
1425    *         Account: user1
1426    *         No. endorsements: 0 | No. comments: 1
1427    *         I can prove!
1428    *         |
1429    *         | > ID: 6
1430    *             Account: user2
1431    *             No. endorsements: 0 | No. comments: 0
1432    *             prove it
1433    * | > ID: 4
1434    *     Account: user3
1435    *     No. endorsements: 4 | No. comments: 0
1436    *     Can't you do better than this?
1437    *
1438    * | > ID: 7
1439    *     Account: user5
1440    *     No. endorsements: 0 | No. comments: 1
1441    *     where is the example?
1442    *     |
1443    *     | > ID: 10
1444    *         Account: user1
1445    *         No. endorsements: 0 | No. comments: 0
1446    *         This is the example!
1447    * </pre>
1448    *
1449    * Continuing with the example, if the method is called for post ID=5
1450    * ({@code showIndividualPost(5)}), the return would be:
1451    *
1452    * <pre>
1453    * ID: 5
1454    * Account: user1
1455    * No. endorsements: 0 | No. comments: 1
1456    * I can prove!
1457    * |
1458    * | > ID: 6
1459    *     Account: user2
1460    *     No. endorsements: 0 | No. comments: 0
1461    *     prove it
1462    * </pre>
1463    *
1464    * @param id of the post to be shown.
```

```
1465     * @return a formatted StringBuilder containing the details of the post and its
1466     *         children.
1467     * @throws PostIDNotRecognisedException if the ID does not match to any post in
1468     *                                      the system.
1469     * @throws NotActionablePostException   if the ID refers to an endorsement post.
1470     *                                      Endorsement posts do not have children
1471     *                                      since they are not endorsable nor
1472     *                                      commented.
1473     */
1474    StringBuilder showPostChildrenDetails(int id) throws PostIDNotRecognisedException,
       NotActionablePostException;
1475
1476    // End Post-related methods ************************************
1477
1478    // Analytics-related methods ************************************
1479
1480    /**
1481     * This method identifies and returns the post with the most number of
1482     * endorsements, a.k.a. the most popular post.
1483     *
1484     * @return the ID of the most popular post.
1485     */
1486    int getMostEndorsedPost();
1487
1488    /**
1489     * This method identifies and returns the account with the most number of
1490     * endorsements, a.k.a. the most popular account.
1491     *
1492     * @return the ID of the most popular account.
1493     */
1494    int getMostEndorsedAccount();
1495
1496    // End Analytics-related methods ************************************
1497
1498    // Management-related methods ************************************
1499
1500    /**
1501     * Method empties this SocialMediaPlatform of its contents and resets all
1502     * internal counters.
1503     */
1504    void erasePlatform();
1505
1506    /**
1507     * Method saves this SocialMediaPlatform's contents into a serialised file, with
1508     * the filename given in the argument.
1509     *
1510     * @param filename location of the file to be saved
1511     * @throws IOException if there is a problem experienced when trying to save the
1512     *                     store contents to the file
1513     */
1514    void savePlatform(String filename) throws IOException;
1515
1516    /**
1517     * Method should load and replace this SocialMediaPlatform's contents with the
1518     * serialised contents stored in the file given in the argument.
1519     * <p>
1520     * The state of this SocialMediaPlatform's must be be unchanged if any
1521     * exceptions are thrown.
1522     *
1523     * @param filename location of the file to be loaded
```

```
1524    * @throws IOException            if there is a problem experienced when trying
1525    *                                to load the store contents from the file
1526    * @throws ClassNotFoundException if required class files cannot be found when
1527    *                                loading
1528    */
1529   void loadPlatform(String filename) throws IOException, ClassNotFoundException;
1530
1531   // End Management-related methods ****************************************
1532
1533 }
1534 package socialmedia;
1535
1536 /**
1537  * SocialMediaPlatform interface. This interface is a more elaborated version of
1538  * the MiniSocialMediaPlatform. The no-argument constructor of a class
1539  * implementing this interface should initialise the SocialMediaPlatform as an
1540  * empty platform with no initial accounts nor posts within it. For Pair
1541  * submissions.
1542  *
1543  * @author Diogo Pacheco
1544  * @version 1.0
1545  *
1546  */
1547 public interface SocialMediaPlatform extends MiniSocialMediaPlatform {
1548
1549   // Account-related methods ***************************************
1550
1551   /**
1552    * The method creates an account in the platform with the given handle and
1553    * description.
1554    * <p>
1555    * The state of this SocialMediaPlatform must be be unchanged if any exceptions
1556    * are thrown.
1557    *
1558    * @param handle      account's handle.
1559    * @param description account's description.
1560    * @throws IllegalHandleException if the handle already exists in the platform.
1561    * @throws InvalidHandleException if the new handle is empty, has more than 30
1562    *                                characters, or has white spaces.
1563    * @return the ID of the created account.
1564    */
1565   int createAccount(String handle, String description) throws IllegalHandleException,
     InvalidHandleException;
1566
1567   /**
1568    * The method removes the account with the corresponding handle from the
1569    * platform. When an account is removed, all of their posts and likes should
1570    * also be removed.
1571    * <p>
1572    * The state of this SocialMediaPlatform must be be unchanged if any exceptions
1573    * are thrown.
1574    *
1575    * @param handle account's handle.
1576    * @throws HandleNotRecognisedException if the handle does not match to any
1577    *                                      account in the system.
1578    */
1579   void removeAccount(String handle) throws HandleNotRecognisedException;
1580
1581   /**
1582    * The method updates the description of the account with the respective handle.
```

```
1583    * <p>
1584    * The state of this SocialMediaPlatform must be be unchanged if any exceptions
1585    * are thrown.
1586    *
1587    * @param handle      handle to identify the account.
1588    * @param description new text for description.
1589    * @throws HandleNotRecognisedException if the handle does not match to any
1590    *                                      account in the system.
1591    */
1592   void updateAccountDescription(String handle, String description) throws
     HandleNotRecognisedException;
1593
1594   // End Post-related methods ************************************
1595
1596   // Analytics-related methods ************************************
1597
1598   /**
1599    * This method returns the current total number of accounts present in the
1600    * platform. Note, this is NOT the total number of accounts ever created since
1601    * the current total should discount deletions.
1602    *
1603    * @return the total number of accounts in the platform.
1604    */
1605   int getNumberOfAccounts();
1606
1607   /**
1608    * This method returns the current total number of original posts (i.e.,
1609    * disregarding endorsements and comments) present in the platform. Note, this
1610    * is NOT the total number of posts ever created since the current total should
1611    * discount deletions.
1612    *
1613    * @return the total number of original posts in the platform.
1614    */
1615   int getTotalOriginalPosts();
1616
1617   /**
1618    * This method returns the current total number of endorsement posts present in
1619    * the platform. Note, this is NOT the total number of endorsements ever created
1620    * since the current total should discount deletions.
1621    *
1622    * @return the total number of endorsement posts in the platform.
1623    */
1624   int getTotalEndorsmentPosts();
1625
1626   /**
1627    * This method returns the current total number of comments posts present in the
1628    * platform. Note, this is NOT the total number of comments ever created since
1629    * the current total should discount deletions.
1630    *
1631    * @return the total number of comments posts in the platform.
1632    */
1633   int getTotalCommentPosts();
1634
1635   // End Management-related methods ************************************
1636
1637 }
1638 import socialmedia.AccountIDNotRecognisedException;
1639 import socialmedia.SocialMedia;
1640 import socialmedia.IllegalHandleException;
1641 import socialmedia.InvalidHandleException;
```

```java
import socialmedia.SocialMediaPlatform;

/**
 * A short program to illustrate an app testing some minimal functionality of a
 * concrete implementation of the SocialMediaPlatform interface -- note you will
 * want to increase these checks, and run it on your SocialMedia class (not the
 * BadSocialMedia class).
 *
 *
 * @author Diogo Pacheco
 * @version 1.0
 */
public class SocialMediaPlatformTestApp {

  /**
   * Test method.
   *
   * @param args not used
   */
  public static void main(String[] args) {
    System.out.println("The system compiled and started the execution...");

    SocialMediaPlatform platform = new SocialMedia();

    assert (platform.getNumberOfAccounts() == 0) : "Innitial SocialMediaPlatform not empty as required.";
    assert (platform.getTotalOriginalPosts() == 0) : "Innitial SocialMediaPlatform nc empty as required.";
    assert (platform.getTotalCommentPosts() == 0) : "Innitial SocialMediaPlatform not empty as required.";
    assert (platform.getTotalEndorsmentPosts() == 0) : "Innitial SocialMediaPlatform not empty as required.";

    Integer id;
    try {
      id = platform.createAccount("my_handle");
      assert (platform.getNumberOfAccounts() == 1) : "number of accounts registered i the system does not match";

      platform.removeAccount(id);
      assert (platform.getNumberOfAccounts() == 0) : "number of accounts registered i the system does not match";

    } catch (IllegalHandleException e) {
      assert (false) : "IllegalHandleException thrown incorrectly";
    } catch (InvalidHandleException e) {
      assert (false) : "InvalidHandleException thrown incorrectly";
    } catch (AccountIDNotRecognisedException e) {
      assert (false) : "AccountIDNotRecognizedException thrown incorrectly";
    }

  }

}
```