1 Account.java

```java
package socialmedia;
import java.util.*;
import java.io.Serializable;

/**
 * Represents an individual account.
 * @author Ashley Card
 * @author Maryia Fralova
 * @version 1.0
 */
public class Account implements Serializable{

   // Attributes
   private String handle;
   private int id;
   private String description;
   private HashMap<String, ArrayList<Integer>> storage;

   // Getter methods

   /** Gets account's handle.
    * @return handle of account
    */
   public String getHandle(){
      return handle;
   }

   /** Gets account's ID.
    * @return ID of account
    */
   public int getID(){
      return id;
   }

   /** Gets account's description.
    * @return description of account
    */
   public String getDescription(){
      return description;
   }

   /** Gets account's storage of posts.
    * @return storage of account
    */
   public HashMap<String, ArrayList<Integer>> getAccountStorage(){
      return storage;
   }

   // Setter methods

   /** Sets account's handle.
    * @param handle handle of account
```

```
     */
    public void setHandle(String handle){
        this.handle = handle;
    }

    /** Sets account's description.
     * @param description description of account
     */
    public void setDescription(String description){
        this.description = description;
    }

    /** Adds original post ID to the account's storage of posts.
     * @param idPost ID of post to be added to account original storage
     */
    public void addToAccountStorageOriginal(int idPost){
        ArrayList<Integer> value = new ArrayList<>();
        value = storage.get("original");
        value.add(idPost);
        storage.put("original", value);
    }

    /** Adds comment post ID to the account's storage of posts.
     * @param idPost ID of post to be added to account comment storage
     */
    public void addToAccountStorageComment(int idPost){
        ArrayList<Integer> value = new ArrayList<>();
        value = storage.get("comments");
        value.add(idPost);
        storage.put("comments", value);
    }

    /** Adds endorsement post ID to the account's storage of posts.
     * @param idPost ID of post to be added to account endorsement storage
     */
    public void addToAccountStorageEndors(int idPost){
        ArrayList<Integer> value = new ArrayList<>();
        value = storage.get("endorsements");
        value.add(idPost);
        storage.put("endorsements", value);
    }

    // Constructors

    /** Creates account with a specified handle.
     * @param handle handle of account
     * @param id ID of account
     */
    public Account(String handle, int id){
        this.handle = handle;
        this.id = id;
        description="";
        storage = new HashMap<String, ArrayList<Integer>>();
        /*
         * original - posts that account made
     * comments - comments that account made
     * endorsements - account made
     * ArrayList contains post IDs
     */
    storage.put("original", new ArrayList<Integer>());
    storage.put("comments", new ArrayList<Integer>());
    storage.put("endorsements", new ArrayList<Integer>());
  }

  /** Creates account with a specified handle and description.
   * @param handle handle of account
   * @param id ID of account
   * @param description description of account
   */
  public Account(String handle,int id, String description){
    this.handle = handle;
    this.id = id;
    this.description = description;
    storage = new HashMap<String, ArrayList<Integer>>();
    /*
     * original - posts that account made
     * comments - comments that account made
     * endorsements - account made
     * ArrayList contains post IDs
     */
    storage.put("original", new ArrayList<Integer>());
    storage.put("comments", new ArrayList<Integer>());
    storage.put("endorsements", new ArrayList<Integer>());
  }
}
```

2 Post.java

```
package socialmedia;
import java.util.*;
import java.io.Serializable;

/**
 * Represents a post.
 * @author Ashley Card
 * @author Maryia Fralova
 * @version 1.0
 */
public class Post implements Serializable{

  // Attributes
  private int numIdentifier;
  private String message;
  private int pointerToOriginal;
  private HashMap<String, ArrayList<Integer>> storage ;

  // Getter methods

  /** Gets post's ID.
```

```java
     * @return numIdentifier ID of post
     */
    public int getNumIdentifier(){
       return numIdentifier;
    }

    /** Gets post's message.
     * @return message of post
     */
    public String getMessage(){
       return message;
    }

    /** Gets post's parent post.
     * @return pointerToOriginal ID of post parent
     */
    public int getPointerToOriginal(){
       return pointerToOriginal;
    }

    /** Gets post's storage with comments and endorsements it received.
     * @return storage of post comment and endorsement IDs
     */
    public HashMap<String, ArrayList<Integer>> getPostStorage(){
       return storage;
    }

    // Setter methods

    /** Adds comment post to the storage of the post being endrosed.
    * @param idPost id of post being added to comment storage
    */
    public void addToPostStorageComment(int idPost){
       ArrayList<Integer> value = new ArrayList<>();
       value = storage.get("comments");
       value.add(idPost);
       storage.put("comments", value);
    }

    /** Adds endrosement post to the storage of the post being endrosed.
    * @param idPost id of post being added to endorsement storage
    */
    public void addToPostStorageEndors(int idPost){
       ArrayList<Integer> value = new ArrayList<>();
       value = storage.get("endorsements");
       value.add(idPost);
       storage.put("endorsements", value);
    }

    /** Sets post's message.
    * @param message post message
    */
    public void setMessage(String message){
       this.message = message;
    }
```

```java

   /** Creates an original post.
   * @param numIdentifier ID of post
   * @param message post message
   */
   public Post(int numIdentifier,String message){
      this.numIdentifier = numIdentifier;
      this.message = message;
      storage = new HashMap<String, ArrayList<Integer>>();
      pointerToOriginal = -1;
      // endorsements - the no. endorsements the post has
      // comments - the no. comments the post has
      // Arraylist contains the postIDs of these
      storage.put("comments", new ArrayList<Integer>());
      storage.put("endorsements", new ArrayList<Integer>());
   }

   /** Creates a comment post.
   * @param numIdentifier ID of post
   * @param message post message
   * @param pointerToOriginal ID of parent post
   */
   public Post(int numIdentifier,String message,int pointerToOriginal){
      this.numIdentifier = numIdentifier;
      this.message = message;
      this.pointerToOriginal = pointerToOriginal;
      storage = new HashMap<String, ArrayList<Integer>>();
      /* endorsements - the no. endorsements the post has
       comments - the no. comments the post has
       */
      storage.put("comments", new ArrayList<Integer>());
      storage.put("endorsements", new ArrayList<Integer>());
   }

   /** Creates an endorsement post.
   * @param numIdentifier ID of post
   * @param pointerToOriginal ID of parent post
   */
   public Post(int numIdentifier,int pointerToOriginal){
      this.numIdentifier = numIdentifier;
      this.pointerToOriginal = pointerToOriginal;
      message = new String();
   }

   /** Creates an empty post.
   * @param numIdentifier ID of post
   */
   public Post(int numIdentifier){
      this.numIdentifier = numIdentifier;
      message = "The original content was removed from the system and is no longer available.";
   }
}
```

3 SocialMedia.java

```java
package socialmedia;

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

/**
 * SocialMedia is a compiling, fully-functioning implementor of
 * the SocialMediaPlatform interface.
 * @author Ashley Card
 * @author Maryia Fralova
 * @version 1.0
 */
public class SocialMedia implements SocialMediaPlatform {

   // Attributes
   private ArrayList<Account> listOfAccounts = new ArrayList<>();
   private ArrayList<Post> listOfPosts = new ArrayList<>();
   private ArrayList<Post> listOfEmptyPosts = new ArrayList<>();
   private int idAccount = 0;
   private int idPost = 0;

   // Getter methods

   /** Gets the array of existing accounts.
    * @return listOfAccounts ArrayList of all account IDs
    */
   public ArrayList<Account> getAccounts() {
      return listOfAccounts;
   }

   /** Gets the array of existing posts.
    * @return listOfPosts ArrayList of all post IDs
    */
   public ArrayList<Post> getPosts() {
      return listOfPosts;
   }

   /** Gets the array of existing empty posts.
    * @return listOfEmptyPosts ArrayList of empty post IDs
    */
   public ArrayList<Post> getEmptyPosts() {
      return listOfEmptyPosts;
   }

   /** Gets account's ID.
    * @return idAccount ID of account
    */
   public int getIDAccount() {
      return idAccount;
   }

   /** Gets posts's ID.
```

```java
     * @return idPost ID of post
     */
    public int getIDPost() {
       return idPost;
    }

    // Methods

    @Override
    public int createAccount(String handle) throws IllegalHandleException, InvalidHandleException {

       int numAccountsInitially = getNumberOfAccounts();
       // Check if the handle already exists
       for (Account i : listOfAccounts) {
          if (i.getHandle().equals(handle)) {
             throw new IllegalHandleException();
          }
       }

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

       // Create a new account object
       Account acc = new Account(handle, idAccount);
       listOfAccounts.add(acc);

       // Asserting if account was added
       assert ((numAccountsInitially + 1) == getNumberOfAccounts()): "The account was added incorrectly";

       return idAccount++;
    }

    @Override
    public int createAccount(String handle, String description) throws IllegalHandleException,
        InvalidHandleException {

       int numAccountsInitially = getNumberOfAccounts();
       // Check if the handle already exists
       for (Account i : listOfAccounts) {
          if (i.getHandle().equals(handle)) {
             throw new IllegalHandleException();
          }
       }

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

       // Create a new account object
       Account acc = new Account(handle, idAccount, description);
       listOfAccounts.add(acc);

```

```java
        // Asserting if account was added
        assert ((numAccountsInitially + 1) == getNumberOfAccounts()): "The account was added incorrectly";

        return idAccount++;
    }

    @Override
    public void removeAccount(int id) throws AccountIDNotRecognisedException {

        // Finds the account with the given id
        int counter = 0;
        int numAccountsInitially = getNumberOfAccounts();
        boolean isThere = false;
        for (Account acc : listOfAccounts) {
            if (acc.getID() == id) {

                // Gets the accounts post storage
                HashMap<String, ArrayList<Integer>> storage = new HashMap<>();
                storage = acc.getAccountStorage();
                ArrayList<Integer> origposts = new ArrayList<Integer>();
                ArrayList<Integer> commposts = new ArrayList<Integer>();
                ArrayList<Integer> endorposts = new ArrayList<Integer>();
                origposts = storage.get("original");
                commposts = storage.get("comments");
                endorposts = storage.get("endorsements");
                ArrayList<Integer> listOfPostsToDelete = new ArrayList<>();

                //Iterates over all of its posts
                for (int post : origposts) {
                    listOfPostsToDelete.add(post);
                }

                for (int post : commposts) {
                    listOfPostsToDelete.add(post);
                }

                for (int post : endorposts) {
                    listOfPostsToDelete.add(post);
                }

                // Deletes each post from everywhere
                for (int i : listOfPostsToDelete) {
                    try {
                        deletePost(i);
                    }
                    catch(PostIDNotRecognisedException e){}
                }

                listOfAccounts.remove(counter);
                isThere = true;
                break;
            }
            counter++;
        }

```

```java
      // Asserting account was removed
      assert ((numAccountsInitially - 1) == getNumberOfAccounts()): "The account was removed incorrectly";

      // If the id doesn't exist
      if (!isThere){
         throw new AccountIDNotRecognisedException();
      }
   }

   @Override
   public void removeAccount(String handle) throws HandleNotRecognisedException {

      // Finds the account with the given handle
      int counter = 0;
      int numAccountsInitially = getNumberOfAccounts();
      boolean isThere = false;
      for (Account acc : listOfAccounts) {
         if (acc.getHandle() == handle) {

            // Gets the accounts post storage
            HashMap<String, ArrayList<Integer>> storage = new HashMap<>();
            storage = acc.getAccountStorage();
            ArrayList<Integer> origposts = new ArrayList<Integer>();
            ArrayList<Integer> commposts = new ArrayList<Integer>();
            ArrayList<Integer> endorposts = new ArrayList<Integer>();
            origposts = storage.get("original");
            commposts = storage.get("comments");
            endorposts = storage.get("endorsements");
            ArrayList<Integer> listOfPostsToDelete = new ArrayList<>();

            //Iterates over all of its posts
            for (int post : origposts) {
               listOfPostsToDelete.add(post);
            }

            for (int post : commposts) {
               listOfPostsToDelete.add(post);
            }

            for (int post : endorposts) {
               listOfPostsToDelete.add(post);
            }

            // Deletes each post from everywhere
            for (int i : listOfPostsToDelete) {
               try {
                  deletePost(i);
               }
               catch(PostIDNotRecognisedException e){}
            }

            listOfAccounts.remove(counter);
            isThere = true;
            break;
         }
```

```java
            counter++;
        }

        // Asserting account was removed
        assert ((numAccountsInitially - 1) == getNumberOfAccounts()): "The account was removed incorrectly";

        // If the handle doesn't exist
        if (!isThere){
            throw new HandleNotRecognisedException();
        }
    }

    @Override
    public void changeAccountHandle(String oldHandle, String newHandle)
            throws HandleNotRecognisedException, IllegalHandleException, InvalidHandleException {

        // Check if newHandle already exists
        for (Account i : listOfAccounts) {
            if (i.getHandle().equals(newHandle)) {
                throw new IllegalHandleException();
            }
        }

        // Check if new handle is valid
        if (newHandle.isEmpty() || newHandle.length() > 30 || newHandle.matches(".*\\s.*")){
            throw new InvalidHandleException();
        }

        // Find the account that matches oldHandle
        boolean isThere = false;
        for (Account i : listOfAccounts) {
            if (i.getHandle().equals(oldHandle)){
                i.setHandle(newHandle);
                isThere = true;

                // Asserting the handle was changed
                assert (newHandle == i.getHandle()): "The handle was changed incorrectly";
                break;
            }
        }

        // If the oldHandle doesn't exist
        if (!isThere){
            throw new HandleNotRecognisedException();
        }
    }

    @Override
    public void updateAccountDescription(String handle, String description) throws
           HandleNotRecognisedException {

        // Find the account with the given handle
        boolean isThere = false;
        for (Account i : listOfAccounts) {
            if (i.getHandle().equals(handle)){
```

```java
            i.setDescription(description);
            isThere = true;

            // Asserting the description was changed
            assert (description == i.getDescription()): "The description was changed incorrectly";
            break;
         }
      }

      // If the handle doesn't exist
      if (!isThere){
         throw new HandleNotRecognisedException();
      }
   }

   @Override
   public String showAccount(String handle) throws HandleNotRecognisedException {

      // Find the account with the given handle
      boolean isThere = false;
      int index=0;
      for (Account i : listOfAccounts) {
         if (i.getHandle().equals(handle)){
            isThere = true;
            break;
         }
         index++;
      }

      // If the handle doesn't exist
      if (!isThere){
         throw new HandleNotRecognisedException();
      }

      // Getting the posts made by the account
      int sumOfEndors = 0;
      HashMap<String, ArrayList<Integer>> storageOfPostsFromAccount =
          listOfAccounts.get(index).getAccountStorage();

      // Iterate over original
      for (int eachID : storageOfPostsFromAccount.get("original")){
         for (Post i : listOfPosts) {
            if (i.getNumIdentifier() == eachID) {
               HashMap<String, ArrayList<Integer>> storageOfPostsFromPost = i.getPostStorage();
               sumOfEndors+=storageOfPostsFromPost.get("endorsements").size();
            }
         }
      }

      // Iterate over comments
      for (int eachID : storageOfPostsFromAccount.get("comments")){
         for (Post i : listOfPosts) {
            if (i.getNumIdentifier() == eachID) {
               HashMap<String, ArrayList<Integer>> storageOfPostsFromPost = i.getPostStorage();
               sumOfEndors+=storageOfPostsFromPost.get("endorsements").size();
            }
         }
      }

      // Find each post that the account's made and count its endorsements
      int sumOfPosts =
          storageOfPostsFromAccount.get("original").size()+storageOfPostsFromAccount.get("comments").size()+storageOfP
      String toReturn = String.format("ID: %d%n"+
                           "Handle: %s%n"+
                           "Description: %s%n"+
                           "Post count: %d%n"+
                           "Endorse count: %d", listOfAccounts.get(index).getID(),
                               listOfAccounts.get(index).getHandle(),
                               listOfAccounts.get(index).getDescription(), sumOfPosts, sumOfEndors);
      return toReturn;
   }

   @Override
   public int createPost(String handle, String message) throws HandleNotRecognisedException,
       InvalidPostException {

      int numPostsInitially = listOfPosts.size();

      // Check if the message is valid
      if (message.isEmpty() || message.length() > 100){
         throw new InvalidPostException();
      }

      // Find the account with the given handle and create the post
      boolean isThere = false;
      int counter = 0;
      for (Account i : listOfAccounts) {
         if (i.getHandle() == handle) {
            listOfAccounts.get(counter).addToAccountStorageOriginal(idPost);
            Post post = new Post(idPost, message);
            listOfPosts.add(post);
            isThere = true;
            break;
         }
         counter++;
      }

      // Asserting post was added
      assert ((numPostsInitially + 1) == listOfPosts.size()): "The post was added incorrectly";

      // If the handle doesn't exist
      if (!isThere){
         throw new HandleNotRecognisedException();
      }
      return idPost++;
   }

   @Override
   public int endorsePost(String handle, int id) throws HandleNotRecognisedException,
       PostIDNotRecognisedException, NotActionablePostException {

      int numPostsInitially = listOfPosts.size();
      // Find the post with the given id
      String message="";
      String newMessage="";
      boolean postIsThere = false;
      int postCounter = 0;

      // Finds the post that is endorsed
      for (Post i : listOfPosts) {
         if (i.getNumIdentifier() == id) {
            postIsThere = true;

            // Checks if it is an endorsement post itself
            if (listOfPosts.get(postCounter).getPostStorage() == null){
               throw new NotActionablePostException();
            }

            // Adds the id of created post to the endorsed post storage
            listOfPosts.get(postCounter).addToPostStorageEndors(idPost);
            message = i.getMessage();
            break;
         }
         postCounter++;
      }

      // If the post doesn't exist
      if (!postIsThere){
         throw new PostIDNotRecognisedException();
      }

      // Find account that was endorsed and edit endorsed message
      for(Account acc : listOfAccounts){
         HashMap<String, ArrayList<Integer>> storage = new HashMap<>();
         storage = acc.getAccountStorage();

         // Iterate over original
         boolean found = false;
         int counter = 0;
         for (int orig : storage.get("original")){
            if (orig == id){
               found = true;
               break;
               }
            counter++;
         }

         // Iterate over comments
         if (!found){
            counter = 0;
            for (int comm : storage.get("comments")){
               if (comm == id){
                  found = true;
                  break;
               }
```

```
                counter++;
            }
        }

        if (found){
            String endorsedHandle = acc.getHandle();
            //Edit message
            newMessage = "EP@" + endorsedHandle + ": " + message;
            break;
        }
    }

    // Find the account with the given handle
    boolean isThere = false;
    int counter = 0;
    for (Account i : listOfAccounts) {
        if (i.getHandle() == handle) {
            listOfAccounts.get(counter).addToAccountStorageEndors(idPost);
            Post post = new Post(idPost, id);
            post.setMessage(newMessage);
            listOfPosts.add(post);
            isThere = true;
            break;
        }
        counter++;
    }

    // Asserting post was added
    assert ((numPostsInitially + 1) == listOfPosts.size()): "The post was added incorrectly";

    // If the handle doesn't exist
    if (!isThere){
        throw new HandleNotRecognisedException();
    }
    return idPost++;
}

@Override
public int commentPost(String handle, int id, String message) throws HandleNotRecognisedException,
        PostIDNotRecognisedException, NotActionablePostException, InvalidPostException {

    int numPostsInitially = listOfPosts.size();

    // Check if the message is valid
    if (message.isEmpty() || message.length() > 100){
        throw new InvalidPostException();
    }

    // Find the post with the given id
    boolean postIsThere = false;
    int postCounter = 0;
    for (Post i : listOfPosts) {
        if (i.getNumIdentifier() == id) {
            postIsThere = true;
            if (listOfPosts.get(postCounter).getPostStorage() == null){
                throw new NotActionablePostException();
            }
            listOfPosts.get(postCounter).addToPostStorageComment(idPost);
            break;
         }
         postCounter++;
      }

      // If the post doesn't exist
      if (!postIsThere){
         throw new PostIDNotRecognisedException();
      }

      // Find the account with the given handle and create the comment
      boolean isThere = false;
      int counter = 0;
      for (Account i : listOfAccounts) {
         if (i.getHandle() == handle) {
            listOfAccounts.get(counter).addToAccountStorageComment(idPost);
            Post post = new Post(idPost, message,id);
            listOfPosts.add(post);
            isThere = true;
            break;
         }
         counter++;
      }

      // Asserting post was added
      assert ((numPostsInitially + 1) == listOfPosts.size()): "The post was added incorrectly";

      // If the handle doesn't exist
      if (!isThere){
         throw new HandleNotRecognisedException();
      }
      return idPost++;
   }

   @Override
   public void deletePost(int id) throws PostIDNotRecognisedException {

      // Find the post with the given id
      boolean postIsThere = false;
      int postCounter = 0;
      int numPostsInitially = listOfPosts.size();
      for (Post i : listOfPosts) {
         List<Integer> counters = new ArrayList<>();
         if (i.getNumIdentifier() == id) {
            postIsThere = true;
            Post post = new Post(id);
            listOfEmptyPosts.add(post);

            // Checks type of post
            boolean isOriginal = false;
            if (i.getPointerToOriginal() == -1) {
               isOriginal = true;
```

```java
            }
            // If it is a comment or endorsement, removes from parent post
            if (!isOriginal) {
               for (Post postmain : listOfPosts) {
                  if (postmain.getNumIdentifier() == i.getPointerToOriginal()) {
                     HashMap<String, ArrayList<Integer>> postMainStorage = postmain.getPostStorage();
                     ArrayList<Integer> idList = new ArrayList<>();
                     idList.add(id);
                     postMainStorage.get("comments").removeAll(idList);
                     postMainStorage.get("endorsements").removeAll(idList);
                  }
               }
            }

            // Checking that it isn't an endorsement
            if (listOfPosts.get(postCounter).getPostStorage() != null){

               // Remove all its endorsements
               HashMap<String, ArrayList<Integer>> storagePosts =
                   listOfPosts.get(postCounter).getPostStorage();
               ArrayList<Integer> value = new ArrayList<Integer>();
               value = storagePosts.get("endorsements");

               // Copy of the storage with postIDs of endorsements to delete from Account storage
               ArrayList<Integer> storageOfEndors = new ArrayList<Integer>();
               for (int val : value){
                  storageOfEndors.add(val);
               }

               // Deleting all the endorsements from listOfPosts
               value.clear();
            }


            // Remove from Account's storage
            for(Account acc : listOfAccounts){
               HashMap<String, ArrayList<Integer>> storage = new HashMap<>();
               storage = acc.getAccountStorage();

               // Iterate over original
               boolean deleted = false;
               int counter = 0;
               for (int orig : storage.get("original")){
                     if (orig == id){
                        storage.get("original").remove(counter);
                        deleted = true;
                        break;
                     }
                     counter++;
               }

               // Iterate over comments
               if (!deleted){
                  counter = 0;
                  for (int comm : storage.get("comments")){
                    if (comm == id){
                        storage.get("comments").remove(counter);
                        deleted = true;
                        break;
                    }
                    counter++;
                }
            }

            // Iterate over comments
            if (!deleted){
                counter = 0;
                for (int endors : storage.get("endorsements")){
                    if (endors == id){
                        storage.get("endorsements").remove(counter);
                        break;
                    }
                    counter++;
                }
            }
            // Checking that it isn't an endorsement
            if (listOfPosts.get(postCounter).getPostStorage() != null){

                // Removes all its endorsements
                HashMap<String, ArrayList<Integer>> storagePosts =
                     listOfPosts.get(postCounter).getPostStorage();
                ArrayList<Integer> value = new ArrayList<Integer>();
                value = storagePosts.get("endorsements");
                ArrayList<Integer> storageOfEndors = new ArrayList<Integer>();

                // Iterating over endorsements
                for (int endor : storage.get("endorsements")){
                    for (int idEndors : storageOfEndors){
                        if (endor == idEndors){
                            counters.add(endor);
                            break;
                        }
                    }
                }
                storage.get("endorsements").removeAll(counters);
            }

        }

        // Removes post from listOfPosts
        listOfPosts.remove(listOfPosts.get(postCounter));

        // also need to remove endorsemetns from ListOfPosts
        int counter = 0;
        ArrayList<Post> indexes = new ArrayList<>();

        // Finds the indexes of the endorsements to be removed
        for (Post a : listOfPosts){
            for (int idInCounters : counters){
                if (a.getNumIdentifier() == idInCounters){
```

```java
                     indexes.add(a);
                     break;
                  }
               }
               counter++;
            }
            listOfPosts.removeAll(indexes);
            break;
         }
         postCounter++;
      }

      // Asserting post was removed
      assert ((numPostsInitially - 1) == listOfPosts.size()): "The post was removed incorrectly";

      // If the post doesn’t exist
      if (!postIsThere){
         throw new PostIDNotRecognisedException();
      }
   }

   @Override
   public String showIndividualPost(int id) throws PostIDNotRecognisedException {

      // Finding the account with the post
      boolean postIsThere = false;
      String handle = "";
      for (Account acc : listOfAccounts){
         HashMap<String, ArrayList<Integer>> storage = new HashMap<>();
         storage = acc.getAccountStorage();

         // Search the original
         for (int orig : storage.get("original")){
            if (orig == id){
               handle = acc.getHandle();
               postIsThere = true;
               break;
            }
         }

         // Search the comments
         for (int comm : storage.get("comments")){
            if (comm == id){
               handle = acc.getHandle();
               postIsThere = true;
               break;
            }
         }

         // Search the endorsements
         for (int endor : storage.get("endorsements")){
            if (endor == id){
               handle = acc.getHandle();
               postIsThere = true;
               break;
            }
         }
      }

      // If the post doesn't exist
      if (!postIsThere){
            throw new PostIDNotRecognisedException();
      }

      // Finds the index of the post in listOfPosts
      int indexNeededPost=0;
      for (Post post : listOfPosts){
         if (post.getNumIdentifier() == id){
            indexNeededPost = listOfPosts.indexOf(post);
            break;
         }
      }

      // Checking if it is an endorsement post
      if (listOfPosts.get(indexNeededPost).getPostStorage() == null){
         String message = listOfPosts.get(indexNeededPost).getMessage();
         return String.format("ID: %d%n Account: %s%n No.endorsements: %d|No.comments: %d%n %s", id, handle,
              0, 0, message);
      }

      // Gets the number of endorsements the post has received
      ArrayList<Integer> postStorageEnd = new ArrayList<>();
      postStorageEnd = listOfPosts.get(indexNeededPost).getPostStorage().get("endorsements");
      int noEndor = postStorageEnd.size();

      // Gets the number of comments the post has received
      ArrayList<Integer> postStorageComm = new ArrayList<>();
      postStorageComm = listOfPosts.get(indexNeededPost).getPostStorage().get("comments");
      int noComm = postStorageComm.size();

      // Gets the message of the post
      String message = listOfPosts.get(indexNeededPost).getMessage();

      return String.format("ID: %d%n Account: %s%n No.endorsements: %d|No.comments: %d%n %s", id, handle,
           noEndor, noComm, message);
   }

   @Override
   public StringBuilder showPostChildrenDetails(int id)
         throws PostIDNotRecognisedException, NotActionablePostException {

      boolean postIsThere = false;
      StringBuilder str = new StringBuilder();

      // Checking for post
      for (Post initialPost : listOfPosts){
         if (initialPost.getNumIdentifier() == id){

            // Check if the post is an endorsement
            if (initialPost.getPostStorage() == null){
                  throw new NotActionablePostException();
            }
            postIsThere = true;
         }
      }

      // If the post doesn't exist
      if (!postIsThere){
            throw new PostIDNotRecognisedException();
      }

      // Calling helper function
      str = showChildPostsHelper(id, 0);
      str.delete(0,5);
      return str;
   }

   /*
    * Helper function for showPostChildrenDetails.
    * Recursively displays children comments of a parent post.
    */
   private StringBuilder showChildPostsHelper(int id, int indentLevel) throws PostIDNotRecognisedException{
      StringBuilder str2 = new StringBuilder();
      boolean postIsThere2 = false;
      String stickthing = "|"; // stick :)

      for (Post initialPost : listOfPosts){
         if (initialPost.getNumIdentifier() == id){
            postIsThere2 = true;

            str2.append((stickthing.indent(indentLevel)));
            str2.append(("| > "+showIndividualPost(id)).indent(indentLevel));

            // Gets children of initial post
            ArrayList<Integer> initialPostStorageComm = new ArrayList<>();
            initialPostStorageComm = initialPost.getPostStorage().get("comments");

            // Loops through every comment in the posts storage
            for (Integer childPostID : initialPostStorageComm){
                    ArrayList<Integer> storage = new ArrayList<>();
                    storage = initialPost.getPostStorage().get("comments");

                    // If the child post has it's own children, calls this function on itself
                    if (!storage.isEmpty()){
                        str2.append(showChildPostsHelper(childPostID, indentLevel + 4));
                    // Otherwise, display the post
                    }else{
                        str2.append((stickthing.indent(indentLevel)));
                        str2.append(("| > "+showIndividualPost(childPostID)).indent(indentLevel));
                    }
            }
         }
      }

      // Checks if the post exists
```

```java
        if (!postIsThere2){
            System.out.println(str2);
            throw new PostIDNotRecognisedException();
        }
        return str2;
    }

    @Override
    public int getNumberOfAccounts() {
        return listOfAccounts.size();
    }

    @Override
    public int getTotalOriginalPosts() {

        int sumPosts = 0;

        // Increments a counter if post is an original
        for (Post post : listOfPosts){
            if (post.getPointerToOriginal() == -1){
                sumPosts++;
            }
        }
        return sumPosts;
    }

    @Override
    public int getTotalEndorsmentPosts() {

        int sumPosts = 0;

        // Increments a counter if post is an endorsement
        for (Post post : listOfPosts){
            if (post.getPostStorage() == null){
                sumPosts++;
            }
        }
        return sumPosts;
    }

    @Override
    public int getTotalCommentPosts() {

        int sumPosts = 0;

        // Increments a counter if post is a comment
        for (Post post : listOfPosts){
            if (post.getPostStorage() != null && post.getPointerToOriginal() != -1){
                sumPosts++;
            }
        }
        return sumPosts;
    }

    @Override
    public int getMostEndorsedPost() {

        int max = 0;
        int idPost = 0;

        // Iterating over every post
        for (Post post : listOfPosts){
            HashMap<String, ArrayList<Integer>> storagePosts = post.getPostStorage();
            ArrayList<Integer> storageEndors = new ArrayList<Integer>();

            // Checks for endorsement storage
            if (post.getPostStorage() != null){
                storageEndors = storagePosts.get("endorsements");

                // Changes max value to current value if greater
                if (storageEndors.size() > max){
                    max = storageEndors.size();
                    idPost = post.getNumIdentifier();
                }
            }
        }
        return idPost;
    }

    @Override
    public int getMostEndorsedAccount() {

        int max = 0;
        int idOfAccount = 0;
        for (Account acc : listOfAccounts){

            // Getting the posts made by the account
            int sumOfEndors = 0;
            HashMap<String, ArrayList<Integer>> storageOfPostsFromAccount = acc.getAccountStorage();

            // Iterate over original
            for (int eachID : storageOfPostsFromAccount.get("original")){
                for (Post i : listOfPosts) {
                    if (i.getNumIdentifier() == eachID) {
                        HashMap<String, ArrayList<Integer>> storageOfPostsFromPost = i.getPostStorage();
                        sumOfEndors+=storageOfPostsFromPost.get("endorsements").size();
                    }
                }
            }

            // Iterate over comments
            for (int eachID : storageOfPostsFromAccount.get("comments")){
                for (Post i : listOfPosts) {
                    if (i.getNumIdentifier() == eachID) {
                        HashMap<String, ArrayList<Integer>> storageOfPostsFromPost = i.getPostStorage();
                        sumOfEndors+=storageOfPostsFromPost.get("endorsements").size();
                    }
                }
            }

            // Changes max value to current value if greater
            if (sumOfEndors > max){
               max = sumOfEndors;
               idOfAccount = acc.getID();
            }
         }
         return idOfAccount;
      }

      @Override
      public void erasePlatform() {
         listOfAccounts.clear();
         listOfPosts.clear();
         listOfEmptyPosts.clear();
         idAccount = 0;
         idPost = 0;
      }

      @Override
      public void savePlatform(String filename) throws IOException {

         // Saves the platform global variables to a file
         try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(filename))) {
            out.writeObject(listOfAccounts);
            out.writeObject(listOfPosts);
            out.writeObject(listOfEmptyPosts);
            out.writeObject(idAccount);
            out.writeObject(idPost);
         } catch (IOException e){
            throw new IOException();
         }
      }

      @Override
      public void loadPlatform(String filename) throws IOException, ClassNotFoundException {

         // Gets each global variable from the saved file
         try (ObjectInputStream in = new ObjectInputStream(new FileInputStream(filename))) {
            Object obj = in.readObject();
            if (obj instanceof ArrayList)
              listOfAccounts = (ArrayList<Account>) obj;
            obj = in.readObject();
            if (obj instanceof ArrayList)
              listOfPosts = (ArrayList<Post>) obj;
             obj = in.readObject();
            if (obj instanceof ArrayList)
              listOfEmptyPosts = (ArrayList<Post>) obj;
            obj = in.readObject();
            if (obj instanceof ArrayList)
              idAccount = (Integer) obj;
            obj = in.readObject();
            if (obj instanceof ArrayList)
              idPost = (Integer) obj;
         } catch (ClassNotFoundException e){
            throw new ClassNotFoundException();
      } catch (IOException e){
         throw new IOException();
      }
   }
}
```