University of Exeter

College of Engineering, Mathematics and Physical Sciences

ECM1410

Object-Oriented Programming

Coursework

Date Set: 17th February 2023
Date Due: 31st March 2023

This coursework comprises 100% of the overall module assessment.

This is a exercise, and your attention is drawn to the guidelines on collaboration and plagiarism in the College Handbook (`exeter.ac.uk/students/administration/complaintsandappeals/academicmisconduct/`).

This assessment covers the use and implementation of a range of object-oriented concepts using the Java programming language that you are covering in ECM1410. The assignment is *summative*. Please ensure you read the entire document before you begin the assessment.

This coursework is devised for you to do in your own time, rather than in the workshop sessions, and will help you enhance your understanding of object-oriented programming. Remember to follow the established deadline and submission guidelines in the module's ELE page, and to use your own words when writing your answers.

1 Development paradigm

This summative CA for ECM1410 is a pair submission — with details as circulated previously in the document *"ECM1410 Object-Oriented Programming Development paradigm in summative CA"*. To reiterate the key points from this earlier document:

- The expectation is that the submission will be weighted 50:50 between pair members. In the unusual circumstance that members of the pair do not contribute 50:50, you have the opportunity to indicate a different ratio you have both agreed on the cover page of the EBART submission, up to a maximum divergence of 60:40.
- Pair programming is categorically **not** two developers working separately on two different machines. Side-by-side communication developing on a single machine is a key aspect of the approach (in person or online).
- **The module leader reserves the right to split pairs where one student is not engaging with the coursework. The coordinator also reserves the right to assign non-contributing students a mark of 0**. In the rare situation that you are paired with a student who is not contributing (e.g. not replying to emails and/or not meeting up for pair-programming sessions) you must inform Dr. Pacheco of the situation **within one week of release of the CA specification** to facilitate the aforementioned splitting of pairs if necessary. Both parties of a split pair will be assigned an individual variant of the CA (however, if there are multiple pairs in this situation it may be possible to reform pairs consisting of participating students, and of non-participating students). It is not permitted for a student working on the *individual* variant of the CA to collaborate with any other student.

  Given the above process and timelines, please ensure that you arrange to meet virtually and start the work as soon as the CA specification is released to reassure yourself that you are partnered with a student who wants to actively contribute to the coursework. It is an expectation that pairs have *at a minimum* two pair-programming sessions in the first week of release of the CA.
- It is not permitted for students working on the pair programming assignment to collaborate on the assignment with any other student *apart* from their named partner. Those doing so will be subject to academic misconduct regulations and penalties. Please refer to the undergraduate handbook for details on collusion, plagiarism, etc (see web link on coversheet of this document).

2 Assignment – Coding (100 marks)

The University of Knowledge (UoK) is extremely pleased with the partnership with their students in the development of their own systems. The experience has been shown to be positive to all. Now, the UoK is planning to create its own social media platform. In order to balance the workload among the students, the UoK divided the system development in two groups, front-end and back-end. This assignment is based around the development of a back-end Java package for the UoK. It has already determined the functionality required of the system, and has provided an *interface* to be used between the back-end you develop, and the front-end the other group is developing. UoK staff should be able to simply compile in the jar file of the package you develop, with the rest of their system, to result in a fully functioning solution.

2.1 The problem

Social media platforms are increasingly growing in popularity and they are among the most used apps and accessed websites. Despite being initially designed to promote online social interactions, they are more than connecting people, they have been used to promote disinformation campaigns, and they are a huge marketplace filled with advertisements and algorithms pushing you to consume and buy.

The UoK plans to build a very simple social media platform, free of adds and disinformation, focusing on posting short messages (text only). The users, i.e., accounts registered in the system, will be able to post original messages and comments, which can also be endorsed by endorsement posts.

Accounts have a unique numerical identifier, but also a unique string handle to be more easily identified throughout the system. They can have a description field to add personal information they want to share. Posts (original, comments, and endorsements) have a unique numerical identifier and contain a message with up to 100-characters. The post ID is a sequential number such that its ordering is a proxy for a post's chronology. They are always associated with an author, i.e., the account who posted it. To allow the creation of meaningful conversation trees, posts must keep track of the list of endorsements and comments they received. Endorsements and comments are also categorized as posts, but with special features. For instance, comments always have to point to another post (original or comment). Endorsements automatically replicate the endorsed message and also refer to original or comment posts. Endorsements are not endorsed or commented. The system should provide basic analytics such as the most popular post and the most popular account.

The UoK wants the back-end of the new social media platform to be compatible with the front-end which is being developed by another group, as such they have already designed a Java interface for the new system, which their front-end application will use. You are to develop a class that implements this interface, and also develop the necessary additional supporting classes in the Java package called `socialmedia`. The operational correctness of the back-end system will be tested through this provided interface on submission.

Your task is to design and write the additional package members to complete the `socialmedia` package. You will need to design and write a class that implements the `SocialMediaPlatform` interface. Individuals not paired should implement a simplified version defined by the `MiniSocialMediaPlatform` interface. Both interfaces are available on the ECM1410 ELE assessment page, as well as at the end of this document.

This implementor class **must** be a public class called `SocialMedia`. If it is not, then the front-end system will be unable to compile with your back-end solution, and the operational component of your mark will be **0**. You will need to also write any other package members you deem appropriate to support this class and its functionality. All classes developed must reside in the `socialmedia` package. Alongside the interface, I have provided in the package a set of exception classes which the interface requires.

2.2 Development considerations

The following points should be noted:

- Your source code should include appropriate comments and assertions.
- When a post is removed from the platform, all of its endorsements should be also removed. Comments to the removed post will remain in the platform, but as "orphans", i.e., the parent link should not exist anymore.
- When an account is removed from the platform, all of its posts (original, comments, and endorsements) should be removed as well.

You will not need to submit an executable application (i.e. you do not need to submit a class with a `public static void main` method which uses the `socialmedia` package). This notwithstanding, it is strongly advised that you do write an application to test that your package conforms to the requirements prior to submission. We have provided a skeleton test class (`SocialMediaPlatformTestApp.java`) in the ELE assessment page.

Apart from the classes you develop yourself, or that you have been given as part of the CA, you must only use those available in the Java built-in packages (`java.*`). The use of any other packages will result in a **penalty of 10 marks**.

Your should consult the `SocialMediaPlatform` and the `MiniSocialMediaPlatform` interfaces for a more detailed description of expected behaviour of a class which implements that interface (provided in the JavaDoc).

3 Submission

The CA requires electronic submission to the EBART online system. Upload your file (see details below) by midday on the due date specified on the cover page of this document. The paths mentioned below (folder structure) should be all lowercase, but the files within the folders should follow the Java naming convention.

You must to create a folder named `ecm1410_coursework` and add four child folders:

- `src` – to keep the source code (*.java files).
- `bin` – to keep the bytecode (*.class files).
- `doc` – to save the Javadoc from your code.
- `res` – to add any other resource you need. This folder has to include:

  1. **Cover Page** – a cover page which details *both* of the student numbers of the corresponding pair. If, unusually, you have agreed a split which is not 50:50, this page should also detail how you would like the final mark to be allocated to the pair, based upon your agreed input. This cannot exceed 60:40. As the submission is anonymous, again please use your student numbers. This page should have a development log, which includes date, time and duration of pair programming sessions, including which role(s) each developer took in these sessions, with each log entry identified by both members using your student numbers to ensure anonymous marking. If you are working solo, you are still required to submit the cover page with your student number, but you don't need to add the development log. This file should be named `cover_sheet.pdf`. We have added an example of cover page in our ELE page[1].

     Failure to submit any coversheet detailing both members student numbers and development log will incur a **penalty of 5 marks**.
  2. **Printout** – a PDF file with a printout of all source files written by you (i.e., **not including** the classes and interface that have been provided to you by the ECM1410 team as part of the coursework), including **the line numbers** for each file independently. This PDF should be named `printout.pdf`.

You need to generate a **Jar Package** with a copy of your **full** finished package, named `socialmedia.jar`. The jar file must include: (i) the bytecode (*.class*), (ii) source files (*.java*) of your submitted package, including the `SocialMediaPlatform` interface and all the exception classes provided to you as part of the CA, (iii) the Javadoc for the package, and (iv) the cover page and printout. I.e. it should be a complete self-contained package, that my test program can interact with, via your `socialmedia.SocialMedia` class.

Assuming you followed the above-mentioned folder structure, here is an example[2] of how you can generate the expected jar package:

```
>> ls
bin     doc     res     src
>> javac -d bin/ src/socialmedia/*.java
>> jar cvf socialmedia.jar -C bin .
(listing files added, omitted output)
>> jar uvf socialmedia.jar -C src .
(listing files added, omitted output)
>> jar uvf socialmedia.jar doc
(listing files added, omitted output)
>> jar uvf socialmedia.jar res
```

You can test your package using the `SocialMediaPlatformTestApp.java` class provided with the interface and other files.

```
>> javac -cp .:socialmedia.jar SocialMediaPlatformTestApp.java
>> java -cp .:socialmedia.jar SocialMediaPlatformTestApp
The system compiled and started the execution...
```

[1] Inside the released zip file, go to `res/cover_sheet_example.pdf`. This example highlights *what* information the cover page needs to include. You don't need to follow this exact format.

[2] This commands were tested in a Mac machine but they should be the same for Linux machines. If you have Windows you can either: use the native *Windows PowerShell* app to use the same commands; or replace the command `ls` by `dir`, the forward slashes "/" by backslashes "\", and colon ":" by semi-colon ";" if you use the *cmd* app.

Finally, you may want to *check* the files within the jar file:

```
>> jar -tf socialmedia.jar
```

or to *extract* its content :

```
>>jar -xvf  socialmedia.jar
```

There will be two types of submission:

- **Pair Member 1 or Solo** – you need to submit the full jar package if you are the first member of a pair or if you are working alone. Pairs and solo implementing `SocialMediaPlatform.java` and `MiniSocialMediaPlatform.java`, respectively.
- **Pair Member 2** - if you are the second member of a pair, you only need to submit the cover page (`cover_sheet.pdf`).

Since EBART does not accept the submission of .jar files, you need to zip your jar to submit.

4 Advice

1. Do not jump straight into the coding: take time to consider the design of your solution first. Think about the objects that you will use, the data they will contain, what the methods they should provide are (in addition to those mandated via the interface), how they relate to one another, etc. Once you are happy with your design, then start programming. Don't be afraid to reassess your design as you go through, but check on the implications of making a changes on all the other objects in your system that use the changed part (this is where one of the strengths of pair programming will come in, in being able to discuss the design implications).
2. Check your objects behave as you intend — use a testing application and use assertions.
3. Slowly fill out functionality — it is far better to submit a solution that supplies most but not all of the required operations correctly, rather than one that doesn't provide any/doesn't compile, as a submission which does not provide any correct functionality at all will get a 0 for the operation criteria. Start off with a `SocialMedia` class that compiles and slowly (incrementally) add functionality. I have provided a class that implements `SocialMediaPlatform` on ELE (called `BadSocialMedia`) that does just this — it compiles, but provides none of the correct functionality. The solo version is called `BadMiniSocialMedia`.
4. Keep copies of your working code. If the worst happens and you had a version that worked on 50% of the operations and you've made changes that seem to have broken everything, it is useful to be able to 'roll-back' to the earlier version and try again.
5. **Do not** change the interface and classes that I have provided for you. If you change them, the markers will not be able to compile my codebase with your submission, and you will receive an 'Operation' component mark of **0**, as the interface will not be able to connect to the front-end of the system.

5 Marking Criteria

This assessment will be marked using the following criteria.

Criterion	**Description**	**Marks Available**
Comments & annotations.	The degree of quality and appropriateness of documentation comments, code comments and annotations.	/5
Java conventions.	The degree of adherence to Java naming conventions and formatting. See lecture notes and e.g. `https://google.github.io/styleguide/javaguide.html`	/5
Operation.	The degree to which the provided `SocialMedia` class operates as required, as supported by the package members. Submission of a jar file that cannot be compiled in with the test code (due to e.g. the interface definition being changed, require package members missing, etc.) will receive an operation mark of 0.	/50
OO design.	The degree to which the code is object-oriented, well structured and presented, with a coherent design and clear and appropriate management of object states, with well encapsulated objects, appropriate distribution of computational load across objects and appropriate use of types and assertions.	/40
Penalty.	Use of non-permitted packages.	−10
Penalty.	Non-submission of coversheet with pair membership details and development log.	−5
Penalty.	Non-submission of code printout.	−5

6 MiniSocialMediaPlatform.java

```
package socialmedia;

import java.io.IOException;
import java.io.Serializable;

/**
 * MiniSocialMediaPlatform interface. The no-argument constructor of a class
 * implementing this interface should initialise the MiniSocialMediaPlatform as
 * an empty platform with no initial accounts nor posts within it. For Solo
 * submissions ONLY.
 *
 * @author Diogo Pacheco
 * @version 1.0
 *
 */
public interface MiniSocialMediaPlatform extends Serializable {

   // Account-related methods ***************************************

   /**
    * The method creates an account in the platform with the given handle.
    * <p>
    * The state of this SocialMediaPlatform must be be unchanged if any exceptions
    * are thrown.
    *
    * @param handle account's handle.
    * @throws IllegalHandleException if the handle already exists in the platform.
    * @throws InvalidHandleException if the new handle is empty, has more than 30
    *                                characters, or has white spaces.
    * @return the ID of the created account.
    *
    */
   int createAccount(String handle) throws IllegalHandleException, InvalidHandleException;

   /**
    * The method removes the account with the corresponding ID from the platform.
    * When an account is removed, all of their posts and likes should also be
    * removed.
    * <p>
    * The state of this SocialMediaPlatform must be be unchanged if any exceptions
    * are thrown.
    *
    * @param id ID of the account.
    * @throws AccountIDNotRecognisedException if the ID does not match to any
    *                                         account in the system.
    */
   void removeAccount(int id) throws AccountIDNotRecognisedException;

   /**
    * The method replaces the oldHandle of an account by the newHandle.
    * <p>
    * The state of this SocialMediaPlatform must be be unchanged if any exceptions
```

```
     * are thrown.
     *
     * @param oldHandle account's old handle.
     * @param newHandle account's new handle.
     * @throws HandleNotRecognisedException if the old handle does not match to any
     *                                      account in the system.
     * @throws IllegalHandleException       if the new handle already exists in the
     *                                      platform.
     * @throws InvalidHandleException       if the new handle is empty, has more
     *                                      than 30 characters, or has white spaces.
     */
    void changeAccountHandle(String oldHandle, String newHandle)
          throws HandleNotRecognisedException, IllegalHandleException, InvalidHandleException;

    /**
     * The method creates a formatted string summarising the stats of the account
     * identified by the given handle. The template should be:
     *
     * <pre>
     * ID: [account ID]
     * Handle: [account handle]
     * Description: [account description]
     * Post count: [total number of posts, including endorsements and replies]
     * Endorse count: [sum of endorsements received by each post of this account]
     * </pre>
     *
     * @param handle handle to identify the account.
     * @return the account formatted summary.
     * @throws HandleNotRecognisedException if the handle does not match to any
     *                                      account in the system.
     */
    String showAccount(String handle) throws HandleNotRecognisedException;

    // End Account-related methods ****************************************

    // Post-related methods ***********************************

    /**
     * The method creates a post for the account identified by the given handle with
     * the following message.
     * <p>
     * The state of this SocialMediaPlatform must be be unchanged if any exceptions
     * are thrown.
     *
     * @param handle handle to identify the account.
     * @param message post message.
     * @throws HandleNotRecognisedException if the handle does not match to any
     *                                      account in the system.
     * @throws InvalidPostException         if the message is empty or has more than
     *                                      100 characters.
     * @return the sequential ID of the created post.
     */
    int createPost(String handle, String message) throws HandleNotRecognisedException, InvalidPostException;

    /**
```

```
 * The method creates an endorsement post of an existing post, similar to a
 * retweet on Twitter. An endorsement post is a special post. It contains a
 * reference to the endorsed post and its message is formatted as:
 * <p>
 * <code>"EP@" + [endorsed account handle] + ": " + [endorsed message]</code>
 * <p>
 * The state of this SocialMediaPlatform must be be unchanged if any exceptions
 * are thrown.
 *
 * @param handle of the account endorsing a post.
 * @param id    of the post being endorsed.
 * @return the sequential ID of the created post.
 * @throws HandleNotRecognisedException if the handle does not match to any
 *                                      account in the system.
 * @throws PostIDNotRecognisedException if the ID does not match to any post in
 *                                      the system.
 * @throws NotActionablePostException if the ID refers to a endorsement post.
 *                                      Endorsement posts are not endorsable.
 *                                      Endorsements are not transitive. For
 *                                      instance, if post A is endorsed by post
 *                                      B, and an account wants to endorse B, in
 *                                      fact, the endorsement must refers to A.
 */
int endorsePost(String handle, int id)
     throws HandleNotRecognisedException, PostIDNotRecognisedException, NotActionablePostException;

/**
 * The method creates a comment post referring to an existing post, similarly to
 * a reply on Twitter. A comment post is a special post. It contains a reference
 * to the post being commented upon.
 * <p>
 * The state of this SocialMediaPlatform must be be unchanged if any exceptions
 * are thrown.
 *
 * @param handle of the account commenting a post.
 * @param id    of the post being commented.
 * @param message the comment post message.
 * @return the sequential ID of the created post.
 * @throws HandleNotRecognisedException if the handle does not match to any
 *                                      account in the system.
 * @throws PostIDNotRecognisedException if the ID does not match to any post in
 *                                      the system.
 * @throws NotActionablePostException if the ID refers to a endorsement post.
 *                                      Endorsement posts are not endorsable.
 *                                      Endorsements cannot be commented. For
 *                                      instance, if post A is endorsed by post
 *                                      B, and an account wants to comment B, in
 *                                      fact, the comment must refers to A.
 * @throws InvalidPostException     if the comment message is empty or has
 *                                      more than 100 characters.
 */
int commentPost(String handle, int id, String message) throws HandleNotRecognisedException,
     PostIDNotRecognisedException, NotActionablePostException, InvalidPostException;

/**
```

```
 * The method removes the post from the platform. When a post is removed, all
 * its endorsements should be removed as well. All replies to this post should
 * be updated by replacing the reference to this post by a generic empty post.
 * <p>
 * The generic empty post message should be "The original content was removed
 * from the system and is no longer available.". This empty post is just a
 * replacement placeholder for the post which a reply refers to. Empty posts
 * should not be linked to any account and cannot be acted upon, i.e., it cannot
 * be available for endorsements or replies.
 * <p>
 * The state of this SocialMediaPlatform must be be unchanged if any exceptions
 * are thrown.
 *
 * @param id ID of post to be removed.
 * @throws PostIDNotRecognisedException if the ID does not match to any post in
 *                                      the system.
 */
void deletePost(int id) throws PostIDNotRecognisedException;

/**
 * The method generates a formated string containing the details of a single
 * post. The format is as follows:
 *
 * <pre>
 * ID: [post ID]
 * Account: [account handle]
 * No. endorsements: [number of endorsements received by the post] | No. comments: [number of comments
     received by the post]
 * [post message]
 * </pre>
 *
 * @param id of the post to be shown.
 * @return a formatted string containing post's details.
 * @throws PostIDNotRecognisedException if the ID does not match to any post in
 *                                      the system.
 */
String showIndividualPost(int id) throws PostIDNotRecognisedException;

/**
 * The method builds a StringBuilder showing the details of the current post and
 * all its children posts. The format is as follows (you can use tabs or spaces to represent
     indentation):
 *
 * <pre>
 * {@link #showIndividualPost(int) showIndividualPost(id)}
 * |
 * [for reply: replies to the post sorted by ID]
 * | > {@link #showIndividualPost(int) showIndividualPost(reply)}
 * </pre>
 *
 * See an example:
 *
 * <pre>
 * ID: 1
 * Account: user1
```

```
* No. endorsements: 2 | No. comments: 3
* I like examples.
* |
* | > ID: 3
*     Account: user2
*     No. endorsements: 0 | No. comments: 1
*     No more than me...
*     |
*     | > ID: 5
*         Account: user1
*         No. endorsements: 0 | No. comments: 1
*         I can prove!
*         |
*         | > ID: 6
*             Account: user2
*             No. endorsements: 0 | No. comments: 0
*             prove it
* | > ID: 4
*     Account: user3
*     No. endorsements: 4 | No. comments: 0
*     Can't you do better than this?
*
* | > ID: 7
*     Account: user5
*     No. endorsements: 0 | No. comments: 1
*     where is the example?
*     |
*     | > ID: 10
*         Account: user1
*         No. endorsements: 0 | No. comments: 0
*         This is the example!
* </pre>
*
* Continuing with the example, if the method is called for post ID=5
* ({@code showIndividualPost(5)}), the return would be:
*
* <pre>
* ID: 5
* Account: user1
* No. endorsements: 0 | No. comments: 1
* I can prove!
* |
* | > ID: 6
*     Account: user2
*     No. endorsements: 0 | No. comments: 0
*     prove it
* </pre>
*
* @param id of the post to be shown.
* @return a formatted StringBuilder containing the details of the post and its
*         children.
* @throws PostIDNotRecognisedException if the ID does not match to any post in
*                                      the system.
* @throws NotActionablePostException if the ID refers to an endorsement post.
*                                    Endorsement posts do not have children
```

```
     *                                since they are not endorsable nor
     *                                commented.
     */
    StringBuilder showPostChildrenDetails(int id) throws PostIDNotRecognisedException,
        NotActionablePostException;

    // End Post-related methods ****************************************

    // Analytics-related methods ***************************************

    /**
     * This method identifies and returns the post with the most number of
     * endorsements, a.k.a. the most popular post.
     *
     * @return the ID of the most popular post.
     */
    int getMostEndorsedPost();

    /**
     * This method identifies and returns the account with the most number of
     * endorsements, a.k.a. the most popular account.
     *
     * @return the ID of the most popular account.
     */
    int getMostEndorsedAccount();

    // End Analytics-related methods ***********************************

    // Management-related methods **************************************

    /**
     * Method empties this SocialMediaPlatform of its contents and resets all
     * internal counters.
     */
    void erasePlatform();

    /**
     * Method saves this SocialMediaPlatform's contents into a serialised file, with
     * the filename given in the argument.
     *
     * @param filename location of the file to be saved
     * @throws IOException if there is a problem experienced when trying to save the
     *                     store contents to the file
     */
    void savePlatform(String filename) throws IOException;

    /**
     * Method should load and replace this SocialMediaPlatform's contents with the
     * serialised contents stored in the file given in the argument.
     * <p>
     * The state of this SocialMediaPlatform's must be be unchanged if any
     * exceptions are thrown.
     *
     * @param filename location of the file to be loaded
     * @throws IOException         if there is a problem experienced when trying
```

```
     *                              to load the store contents from the file
     * @throws ClassNotFoundException if required class files cannot be found when
     *                              loading
     */
    void loadPlatform(String filename) throws IOException, ClassNotFoundException;

    // End Management-related methods ***************************************

}
```

7 SocialMediaPlatform.java

```
package socialmedia;

/**
 * SocialMediaPlatform interface. This interface is a more elaborated version of
 * the MiniSocialMediaPlatform. The no-argument constructor of a class
 * implementing this interface should initialise the SocialMediaPlatform as an
 * empty platform with no initial accounts nor posts within it. For Pair
 * submissions.
 *
 * @author Diogo Pacheco
 * @version 1.0
 *
 */
public interface SocialMediaPlatform extends MiniSocialMediaPlatform {

    // Account-related methods ***************************************

    /**
     * The method creates an account in the platform with the given handle and
     * description.
     * <p>
     * The state of this SocialMediaPlatform must be be unchanged if any exceptions
     * are thrown.
     *
     * @param handle    account's handle.
     * @param description account's description.
     * @throws IllegalHandleException if the handle already exists in the platform.
     * @throws InvalidHandleException if the new handle is empty, has more than 30
     *                              characters, or has white spaces.
     * @return the ID of the created account.
     */
    int createAccount(String handle, String description) throws IllegalHandleException,
        InvalidHandleException;

    /**
     * The method removes the account with the corresponding handle from the
     * platform. When an account is removed, all of their posts and likes should
     * also be removed.
     * <p>
     * The state of this SocialMediaPlatform must be be unchanged if any exceptions
     * are thrown.
     *
```

```
 * @param handle account's handle.
 * @throws HandleNotRecognisedException if the handle does not match to any
 *                                      account in the system.
 */
void removeAccount(String handle) throws HandleNotRecognisedException;

/**
 * The method updates the description of the account with the respective handle.
 * <p>
 * The state of this SocialMediaPlatform must be be unchanged if any exceptions
 * are thrown.
 *
 * @param handle    handle to identify the account.
 * @param description new text for description.
 * @throws HandleNotRecognisedException if the handle does not match to any
 *                                      account in the system.
 */
void updateAccountDescription(String handle, String description) throws HandleNotRecognisedException;

// End Post-related methods ****************************************

// Analytics-related methods ***************************************

/**
 * This method returns the current total number of accounts present in the
 * platform. Note, this is NOT the total number of accounts ever created since
 * the current total should discount deletions.
 *
 * @return the total number of accounts in the platform.
 */
int getNumberOfAccounts();

/**
 * This method returns the current total number of original posts (i.e.,
 * disregarding endorsements and comments) present in the platform. Note, this
 * is NOT the total number of posts ever created since the current total should
 * discount deletions.
 *
 * @return the total number of original posts in the platform.
 */
int getTotalOriginalPosts();

/**
 * This method returns the current total number of endorsement posts present in
 * the platform. Note, this is NOT the total number of endorsements ever created
 * since the current total should discount deletions.
 *
 * @return the total number of endorsement posts in the platform.
 */
int getTotalEndorsmentPosts();

/**
 * This method returns the current total number of comments posts present in the
 * platform. Note, this is NOT the total number of comments ever created since
 * the current total should discount deletions.
```

```
     *
     * @return the total number of comments posts in the platform.
     */
    int getTotalCommentPosts();

    // End Management-related methods ***************************************

}
```