```java
package socialmedia.internal;

/**
 * This class represents a post in the platform.
 * It stores information that post need to have to work on the platform.
 */

public class Post {
    // Post's unique ID
    private int postid;

    // Post's message.
    private String message;

    // Post's creator.
    //private Account account;

    // Handle of the post creator.
    private String handle;

    // True if post is original | False if post is endorsement/comment.
    private boolean original;

    // True if post is a comment | False if post is original post/enodresement.
    private boolean comment;

    // Number of times post has been endorsed.
    private int numberOfEndorsements;

    // Number of times post has been commented.
    private int numberOfComments;

    // Assigns fatherLink (ID of original post) if post is endorsement/comment.
    public int fatherLink;

    // Counts number of posts created in the platform and assigns a chronological ID to the
```

```
posts respectively.
    private static int postIdCounter = 1;

    private String endorsingHandle;



    /**
     * Creates a new post with the given handle and message.
     * @param handle The user's handle and message.
     */
    public Post(String handle, String message) {
        this.handle = handle;
        this.message = message;
        this.postid = generatePostId(); // this is to generate a unique ID
        this.original = true;

    }

    /**
     * Get endorsing handle
     * @return Endorsing handle
     */
    public String getEndorsingHandle() {
        return endorsingHandle;
    }

    /**
     * This generate an id for the post
     * @return Post ID (postIdCounter)
     */

    public static synchronized int generatePostId() {
        return postIdCounter++;
    }

    /**
     * Get a posts's fatherlink if there is one.
```

```
     * @return father link (original post id)
     */
    public int getFatherLink() {
        return fatherLink;
    }

    /**
     * Get post's message.
     * @return Post's message.
     */
    public String getMessage() {
        return message;
    }
    /**
     * Get post's handle (username).
     * @return Post's handle (username).
     */
    public String getHandle(){
        return handle;
    }

    /**
     * Get post's ID.
     * @return Post's ID.
     */
    public int getPostid() {
        return postid;
    }

    /**
     * Set message for a post.
     * @param a message
     */
    public void setMessage(String message) {
        this.message = message;
    }

    /**
     * Get original (Check if post is original)
     * @return True(Original)/False(Not original).
     */
```

```
    public boolean getOriginal() {
        return original;
    }

    /**
     * Get original of a comment (Check if post is a comment)
     * @return True(Post is a comment)/False(Post is not a comment).
     */
    public boolean getCommentOriginal() {
        return comment;
    }

    /**
     * Change a state of post original (Make it original)
     */
    public void changeOriginalToTrue() {
        this.original = true;
        fatherLink = 0;
    }

    /**
     * Make a post NOT original (endorsement or comment)
     * @param Post ID of original post that has been commented or endorsed.
     */
    public void changeOriginalToFalse(int postId) {
        this.original = false;
        fatherLink = postId;
    }

    /**
     * Change post to a comment.
     * @param Post ID of original post that has been commented.
     */
    public void changeCommentToTrue(int postId) {
```

```
        this.comment = true;
        fatherLink = postId;
    }

    /**
     * Change comment to a post
     */
    public void changeCommentToFalse() {
        this.comment = false;
        fatherLink = 0;
    }

    /**
     * Get number of times post has been endorsed
     * @return number of endorsements.
     */
    public int getNumberOfEndorsements() {
        return numberOfEndorsements;
    }

    /**
     * Method to return the number of comments associated with a post.
     * @param Number of comments.
     */
    public int getNumberOfComments() {
        return numberOfComments;
    }

    /**
     * Method to add a new endorsement and update the number of endorsements.
     * @return Number of endorsements.
     */
    public int addNumberOfEndorsements() {
        numberOfEndorsements += 1;
        return numberOfEndorsements;
    }

    /**
     * Method to add a new comment and update the
```

```
number of comments.
     * @return Number of comments.
     */
    public int addNumberOfComments() {
        numberOfComments +=1;
        return numberOfComments;
    }


}
```

```java
package socialmedia.internal;
import java.util.UUID;

public class Account {
    // Handle (username) of an account.
    private String handle;
    // Automatically generated user id (sequential number).
    private int userid;
    // Description of an account.
    private String description;
    // Number of times account's posts have been endorsed.
    private int numberOfAccountEndorsements;
    // Number of times account's posts have been commented.
    private int numberOfAccountComments;
    //Number of account's posts.
    private int numberOfAccountPosts;


    /**
     * Creates a new account with the given handle.
     * @param handle The user's handle
     */
    public Account(String handle) {
        this.handle = handle;
        this.userid = generateUserid();
    }

   /**
     * Generates user ID.
     * @return The User ID
     */
    private int generateUserid() {
        UUID uuid = UUID.randomUUID();
        long uuidlong = uuid.getMostSignificantBits() & Long.MAX_VALUE;
        int userid = (int) (uuidlong % Integer.MAX_VALUE);
        return userid;
```

```
    }

    /**
     * Return Account's handle(username).
     * @return handle (username)
     */
    public String getHandle() {
        return handle;
    }

    /**
     * Return Account's user ID.
     * @return the User ID
     */
    public int getUserid() {
        return userid;
    }

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

    /**
     * Set description of the Account.
     * @param String description to set.
     */
    public void setDescription(String description
) {
        this.description = description;
    }

    /**
     * Change handle of the Account.
     * @param String new handle.
     */
    public void changeHandle(String new_handle) {
        this.handle = new_handle;
```

```
    }

    /**
     * Get number of Account's posts.
     * @return Number of Posts.
     */
    public int getNumberOfPosts() {
        return numberOfAccountPosts;
    }

    /**
     * //Adds 1 to a number of Account's posts.
     */
    public void addNumberOfPosts() {
        numberOfAccountPosts++;
    }

    /**
     * // Adds 1 number of endorsements to Account based on case that Account's post has been endorsed.
     */
    public void addNumberOfAccountEndorsements() {
        this.numberOfAccountEndorsements += 1;
    }

    /**
     * // Get number of times Account's posts have been endorsed.
     * @return Number of Endorsements.
     */
    public int getNumberOfAccountEndorsements() {
        return numberOfAccountEndorsements;
    }

     /**
     * // Adds 1 number of comments if Account's posts are commented.
     */
    public void addNumberOfAccountComments() {
        this.numberOfAccountComments += 1;
```

```
    }

    /**
     * // Get number of comments.
     * @return Number of Comments
     */
    public int getNumberOfAccountComments() {
        return numberOfAccountComments;
    }

    public StringBuilder showAccount(StringBuilder builder) {
        return builder.append("ID: ").append(getUserid()).append("\n")
                .append("Handle: ").append(getHandle()).append("\n")
                .append("Description: ").append(getDescription()).append("\n")
                .append("Post Count: ").append(getNumberOfPosts()).append("\n")
                .append("Endorse Count: ").append(getNumberOfAccountEndorsements()).append("\n");

    }

}
```

```java
package socialmedia.internal;

public class Comment extends Post {

    public Comment(String handle, String message, int fatherLink) {
        super(handle, message);
        this.fatherLink = fatherLink;
    }

    public int getFatherLink() {
        return fatherLink;
    }

}
```

```java
package socialmedia.internal;



public class Endorsement extends Post {


    public Endorsement(String handle, String message, int fatherLink) {
        super(handle, message);
        this.fatherLink = fatherLink;

    }

    public int getFatherLink() {
        return fatherLink;
    }
}

```

```java
package socialmedia;

import socialmedia.internal.*;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.*;



public class SocialMedia implements MiniSocialMediaPlatform {

    public static HashMap<String, Account> accounts = new HashMap<String, Account>();
    public static HashMap<String, List<Post>> posts = new HashMap<String, List<Post>>();

    private int postCount;




    /**
    * Creates a new account with the given handle and returns the user ID of the newly created account.
    *
    * @param handle The handle to be associated with the new account
    * @return The user ID of the newly created account
    * @throws IllegalHandleException if an account with the same handle already exists in the platform
    * @throws InvalidHandleException if the handle is not a single word or contains more than 30 characters
    */
```

```java

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

        // Check if the handle already exists in the accounts map, and throw an exception if it does
        checkHandleException(handle);
        // Check if the handle is a single word and has a length of less than 30 characters
        checkHandle(handle);
        // If the handle passes the validation, create a new account with the given handle
        Account account = new Account(handle);
        // Add the created account to the accounts map using the handle as the key
        accounts.put(handle, account);
        // Return the user ID of the newly created account
        return account.getUserid();
    }

    /**
    Removes an account from Accounts-Hashmap using the account's id.
    @param id The user ID associated with the account to be removed
    @throws AccountIDNotRecognisedException if an account with the given ID does not exist in the platform
    */
    public void removeAccount(int id) throws AccountIDNotRecognisedException { //GOOD
        Account accountToRemove = null;                // Creates an Account object
        for (Account account: accounts.values()) {     // Loop through values in Accounts-Hashmap
            if (account.getUserid() == id) {
                System.out.println(account.getHandle());     // Checks if the id exists in Accounts-Hashmap
```

```
                accountToRemove = account;
                account = null;
            accounts.remove(accountToRemove.getHandle());  // Removes the account with given id from Accounts-Hashmap
            posts.remove(accountToRemove.getHandle());
            break;
            }

        }
        if (accountToRemove == null) {
            throw new AccountIDNotRecognisedException("Account with id " + id + " does not exist");

        }
    }

    /**

    Changes the handle of an account in the Accounts-Hashmap.
    @param handle The current handle of the account
    @param new_handle The new handle to be associated with the account
    @throws HandleNotRecognisedException if an account with the current handle does not exist in the platform
    */

    public void changeAccountHandle(String handle, String new_handle) throws HandleNotRecognisedException {
        checkHandleRecognition(handle);
        Account account = accounts.get(handle);
        account.changeHandle(new_handle); // changes the handle of the account
        accounts.put(new_handle, account); //// updates the Accounts-Hashmap with the new handle
        accounts.remove(handle); //// removes the
```

```
old handle from the Accounts-Hashmap

        List<Post> postList = posts.get(handle);
        posts.remove(handle);
        posts.put(new_handle, postList);

        System.out.println("Original handle: " + handle + " has been changed to " + new_handle);

    }



    /**
    * Shows the details of an account in the Accounts-Hashmap.
    *
    * @param handle The handle of the account
    * @throws HandleNotRecognisedException if the handle is not recognized in the Accounts-Hashmap
    */
    public String showAccount(String handle) throws HandleNotRecognisedException {
        // Checks if the handle exists in the Accounts-Hashmap
        checkHandleRecognition(handle);

        Account accountReturned = null;
        // Gets the account with the specified handle
        Account account = accounts.get(handle);
        // Prints out the details of the account

        accountReturned = account;

        return accountReturned.showAccount(new StringBuilder()).toString();
    }


    // End Account-related methods
```

```
 ****************************************

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

    /**
     * Creates a new post for the specified handle with the given message.
     * @param handle the handle of the account to create the post for
     * @param message the message to be included in the post
     * @throws HandleNotRecognisedException if the account with the given handle does not exist
     * @throws InvalidPostException if the message is empty or longer than 100 characters
     * @return the ID of the created post
    */
    public int createPost(String handle, String message) throws HandleNotRecognisedException, InvalidPostException { //GOOD
        // Check if the message is empty or too long, and throw an exception
        checkMessage(message);
        // Check if the account with the given handle exists
        checkHandleRecognition(handle);
        // Create a new post with the given handle and message
        Post post = new Post(handle, message);
        Account account = accounts.get(handle); // Retrieve the account associated with the handle and increment the number of posts for the account
        account.addNumberOfPosts();
        // Add the post to the list of posts for the account
        if (!posts.containsKey(handle)) {
        posts.put(handle, new ArrayList<Post>());
        posts.get(handle).add(post);
        }
        else {
```

```
            posts.get(handle).add(post);
        }
        // Update the account in the map of accounts
        accounts.put(handle, account);
        //return id of the new post
        postCount +=1;
        System.out.println(postCount);
        return post.getPostid();
    }

     /**
    * Creates a new repost or endorsement for the specified post and handle.
    *
    * @param handle the handle of the account making the repost/endorsement
    * @param postId the ID of the post to be reposted/endorsed
    * @throws HandleNotRecognisedException if the account with the given handle does not exist
    * @throws PostIDNotRecognisedException if the post with the given ID does not exist
    * @throws NotActionablePostException if the post cannot be endorsed (i.e. it is a comment, not an original post, or has already been endorsed)
    * @return the ID of the reposted/endorsed post
    */
    public int endorsePost(String handle, int postId) throws HandleNotRecognisedException, PostIDNotRecognisedException, NotActionablePostException {
        checkHandleRecognition(handle);
        checkPostId(postId);
        List<Endorsement> endorsementsToAdd = new ArrayList<>(); ////////

        for (List<Post> postlist : posts.values()) {
            for (Post post: postlist) {
                if (post.getPostid() == postId) {
```

```
                    String originalHandle = post.getHandle();
                    if (post instanceof Endorsement) {
                        throw new NotActionablePostException("Endorsement post cannot be endorsed.");
                    }
                    Endorsement endorsement = new Endorsement(handle, post.getMessage(), postId);

                    endorsementsToAdd.add(endorsement); ///////
                    post.addNumberOfEndorsements();


                    Account endorsingAccount = accounts.get(originalHandle);
                    endorsingAccount.addNumberOfAccountEndorsements();
                    accounts.put(originalHandle, endorsingAccount);

                }
                }
            }

        for (Endorsement endorsement : endorsementsToAdd) {
                if (posts.containsKey(handle)) {
                    posts.get(handle).add(endorsement);
                } else {
                    posts.put(handle, new ArrayList<Post>());
                    posts.get(handle).add(endorsement);
                }
            }
        postCount +=1;
```

```
        return postCount;
    }



    /**
     * This method allows a user to make a comment on an original post.
     * @param handle the handle of the user making the comment
     * @param id the ID of the original post to be commented on
     * @param message the comment message to be posted
     * @return the ID of the new comment post
     * @throws HandleNotRecognisedException if the handle is not recognised
     * @throws PostIDNotRecognisedException if the post ID is not recognised
     * @throws NotActionablePostException if the post cannot be commented on
     * @throws InvalidPostException if the comment message is invalid
     * */
    public int commentPost(String handle, int postId, String message) throws HandleNotRecognisedException, PostIDNotRecognisedException,
    NotActionablePostException, InvalidPostException {
        checkHandleRecognition(handle);
        checkPostId(postId);
        checkMessage(message);
        List<Comment> commentsToAdd = new ArrayList<>(); ////////

        for (List<Post> postlist : posts.values()) {
            for (Post post: postlist) {
                if (post.getPostid() == postId) {
                    String originalHandle = post.
```

```
getHandle();
                    if (post instanceof Endorsement) {
                        throw new NotActionablePostException("Endorsement post cannot be commented.");
                    }
                    Comment comment = new Comment(handle, message, postId);

                    commentsToAdd.add(comment); ///////
                    post.addNumberOfComments();


                    Account commentingAccount = accounts.get(originalHandle);
                    commentingAccount.addNumberOfAccountComments();
                    accounts.put(originalHandle, commentingAccount);

                }
                }
            }

        for (Comment comment : commentsToAdd) {
                if (posts.containsKey(handle)) {
                    posts.get(handle).add(comment);
                } else {
                    posts.put(handle, new ArrayList<Post>());
                    posts.get(handle).add(comment);
                }
            }
        postCount +=1;
        return postCount;
    }

```

```java

    /**
    * This method deletes a post by its ID. Firstly, it checks if the ID exists in the posts hashmap. If it does not exist,
    * it throws a PostIDNotRecognisedException. If it exists, it will remove the post from the hashmap with the remove(id).
    * @param id the ID of the post to be deleted.
    * @throws PostIDNotRecognisedException if the given post ID is not recognised.
    */
    public void deletePost(int id) throws PostIDNotRecognisedException {
        boolean postFound = false;
        for (List<Post> postlist: posts.values()) {
            Iterator<Post> iterator = postlist.iterator();
            while (iterator.hasNext()) {
                Post post = iterator.next();
                if (post.getFatherLink() == id || post.getPostid() == id) {
                    iterator.remove();
                    postFound = true;
                }
            }
        }

        if (!postFound) {
            throw new PostIDNotRecognisedException("The given post ID is not recognised please enter a correct one: " + id);
        }
    }


    /**
     * Returns a StringBuilder object containing the individual details of the post with the specified ID.
```

```
     * @param id the ID of the post to get the details of
     * @return a StringBuilder object containing the individual details of the post
     */
    public String showIndividualPost(int id) throws PostIDNotRecognisedException {
        boolean postFound = false;
        StringBuilder detailsGlobal = new StringBuilder();

        for (List<Post> postlist : posts.values()) {
            for (Post post : postlist) {
                if (post.getPostid() == id) {
                    // If the post is not an original post, throw a NotActionablePostException
                    // the append is a method used to append a value to the current sequence.
                    detailsGlobal.append("ID: " + post.getPostid()).append("\n");
                    detailsGlobal.append("Account : " + post.getHandle()).append("\n");
                    detailsGlobal.append("No. endorsements: ")
                                .append(post.getNumberOfEndorsements()).append(" | ");
                    detailsGlobal.append("No. comments: ")
                                .append(post.getNumberOfComments()).append("\n");
                    detailsGlobal.append("Message : " + post.getMessage()).append("\n");
                    postFound = true;
                }


            }
        }

        if (!postFound) {
```

```
                    throw new PostIDNotRecognisedException("Post ID is not recognised");
                }

        return detailsGlobal.toString();
    }


        /**
     * This method shows the details of a post with the given ID.
     * It will search through all posts in the hashmap to find the post with the matching ID.
     * If the post is an original post, it will return the number of endorsements and comments.
     * If the post is not an original post, a NotActionablePostException will be thrown.
     * @param id the ID of the post to show details for
     * @return a StringBuilder containing the details of the post
     * @throws PostIDNotRecognisedException if the given post ID is not recognised
     */
    public StringBuilder showPostChildrenDetails(int id) throws PostIDNotRecognisedException, NotActionablePostException {
        boolean postFound = false;
        StringBuilder detailsGlobal = new StringBuilder();

        for (List<Post> postlist : posts.values()) {
            for (Post post : postlist) {
                if (post.getPostid() == id) {
                    // If the post is not an original post, throw a NotActionablePostException
                    // the append is a method used to append a value to the current sequence.
                    detailsGlobal.append("ID: " +
```

```
post.getPostid()).append("\n");
                    detailsGlobal.append("Account : " + post.getHandle()).append("\n");
                    detailsGlobal.append("No. endorsements: ")
                            .append(post.getNumberOfEndorsements()).append(" | ");
                    detailsGlobal.append("No. comments: ")
                            .append(post.getNumberOfComments()).append("\n");
                    detailsGlobal.append("Message : " + post.getMessage()).append("\n");
                    postFound = true;
                }

            }
        }

        for (List<Post> postlist : posts.values()) {
            for (Post post : postlist) {

                if (post.getFatherLink() == id && post.getCommentOriginal()) {
                    detailsGlobal.append("| \n");
                    detailsGlobal.append("| ID: " + post.getPostid()).append("\n");
                    detailsGlobal.append("Account : " + post.getHandle()).append("\n");
                    detailsGlobal.append("No. endorsements: ")
                            .append(post.getNumberOfEndorsements()).append(" | ");
                    detailsGlobal.append("No. comments: ")
                            .append(post.getNumberOfComments()).append("\n");
                    detailsGlobal.append("Message : " + post.getMessage()).append("\n");
                }
```

```java


            }
        }



        if (!postFound) {
                throw new PostIDNotRecognisedException("Post ID is not recognised");
        }

        return detailsGlobal;
    }


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

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



    /**
     * This method is used to retrieve the post with the highest number of endorsements. It loops through all posts in the
     * posts map and updates the postId and endorsement count if a post with a higher endorsement count is found. The postId
     * and endorsement count are then returned by the method.
     * @return The number of endorsements for the most endorsed post
     */
    public int getMostEndorsedPost() {
        //A variable for post ID and endorsements should be initialized
        int number = 0;
        int postId = 0;
```

```
        // This is a loop through all post in the posts map
        for (List<Post> postlist: posts.values()) {
            for (Post post: postlist) {
            //Variables are updated if there are more endorsements than there are currently
            if (post.getNumberOfEndorsements() > number) {
                number = post.getNumberOfEndorsements();
                postId = post.getPostid();
            }
        }
    }
        //// this prints the account which post has the most endorsement
        System.out.println("Most endorsed post " + postId + " with " + number + " number of endorsements.");

        return postId;
    }

    /**
     * This method returns the number of endorsements received by the account with the most endorsements.
     * It loops through all the accounts in the accounts map, updating the endorsement number and account handl variables
     * if the current account has more endorsements than the current highest endorsement count.
     * It then returns the highest endorsement count.
     * @return The number of endorsements received by the account with the most endorsements.
     */
    public int getMostEndorsedAccount() {
        //Initiate the endorsement number and
```

```
account handle variables with 0;
        Account mostEndorsedAccount = null;
        int number = 0;
        String accountHandle = "";
        //// This a loop through allthe accounts in the accounts map
        for (Account account: accounts.values()) {
            // Variables should be updated if the current account has more endorsements than it currently has
            if (account.getNumberOfAccountEndorsements() > number) {
                number = account.getNumberOfAccountEndorsements();
                accountHandle = account.getHandle();
                mostEndorsedAccount = account;
            }
        }
        // this prints the account which has the most endorsement
        System.out.println("Most endorsed account " + accountHandle + " with " + number + " number of endorsements.");

        return mostEndorsedAccount.getUserid();
    }


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

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

    public void erasePlatform() {
        accounts.clear();
        posts.clear();
        System.out.println("Platform erased successfully.");
    }
```

```
    //// Method to save the platform to a file
    public void savePlatform(String filename) throws IOException {
        FileOutputStream fos = new FileOutputStream(filename);
        ObjectOutputStream oos = new ObjectOutputStream(fos);
        // Write the accounts and posts HashMaps to the file
        oos.writeObject(accounts);
        oos.writeObject(posts);
        oos.close();
        fos.close();
        System.out.println("Platform saved to file: " + filename);
    }

    public void loadPlatform(String filename) throws IOException, ClassNotFoundException {
        FileInputStream fis = new FileInputStream(filename);
        ObjectInputStream ois = new ObjectInputStream(fis);
      //  accounts = (HashMap<String, Account>) ois.readObject();
      //  posts = (HashMap<String, List<Post>>) ois.readObject();
        ois.close();
        System.out.println("Platform loaded successfully from file: " + filename);
    }

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

    // Exception-related methods ****************************************

    protected static void checkHandle(String handle) throws InvalidHandleException {

```

```
        if (handle.length() > 30) {
            throw new InvalidHandleException("Handle should has than 30 characters.");
        }
        if (handle.matches(".*\\s.*")) {
            throw new InvalidHandleException("Has should NOT have white spaces.");
        }
        if ((handle == null) || (handle.isEmpty())) {
            throw new InvalidHandleException("Handle cannot be empty.");
        }
  }

    protected static void checkHandleException(String handle) throws IllegalHandleException {
        if (accounts.containsKey(handle)) {
            throw new IllegalHandleException("Account with this handle already exists in the system.");
        }
    }

    protected static void checkHandleRecognition(String handle) throws HandleNotRecognisedException {
        if (!accounts.containsKey(handle)) { // Check if the account with the given handle exists, and throw an exception if not
            throw new HandleNotRecognisedException("Account with this handle does not exist");
        }
    }

    protected static void checkMessage(String message) throws InvalidPostException{
        if (message.isEmpty()) {
            throw new InvalidPostException("Message cannot be empty");
        }
```

```
        if (message.length() > 100) {
            throw new InvalidPostException("Message should be less than 100 characters");
        }
    }

    protected static void checkPostId(int id) throws PostIDNotRecognisedException {
       boolean postFound = false;

    for (List<Post> postlist : posts.values()) {
        for (Post post : postlist) {
            if (post.getPostid() == id) {
                postFound = true;
                break; // Break out of the inner loop since we found the post with the given ID
            }
        }
        if (postFound) {
            break; // Break out of the outer loop since we found the post with the given ID
        }
    }

    if (!postFound) {
        throw new PostIDNotRecognisedException("Post with such ID was not found");
    }
}

    protected static void checkActionablePost(Post post) throws NotActionablePostException{

        if (post instanceof Endorsement) {
            throw new NotActionablePostException("Endorsement cannot be endorsed");
        }

    }


```

```

}
```