RELAZIONE PROGETTO JAVA

Breve Descrizione:

Il progetto mira ad emulare e gestire le relazioni di un SocialNetwork, gestendo appunto le relazioni tra i vari User e i Post che vengono creati.

Regole di Funzionamento:

All'interno del SocialNetwork è consentito creare uno o più post, mettere like ad uno o più post e seguire gli altri utenti.
Queste azioni sono limitate dalle seguenti regole:

- Un Utente non può seguire due volte la stessa persona
- Un Utente non può mettere like più di una volta allo stesso post
- Un Utente non può mettere like ad un post pubblicato da se stesso, e seguirsi da solo.

Descrizione delle Classi utilizzate:

Per la realizzazione del progetto ho scelto di creare 3 Classi diverse:

- SocialNetwork → MicroBlog
- User → MyUser
- Post

Ho scelto di implementare la classe User anche se non esplicitamente espressa nel testo, in quanto mi consente di avere più strutture di supporto grazie alle quali è stato possibile scrivere il codice in maniera più elegante, semplice e rapida.

SocialNetwork → MicroBlog:

L'interfaccia SocialNetwork viene implementata dalla classe MicroBlog.
In questo tipo di dato realizzo le funzioni richieste dalla traccia nella 2 parte.
MicroBlog implementa - oltre alla struttura richiesta nella traccia - un'ulteriore struttura, che conterrà tutti i post creati sul social network, e inoltre si avvale di altri metodi non specificati nell'interfaccia, che si occupano di aggiornare la struttura Map<String, Set<String>>.

User → MyUser:

L'interfaccia User viene implementata dalla classe MyUser.
Questa classe si occupa delle azioni che un User può compiere, ovvero mettere like e

seguire un altro utente, e ne gestisce i casi limite e gli errori.
La Classe utilizzerà strutture dati di supporto che servono a memorizzare varie cose, le cui importanti sono:

- una lista che contiene tutti i post realizzati dall'utente
- una lista contenente tutti gli utenti che seguono "this"
- una lista contenente tutti gli utenti seguiti da "this"

Post:

La Classe Post si occupa della realizzazione e della gestione dei post.
Oltre ai campi richiesti dalla traccia al suo interno si può trovare una lista contenente tutti gli utenti che hanno messo like a "this".

Custom Exception:

Per indicare situazioni di errore ho scelto di creare due nuovi tipi di exceptions:

- IllegalAction → è usata per segnalare azioni illegali (mettere like due volte allo stesso post, seguirsi da soli, seguire due volte lo stesso utente).
- InvalidUsername → è usata quando si cerca di creare un Utente con una username non valida (vuota, composta da soli spazi, oppure nulla).

Funzionalità di Report:

Per implementare l'opzione di report (segnalazione di contenuti offensivi), l'idea proposta è quella di introdurre una struttura dati e un metodo alla classe MicroBlog.
La struttura dati sarebbe di tipo ***List<String> ForbiddenWords*** di tipo statico, contenente tutte le parole "vietate" all'interno della rete.
Il metodo che consente ad un utente di segnalare un determinato post avrebbe questa firma: ***public void ReportPost(MyUser user, Post ps)*** i cui parametri sono appunto l'utente che vuole effettuare un report ed il post da segnalare.
Il metodo funzionerebbe cosi: Se il testo del post, passato come parametro, contiene una o più delle parole contenute in *ForbiddenWords,* allora l'utente che ha segnalato sarà aggiunto ad una list di User *"ReportedBy"* facente parte del tipo di dato Post.
Una volta che ReportedBy supera un certo numero n di segnalazioni, il post viene rimosso dalla rete.

Istruzioni per l'esecuzione:

Per eseguire progetto bisogna prima compilare tutti i files con il seguente comando:

javac IllegalAction.java InvalidUsername.java MicroBlog.java MyUser.java Post.java SocialNetwork.java Tests.java User.java

Poi va eseguito il file Tests.java con il seguente comando:
java -ea Tests