Relazione

Metodo di Sviluppo del Progetto

Il progetto e stato sviluppato mediante l'ausilio dell'IDE *IntelliJ IDEA*, senza però utilizzare strumenti esterni di automatizzazione della compilazione o del testing del codice. L'unica eccezione è l'utilizzo del plugin di *code coverage* integrato nell'IDE, utilizzato per essere sicuri di aver coperto con i test tutti i casi presi in considerazione dal codice scritto. Questo tuttavia ovviamente non è più richiesto per alcuna operazione, dopo la scrittura dei test, per cui è possibile compilare ed eseguire il codice senza far uso di alcun tool esterno al JDK.
Per compilare ed eseguire il codice è sufficiente usare i seguenti comandi:

```
cd ./Project/src
javac *
java Main
```

Overview Generale sul Progetto e Scelte di Implementazione

La consegna del progetto richiedeva di implementare un programma Java che rappresentasse uno strumento di gestione e di analisi di una rete sociale denominata *MicroBlog*.

La rappresentazione di questa social network è stata realizzata mediante due classi, `Post` e `SocialNetwork`, che rappresentano rispettivamente un post condiviso da un utente, e la rete sociale nella sua interezza, comprendente una lista di utenti, una lista di post, ed una mappa che associa ad ogni utente un insieme di utenti da questi seguiti. `SocialNetwork` è stata successivamente estesa con `ReportableSocialNetwork` per permettere la segnalazione di contenuti offensivi o comunque che non rispettino le norme di condotta di *MicroBlog*.

È stato scelto di **non** realizzare una classe apposita per rappresentare un **utente**, ma di utilizzare invece una `String` contenente lo username dell'utente, e di lasciar gestire al `SocialNetwork` i controlli su validità dello username al momento della registrazione, di esistenza dello username al momento di pubblicazione di un post, e altri controlli e meccanismi necessari al corretto funzionamento della classe.

È stato scelto di considerare l'azione di **follow** di un utente separata dalla scrittura di un post, e tale azione si effettua mediante la chiamata di un metodo `follow` appartenente alla classe `SocialNetwork`. Un'eventuale estensione del progetto potrebbe considerare la creazione di un metodo `unfollow` per rimuovere un *follow* dalla mappa della rete sociale.

È stato scelto, qualora ritenuto opportuno, di rappresentare metodi come funzioni vacuamente vere, o vacuamente false, a seconda dei casi. Ad esempio, un metodo che controlla se in una lista di post esiste un post con un determinato id, restituirà `false` nel caso in cui il dominio (la lista di post, in questo caso) sia vuoto.

Overview delle Classi Implementate

Post

La classe `Post` è una classe *immutable* che rappresenta un post scritto e condiviso da un utente all'interno della social network.
La sua implementazione comprende 4 campi:

- `author`: Lo username dell'autore del post. È compito della `SocialNetwork` controllare che questo sia uno username valido e appartenente ad un utente registrato nella social network.
- `id`: Un codice identificativo univoco per il post. È realizzato mediante concatenazione del timestamp e dello username dell'autore, il che dovrebbe garantirne l'unicità, poiché non è realistico che uno stesso utente crei più di un post nello stesso millisecondo. Nel caso questa diventi una possibilità è possibile aggiungere un elemento randomico o controlli al momento della creazione del `Post` per fare in modo che l'`id` risulti comunque univoco.
- `text`: Il testo del post creato dall'utente. Gli unici controlli effettuati al momento della creazione sono sulla presenza di un valore e sulla lunghezza del testo, limitata a 140 caratteri come da specifica. Per questo motivo, in caso di un eventuale salvataggio su database SQL o altra tecnologia in cui i dati devono essere controllati, tali controlli vanno effettuati al di fuori del costruttore della classe.
- `timestamp`: Un timestamp in formato UNIX che indica il momento in cui il `Post` è stato creato.

SocialNetwork

La classe `SocialNetwork` rappresenta la vera e propria rete sociale.
I suoi campi sono:

- `followers`: Una Map che associa ogni utente all'insieme degli altri utenti da questi seguito.
- `posts`: Una lista contenente tutti i post pubblicati sulla social network.
- `users`: Una lista di utenti registrati sulla social network.

La scelta di avere una lista di utenti in `users` nonostante si possa ottenere a partire dalla mappa `followers` è derivata dalla facilità che ciò comporta nel definire alcuni metodi della classe.

`SocialNetwork` contiene metodi necessari a registrare utenti nuovi, creare post, seguire utenti, e altri utili al fine di fare ricerche all'interno della rete senza però infrangere la barriera di rappresentazione che rappresenta la classe. A questo fine tutti i campi della classe sono `private` o `protected`, e, essendo la classe `Post` immutable, neanche gli osservatori possono porre rischio di invalidare la rappresentazione.

ReportableSocialNetwork

La classe `ReportableSocialNetwork` è una sottoclasse di `SocialNetwork` che permette la segnalazione dei `Post`. La segnalazione avviene mediante il semplice spostamento del post in questione dalla lista dei post pubblicati (definita in `SocialNetwork`) ad un'altra lista di post segnalati. È stato scelto di agire in questo modo per poter permettere ad eventuali amministratori o a strumenti automatizzati di poter accedere ai post segnalati per verificare se tali post vanno ripristinati (usando il metodo `restorePost`), o cancellati definitivamente (mediante il metodo `deletePost`), e fino a quel momento si nasconde il post agli utenti, rimuovendolo dalla lista di post pubblici. Il meccanismo di segnalazione di un post **non** è stato realizzato in modo **automatico** mediante filtri o controlli automatizzati del testo contenuto nei test, ma è **manuale** e richiede l'invocazione di un metodo apposito chiamato `reportPost`. Il metodo non effettua alcuna verifica su chi lo sta invocando, quindi eventuali verifiche per evitare che qualsiasi utente possa segnalare qualsiasi post vanno eseguite esternamente.
Unico campo che la sottoclasse aggiunge è:

- `reportedPosts`: La lista contenente tutti i `Post` segnalati.