Progetto Java[Corso B]

3 Settembre 2021

UNIVERSITÀ DI PISA

I progetto ha l'obiettivo di applicare i concetti e le tecniche di programmazione Object-Oriented esaminate durante il corso. Lo scopo del progetto è lo sviluppo di un componente software di supporto alla gestione e l'analisi di una rete sociale (SocialNetwork) denominata MicroBlog.

Mario Pietrunti 559997 m.pietrunti@studenti.unipi.it

Indice

1. LA CLASSE POST

Post è una classe mutabile contenente un id(univoco), un autore, testo del messaggio, un timestamp e ho pensato di aggiungere un accumulatore di segnalazioni nel caso in cui un post venga segnalato da un utente.
La classe implementa l'interfaccia Comparable, questo per mantenere i post ordinati in base all'id in un TreeSet. Sono presenti dei getter di ciascuna variabile, invece i setter riguardano solo il testo e l'autore di un post e ho aggiunto il metodo aggiungiSegnalazione() che incrementa il numero di segnalazioni ogni qual volta il post viene segnalato.
L'autore e il testo non possono avere valore nullo e per quanto riguarda il testo deve avere una lunghezza tra 1 e 140 caratteri.
Infine abbiamo un metodo toString() che permette di stampare il post nel formato in esso specificato.

2. LA CLASSE SOCIALNETWORK

SocialNetwork è la classe fulcro del progetto che gestisce la maggior parte delle funzioni della rete sociale. Un utente deve essere obbligatoriamente registrato per poter usufruire dei servizi della rete (ex. creare un post) e sia attraverso un "follow"/"like" ad un post, si inizia a seguire la persona indicata nel posto o autore del post alla quale è stato messo il like. Vi sono dei vincoli, in quanto un utente non può seguire se stesso e mettere mi piace ad un post inesistente.

Oltre alla HashMap social richiesta dalla specifica, ho usato una HashMap per tener traccia dei Followers di un utente che ho denominato "Followers".
Per poter gestire i post della reste sociale ho utilizzato un TreeSet per evitare di avere duplicati in fase di aggiunta e soprattutto per mantenere i post ordinati per id(univoco).

Metodi richiesti:

1. **guessFollowers(List ps)**
    - *Ho dichiarato 3 HashMap nelle quali ho distribuito i post normali, quello con dentro il "follow" e quelli con il "Like".*
    - *Controllo l'HashMap dei follow: verifico che l'utente sia registrato e aggiorno sia "social" che "followers".*
    - *Controllo l'Hashmap dei like: controllo che l'id esista e, se esiste, mi recupero l'autore del post e aggiorno "social" e "followers".*
    - *In conclusione faccio un return di "social".*

2. **Influencers**()
    - *Dichiaro una HashMap contentente tutte le coppie di utenti e numero di follower ricavate dalla rete sociale .*
    - *Mi servo di una lista di appoggio per ordinare l'hashmap ricavata per ordine decrescente di follower.*
    - *Scorro la lista ricavata e aggiungo i nomi degli influencer in una ulteriore lista, che infine restituisco.*

3. **getMentionedUsers()** e **getMentionedUsers(ps)**
   Nel primo metodo restituisco una lista di tutti gli utenti registrati. Nel secondo metodo invece restituisco i nomi degli utenti menzionati nella lista di post come se si comportasse come il tag (@utente).

4. **writtenBy(user)** e **writtenBy(ps, user)**
   Nel primo metodo restituisco tutti i post creati da quell'autore, invece nel secondo metodo cerco i post di quell'autore nella lista passata per argomento.
5. **containing(words)**
   In questo metodo effettuo una ricerca tra i post della rete sociale se sono presenti le parole contenute nella lista passata per argomento. E' sufficiente che una delle parole sia presente per poter inserire il post nella lista da ritornare.

3. LA CLASSE SECURESOCIAL CHE ESTENDE SOCIALNETWORK

La classe SecureSociale estende la classe SocialNetwork e permette di poter avere un sistema di segnalazione e rimozione di post con contenuti ritenuti offensivi in base alla presenza di un parola generica nel testo del post che ho voluto denominare "BADWORD".

Ogni utente può segnalare un post che ritiene offensivo a patto che si tratti del post di un altro autore, dopo che viene effettuata la segnalazione, quest'ultima viene valutata controllando che all'interno del testo del post segnalato sia presente o meno la "BADWORD", se la valutazione ha esito positivo il numero di segnalazioni di quel post viene incrementato di 1 e appena il numero di segnalazioni sarà pari a 2 il post verrà rimosso dalla lista ps, invece se si ha esito negativo la segnalazione verrà respinta e non contata. In conclusione ogni post non può essere segnalato più di una volta dallo stesso utente.

*La **struttura dati** utilizzata oltre a quelle ereditate da SocialNetwork è :*

Una HashMap che contiene le coppie(id, utente) dei post segnalati, chiamata "segnalati"

***Metodi**:*
//Questo metodo permette di segnalare un post della rete sociale

```
public void segnalaPost(int id, String segnalatore)
```

//Questo metodo valuta una determinata segnalazione ad un determinato post

```
public boolean valutaSegnalazione(Post p)
```

4. METODI AGGIUNTIVI

Oltre ai vari metodi di stampa aggiunti per poter visualizzare meglio gli effetti dei metodi principali sulla rete, ho aggiunti dei metodi che permettono di creare e cancellare un utente o un post e altri che permettono di seguire/smettere di seguire un utente e che restituiscono i folllowers e i seguiti di un autore.

5. ECCEZIONI

Sono presenti 4 tipi di eccezioni oltre quelle già presenti nel linguaggio Java:

- ***TxtFormatException*** *viene lanciata quando un campo String non è della giusta lunghezza o contiene caratteri proibiti*
- ***DuplicatiException*** *viene lanciata quando un elemente che si vuole inserire è già presente nella rete sociale*
- ***NotAllowedActionException*** *viene lanciata quando si vuole eseguire un'azione che non è prevista dal programma*
- ***PermessoNegatoException*** *viene lanciata quando si vuole eseguire un'azione prevista, ma che l'utente non può eseguire perché non è autorizzato*

6. TESTING

Il testing è stato gestito dividendo le eccezioni dalla corretta esecuzione del codice. Per quanto riguarda le eccezioni è stata creata una classe "BatteriaDiTest" dove queste ultime vengono testate, invece la corretta esecuzione del codice è stata testata direttamente nel "Main".