Relazione Progetto Java
Programmazione II

Stefano Passanante
Matricola: 597415
10/07/2021

1 Desrizione Progetto

Il 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. La rete sociale consente di inviare messaggi di testo di breve lunghezza, con un massimo di 140 caratteri, chiamati post. Gli utenti possono 'seguire' i post di altri utenti. Una persona è rappresentata dal nome sulla rete sociale. Gli utenti della rete sociale non possono seguire se stessi.

2 Descrizione Classe Post

Come richiesto dal Progetto, ho implementato un tipo di dato Post che è descritto da 5 variabili d'istanza:

- Id: Identificatore univoco del post
- Author: autore del post
- Text: testo del post(deve essere inferior ai 140 caratteri)
- Timestamp: indica la data e l'ora di creazione del post
- Likes: set che contiene gli utenti che hanno messo like al post

2.1 Eccezioni usate

Ho usato l'eccezione NullPointerException che viene lanciata quando il campo Author o testo contengono un valore nullo, inoltre ho definito una nuova eccezione: NewException che viene usata quando il testo che si vuole scrivere per il post contiene più di 140 caratteri, e quando il campo Author o il campo Text sono stringhe vuote

2.2 Metodi implementati

Ho implementato la get per ogni variabile del tipo di dato post, per quanto riguarda i metodi set, sono stai implementati per la varibile likes e Text. In aggiunta ai get e i set, c'è il metodo toString che ci permette di stampare un pot. Questi metodi implemtentati, rispettano i vincoli dell'invariante di rappresentazione: I getter sono degli osservatori e quindi non alterano l'IR. I setter invece rispettano l'invariante perchè ogni volta che

dobbiamo modificare la variabile text o likes, contralliamo se ciò che dobbiamo settare non sia null o una stringa vuota, e per quanto riguarda likes controlliamo se il campo utente che vuole mettere like non sia una stringa vuota o null e inoltre controlliamo se si sta mettendo like a se stessi, facendo tutto questo siamo sicuri che ogni metodo riseptti l'invariante.

3 Descrizione Classe MicroBlog

La classe MicroBlog implementa l'interfaccia SocialNetwork1. Per la classe MicroBlog ho utilizzato le seguenti strutture:

- Map<String, Set<String>> Follows: usata per tenere traccia per ogni utente: la lista di utenti che segue. Tener conto che un utente A , segue un altro utente B se e solo se: A ha messo almeno un like ad un post di B.
- Map<String, Set<String>>Followers: usata per tenere traccia per ogni utente: la lista di utenti che lo seguono. Tener conto che un utente A, è seguito da un utente B se e solo se: B ha messo almeno un like ad un post A. Ho deciso di avere anche questa map per tener traccia in maniera distanta, i Followers dai Follows, per come avviene nella maggior parte di social network, in cui noi possiamo vedere sia i follows che i followers.
- Map<String, Set<Post>>PostCaricati: questa map usata per tener traccia, per ogni utente: lista dei post caricati dall'utente. Mi è stata di fondamentale importanza per implementare il metodo getMentionedUsers(), questo perchè un utente all'interno della rete sociale, è stato menzionato se e solo se ha scritto almeno un post.
- Set<String> Utenti: Un set in cui ci sono tutti gli utenti iscritti alla rete sociale. Ogni utente quando scrivi per la prima volta un post, viene automaticamente iscritto alla rete sociale, se invece un utente non iscritto, vuole mettere un like ad un post deve obbligatoriamente iscriversi prima.
- Set<Post> Posts: Un set in cui ci sono tutti i post caricati nella rete sociale

Ho deciso di avere un Set Utenti e un Set Posts, nonostanti questi possano essere recuperati a partire da Map Follows e Map PostCaricati, perchè usando degli iteratori sui set, ho un accesso più facile ai singoli dati.

3.1 Eccezioni usate

Oltre ad aver usato l'eccezione NullPointerException, ho preferito definirmi un'altra eccezione: BlogException che viene lanciata ogni volta che c'è una stringa vuota per Username che vuole metter un like o se si vuole mettere like ad un post creato da se stessi.

3.2 Metodi Implementati

Per implementare i metodi dell'interfaccia SocialNetwork1, ho dovuto prima definite altri metodi che mi permettessero di: creare e inserire post nella rete sociale, inserire post creati dall'esterno della rete sociale, inserire e rimuovere like ai post.
InserisciLike(int ide, String Username):con questo metodo inseriamo il like di Username al post con id==ide, per implementare questo metodo, ho usato un iteratore ricerca, che contiene tutti i post del set Posts, quindi scorriamo questo iteratore fino a quando non troviamo il post che ha l'id che corrispondente all'ide passato da parametro. Se la map Follows contiene già Username come chiave aggiungiamo post.author() al set di utenti seguiti da Username, in caso contrario aggiungiamo username come chiave nella map e in seguito aggiungiamo post.author al set di utenti seguiti da Username(il set conterrà per ora, solamente il creatore del post a cui è stato messo il like),facciamo la stessa cosa per la map Followers invertendo post.author() con Usernarme.
TogliLike(int ide, String Username):Usiamo sempre un iteratore che conterrà tutti i post del set Posts, scorriamo fino a quando non troviamo il post a cui dobbiamo rimuovere il like, appena trovato, rimuoviamo il like e chiamiamo una funzione che conta i like messi da Username, se dopo la rimozione risultano 0,vuol dire che Username non segue più il creatore del post e quindi modifichiamo la map Follows e Followers.

InserisciPost(Post post):Controlliamo se il creatore del post ha precedentemente inserito altri post, se lo ha fatto aggiungiamo il post al set di post

caricati di post.author(). Se è il primo post che scrive aggiungiamo l'autore alla map PostCaricati, Follows e Followers. Non controllo se l'utente che vuole inserire il post è iscritto o meno alla rete sociale, perchè ho assunto che ogni utente che scrive un post viene automaticamente registrato nella rete sociale(come accade in alcuni blog, dove basta inserire l'username per scrivere qualcosa) invece per inserire un like devi essere registrato. Quindi, una volta aggiunto l'username alle strutture dati in questione controlliamo se è un utente registrato(Se appartiene al set Utenti), se non lo è, lo aggiungiamo al set.

CreaPost(int ide, String Username): Crea un nuovo post in base ai parametri passati e lo inserisce nella rete sociale chiamando il metodo InserisciPost.
GuessFollowers(List<Post> ps): Questo è il primo metodo richiesto dall'interfaccia SocialNetwork1 che riceve come parametron una lista di Post ps e deve restituire la rete sociale derivate dalla lista di post. Per fare questo, chiamo il costruttore MicroBlog() all'interno del metodo, per crearmi una nuova rete sociale. Per ogni post della lista di post inserisco tutti gli utenti(sia quelli che hanno scirtto il post che quelli che hanno messo like) e i post nella nuova rete sociale e infine restituisco la map Follows creatasi.
GetMentionedUsers(List<Post> ps): Secondo metodo richiesto dall'interfaccia SocialNetwork1 che riceve come parametron una lista di Post ps e deve restituire il set di utenti menzionati(Per utenti menzionati intendo tutti gli utenti che hanno scritto almeno un post) nella rete sociale formatasi dalla lista di Post ps. Per fare ciò, chiamo nuovamente il costruttore per avere una nuova rete sociale, per ogni post della lista di post inserisco gli utenti(sia quelli che hanno creato i post che quelli che hanno messo like) e i post nella rete sociale, e infine restituisco il keyset della map PostCaricati.
GetMentionedUsers(): Si richiede di restituire gli utenti menzionati(inclusi) nella rete sociale che abbiamo (per utenti menzionati intendiamo sempre tutti gli utenti che hanno scritto almeno un post), per fare ciò restituiamo semplicemente il keyset della map PostCaricati.
writtenBy(String Username): Si richiede di restituire la lista di post scritti da Username(passato come parametron) nella rete sociale. Per fare ciò, dopo un controllo sull'username, se non vengono lanciate eccezioni uso un

iteratore che contiene tutti i post scitti da Username, e inserisco ogni post nella lista di post da restituire.
writtenBy(List<Post> ps, String Username): Restituisci la lista dei post effettuati dall'utente il cui nome è dato dal parametron Username, presenti nella lista di Post ps, passata anche essa come parametron. Per fare ciò, dopo un controllo su Username e Post, se non vengono lanciate eccezioni, chiamo il costruttore MicroBlog(ps) per creare una nuova rete sociale, inserisco nuovamente tutti gli utenti e tutti i post nella nuova rete sociale, uso un iteratorre che conterrà tutti i post di Username e inserisco ogni post nella lista da restituire.
Containing(List<String> words): Deve restituire la lista di post presenti nella rete sociale che includono almeno una della parole della lista di parole passata come parametro. Per fare ciò ho preso due iteratori, il primo(lo chiamiamo Iteratore1) che contiene la lista di post della rete sociale, il secondo(lo chiamiamo Iteratore2) che contiene l'insieme di parole che dobbiamo contrallare siano nei post. Per ogni post che contiene una parola di words, aggiungo il post alla lista di post da restituire, una volta controllati tutti i post, restituisco la lista di post creatasi.
Influencers(Map<String,Set<String>> followers): Restituisci la lista dei 3 utenti più influenti(per utenti più influenti intendo gli utenti che hanno più seguaci) per fare ciò:Uso un iteratore che prende il keyset di Followers, per ogni utente dell'iteratore controllo la dimensione del suo set di Follows e in base a questa dimensione, prendo i 3 utenti con più seguaci e li inserisco in ordine nella list che deve essere restituita.

4 Descrizione estensione MicroBlogFiltrato

Come richiesto nel Progetto: definire come sia possibile progettare una estensione gerarchica del tipo di dato MicroBlog che filtri i contenuti offensive nella rete sociale. Per fare ciò passo al costruttore MicroBlogFiltrato la lista di parole che non voglio compaino nella rette sociale. Quando definiamo un'estensione gerarchica, in questo caso MicroBlogFiltrato esso, eredita tutti i metodi della superclasse. In questo caso vado a ridefinire il metodo InserisciPost per fare in modo che, se si sta inserendo nella rete sociale un post che contiene una delle parole della lista di parole passate come parametro, il testo del post viene modificato inserendo "****" al posto delle parole non ammesse.