Primo progetto intermedio
Programmazione II

Università di Pisa - Facoltà di Informatica
Jacopo Raffi 598092

25 novembre 2020

Indice

1 Sommario

Il progetto trattato riguarda la realizzazione di un social dove gli utenti registrati possono pubblicare i propri post e interagire con altre persone e i loro posts.

2 *Abstract data type*: Post

Post è un ADT immutabile la cui realizzazione viene implementata tramite la classe *MyPost* e la relativa interfaccia *Post*.

Le variabili d'istanza della classe *MyPost* sono:

- Una Stringa **author**: il nome di chi ha pubblicato il post;
- Una Stringa **text**: il testo del post;
- Una Stringa **ID**: il codice identificativo del post;
- Una Stringa **timestamp**: data e ora di pubblicazione del post;

- Un valore numerico **timeStampInMillis**: i millisecondi trascorsi dal 01/01/1970 alla pubblicazione del post. Questa variabile viene usata per la creazione dell'ID del post.

Gli otto metodi pubblici descritti all'interno della classe sono:

- Un metodo costruttore;
- Cinque metodi osservatori utili a mostrare una copia dello stato dell'istanza specifica;
- Una "compareTo" presente nell'interfaccia *Comparable*;
- Un *Override* di due metodi della superclasse *Object*, "toString" e "equals".

Bisogna speificare che l'univocità del post è garantita dall'impossibilità di pubblicare due post in un millisecondo.

3 *Abstract data type*: User

User è un ADT mutabile ideato per dare un'identità consistente agli utenti del social e per semplificare l'implementazione dei metodi presenti in "SocialNetwork" e "ReportSocialNetwork". "User" è descritto dalla classe *MyUser* e dalla relativa interfaccia *User*.

Le variabili d'istanza della classe sono:

- Una Stringa **nickname**: il nome dell'utente;
- Un insieme **UserPosts**: i posts pubblicati dall'utente;
- Un insieme **followers**: gli utenti che seguono l'utente;
- Un insieme **followed**: gli utenti seguiti dall'utente;
- Un insieme **likedPosts**: i posts a cui l'utente ha messo "like";
- Un insieme **reportedPosts**: i posts segnalati dall'utente.

I metodi presenti nella classe sono:

- Un metodo costruttore;
- Otto metodi osservatori utili ad analizzare una copia dello stato dell'istanza;
- Un *Override* di due metodi della superclasse *Object*, "toString" e "equals";
- Una "compareTo" presente nell'interfaccia *Comparable*;

- Sette metodi "default" e due privati che modificano le variabili d'istanza della classe.

Non tutti i ventuno metodi presenti nella classe *MyUser* sono pubblici. Questo perchè alcuni metodi devono essere usati solamente dalle classi all'interno del Package Social, in particolare da *MySocialNetWork* e da *ReportMySocialNetWork*.

4 *Abstract data type*: SocialNetwork

SocialNetwork è l'ADT principale del progetto. Viene realizzato tramite la classe *MySocialNetWork* e la relativa interfaccia *SocialNetWork*. SocialNetwork gestisce i tipi di dato User e Post all'interno delle proprie variabili e dei propri metodi.

Le variabili d'istanza sono:

- Una Map **microBlog**: un dizionario che associa ad ogni nome di utente l'insieme degli utenti seguiti;
- Una List **socialPosts**: la lista di posts pubblicati nel social;
- Un insieme **socialUsers**: l'insieme degli utenti registrati all'interno del social;
- Una Map **likesPosts**: un dizionario che associa ad ogni post il numero di *like* ricevuti.

Nei 21 metodi presenti in *MySocialNetWork* solo uno è privato, il resto è pubblico. Questi metodi sono:

- **influencers** con parametro: restituisce massimo le dieci persone più influenti data una rete sociale come parametro;
- **influencers** senza parametro: restituisce massimo le dieci persone più influenti del social;
- **guessFollowers**: restituisce la rete sociale derivata dalla lista di posts passata come parametro;
- **getMentionedUsers** senza parametro: restituisce gli utenti menzionati nei posts del social;
- **getMentionedUsers** con parametro: restituisce gli utenti menzionati nella lista di posts passata come parametro;

- **getAuthorsUser** con parametro: restituisce il nome degli autori della lista di posts passata come parametro;
- **getAuthorsUser** senza parametro: restituisce i nomi degli utenti del social;
- **writtenBy** con un parametro:
- **writtenBy** con due parametri:
- **containing**: restituisce la lista di posts che contengono almeno una delle parole nella lista passata come parametro;
- **addUser**: registra un utente nel social;
- **addPost**: aggiunge un post nel social;
- **follow**: riceve come parametri l'utente A e l'utente B e aggiunge B nei followers di A e A nei followed di B;
- **unfollow**: riceve come parametri l'utente A e l'utente B e rimuove B dai followers di A e A dai followed di B;
- **like**: riceve come parametri l'utente che mette il *like* e il post che lo riceve. Aggiunge il *like* al post da parte dell'utente;
- **removeLike**: riceve come parametri l'utente che rimuove il *like* e il post dal quale viene rimosso. Rimuove il *like* dell'utente dal post;
- **getSocialPosts**: restituisce i posts del social;
- **getLikesPosts**: restituisce il dizionario che associa il numero di "like" ad ogni singolo post;
- **getMicroBlog**: restituisce il microBlog derivato dal social;
- **getSocialUsers**: restituisce gli utenti del social;
- il costruttore della classe.

5 *Abstract data type*: ReportSocialNetwork

ReportSocialNetwork è un'estensione gerarchica dell'ADT SocialNetwork. La realizzazione avviene tramite la classe *ReportMySocialNetWork* che estende *MySocialNetWork* e la relativa interfaccia *ReportSocialNetwork*.

Le viariabili d'istanza sono:

- Una Map **postReports**: un dizionario che associa ad ogni post il numero di segnalazioni ricevute.

I metodi presenti nella classe sono:

- Il metodo costruttore;
- **report**: aggiunge una segnalazione al post da parte di un utente. Utente e post sono passati come parametro;
- **checkPostBan**: restituisce il numero di segnalazioni dei posts che sono stati segnalati almeno una volta;
- **postReports**: restituisce il dizionario che associa ai posts il numero di segnalazioni.

6 Conclusioni

Durante la fase di *debugging* ho utilizzato un metodo privato all'interno di ogni classe. Il metodo, denominato "checkRep", ha garantito che l'invariante di rappresentazione dei vari ADT non venisse violato all'interno del codice. Bisogna specificare che la correttezza del progetto è garantita dai test eseguiti tramite le *assert* ed eccezzioni gestite tramite opportuni *try catch*.