Relazione Progetto MicroBlog

Niccolò Campitelli – 583283

Composizione

Il progetto si compone di 17 files:

- MainClass — batteria di test e metodi per eseguirla
- Post — classe del tipo di dato Post
- SocialNetwork — classe del tipo di dato SocialNetwork
- ReportType — enumerazione delle tipologie di segnalazione
- Report — classe per la creazione di una segnalazione
- ReportSocialNetwork — classe che estende SocialNetwork, gestisce segnalazioni
- [...]Exception — una classe per ogni eccezione personalizzata

Rappresentazione astratta del tipo di dato Post

Il **Post** viene astrattamente visto come una n-upla di 6 elementi:

- Identificatore che ne determina l'univocità
- Nome dell'autore
- Testo
- Data e orario di creazione
- Data e orario di ultima modifica
- Insieme di nomi delle persone che hanno messo LIKE

Rappresentazione concreta del tipo di dato Post

- id — **long** — valore non negativo univoco
- autore — **String** — nome dell'autore
- text — **String** — testo
- data_pub — **long** — timestamp di data e orario di creazione
- data_mod — **long** — timestamp di data e orario di ultima modifica
- likes — **ArrayList<String>** — nomi delle persone che hanno messo LIKE

La classe Post utilizza uno **static long** inizializzato a 0 e di volta in volta incrementato dal costruttore per assegnare id univoci ai post creati.
La classe Post implementa **Cloneable** ed offre il metodo **clone()** che restituisce un **Post** equivalente a quello su cui il metodo è invocato, ma che non conserva riferimenti ai valori del vecchio **Post** [DEEP-COPY].

Il testo di un post può essere modificato tramite il metodo *modificaText(...)*.
I likes possono essere aggiunti e rimossi tramite i metodi *addLike(...)* e *removeLike(...)*.

Proprietà del tipo di dato Post

- nessun valore nullo
- text non è la stringa vuota e ha al massimo 140 caratteri
- likes non contiene il nome dell'autore
- likes non contiene nomi duplicati

Definizioni interne al SocialNetwork / ReportSocialNetwork

Utente A ***valido***	SSE	A scritto con caratteri alfanumerici minuscoli e '_' il primo carattere non è una cifra, A non è vuoto e non è formato da più di 24 caratteri
Utente A ***segue*** Utente B	SSE	A ha messo LIKE ad un post di B
Utente A è un ***influencer***	SSE	A è seguito da più utenti di quelli che segue
Utente A è ***menzionato***	SSE	A è nel testo di un post nella forma "@A"

Rappresentazione astratta del tipo di dato SocialNetwork

Il **SocialNetwork** viene astrattamente visto come un insieme di triple:

- Nome dell'utente u
- Insieme dei nomi degli utenti "seguiti" dall'utente u
- Insieme dei Post aventi come autore l'utente u

Rappresentazione concreta del tipo di dato SocialNetwork

I nomi degli utenti astrattamente presenti nelle triple diventano le chiavi di due tabelle:

- followers Hashmap<String, HashSet<String>> utenti che la chiave segue
- post Hashmap<String, HashSet<Post>> post di cui la chiave è autore

La classe SocialNetwork presenta un costruttore senza argomenti che istanzia una rete sociale vuota e un costruttore con argomento una lista di Post che istanzia la rete sociale derivata.

Oltre ai metodi richiesti, sono stati introdotti:

checkValidUsername(...)	privato, restituisce true sse un nome utente è valido
createUser(...)	aggiunge un nuovo utente senza post alla rete sociale
deleteUser(...)	elimina un utente e restituisce l'insieme di post di cui è autore
createPost(...)	crea e aggiunge un nuovo Post alla rete sociale
deletePost(...)	elimina e restituisce un Post della rete sociale
deleteAllPost(...)	elimina e restituisce l'insieme di Post di un utente della rete sociale
getPost(...)	restituisce un Post della rete sociale
addLike(...)	un utente mette un like ad un post della rete sociale
removeLike(...)	un utente toglie un like ad un post della rete sociale
stampa()	mostra su standard-output lo stato della rete sociale

Proprietà del tipo di dato SocialNetwork

- nessun valore nullo
- gli insiemi delle chiavi di followers e post coincidono
- ogni chiave è valida
- ogni utente seguito da una chiave è anch'esso una chiave
- ogni Post associato ad una chiave ha quella chiave come autore
- ogni like di ogni Post associato ad una chiave è una chiave
- ogni utente seguito da una chiave è diverso dalla chiave
- una chiave segue un utente SSE all'utente è associato un Post tra i cui likes c'è quella chiave

Rappresentazione astratta del tipo di dato Report

Il **Report** viene astrattamente visto come una coppia di elementi:

- Nome della persona segnalatrice
- Tipo di segnalazione

Rappresentazione concreta del tipo di dato Report

- username **String** con nome del segnalatore
- type **ReportType** con tipo di segnalazione

Rappresentazione astratta del tipo di dato ReportSocialNetwork

Il **ReportSocialNetwork** viene astrattamente visto come una coppia di insiemi:

Il primo rappresenta il tipo SocialNetwork come precedentemente descritto.
Il secondo insieme è composto da coppie di elementi:

- Post segnalato
- Report con utente segnalatore e tipo di segnalazione

Rappresentazione concreta del tipo di dato ReportSocialNetwork

ReportSocialNetwork estende **SocialNetwork** senza alterare l'implementazione di quest'ultimo.
Introduce una tabella che ha per chiavi i Post segnalati:

- reports Hashmap<Post, HashSet<Report>>

La classe ReportSocialNetwork presenta un costruttore senza argomenti che istanzia una rete sociale vuota priva di segnalazioni e un costruttore con argomento una lista di Post che istanzia la rete sociale derivata anch'essa priva di segnalazioni.

Sono stati introdotti i seguenti metodi:

addReport(...)	un utente aggiunge una segnalazione da un Post della rete sociale
getOriginalPost(...)	privato, restituisce il riferimento ad un Post della rete (senza DEEP-COPY)
stampaSegnalazioni()	mostra su standard-output lo stato delle segnalazioni nella rete sociale

Proprietà del tipo di dato ReportSocialNetwork [extends SocialNetwork]

- nessun valore nullo
- chiavi di reports sono sottoinsieme dei Post associati alle chiavi del SocialNetwork esteso
- un Post è una chiave di reports SSE ha almeno un Report associato
- ogni Post che è chiave di reports ha autore diverso dall'username di ogni Report associato
- ogni Report associato ad un Post ha username diverso da tutti gli altri username di Report associati allo stesso Post
- ogni Report associato ad un Post ha username che è chiave del SocialNetwork esteso

Relazioni tra tipi di dato

Ogni tipo di dato esterno al SocialNetwork / ReportSocialNetwork non ha alcuna relazione con esso e con gli altri tipi di dato esterni ad esso.

Ogni Post è univoco, indipendentemente dalla sua appartenenza al SocialNetwork / ReportSocialNetwork.

Identificatore, testo, date e orari di creazione e ultima modifica di un Post sono sempre validi indipendentemente dall'appartenenza del Post al SocialNetwork / ReportSocialNetwork.

L'insieme di persone che hanno messo LIKE e l'autore di un Post creato all'esterno del SocialNetwork / ReportSocialNetwork possono non essere validi quando si tenta di aggiungere il Post ad esso.

I SocialNetwork / ReportSocialNetwork non hanno alcuna relazione tra loro.

Deep-Copy e indipendenza

Per garantire le proprietà di indipendenza tra Post creati all'esterno del SocialNetwork / ReportSocialNetwork e Post presenti nella rete sociale, i metodi che si occupano di importare Post esterni o di restituire Post che continuano ad essere presenti nella rete effettuano preventivamente la DEEP-COPY attraverso il metodo **clone()**.

Si avranno quindi due Post **p1**, **p2** tali che **p2** = (Post)**p1.clone()** e **p1.equals(p2) == true** poiché manterranno lo stesso id, ma occuperanno spazi in memoria separati e le successive modifiche si rifletteranno unicamente sull'istanza scelta.

[Parte 3] Gestione delle segnalazioni tramite estensione del SocialNetwork

E' stata fornita l'implementazione di un'estensione della classe SocialNetwork che introduce la possibilità di effettuare segnalazioni:

- ogni utente può segnalare i Post presenti nella rete sociale di cui non è autore
- ogni utente può effettuare al più una segnalazione dello stesso Post
- ogni utente indica il tipo di contenuto che intende segnalare (tramite ReportType)
- le segnalazioni vengono raggruppate per Post e memorizzate nella tabella **reports**

E' possibile estendere ulteriormente la complessità della rete sociale introducendo:

- una lista di utenti-amministratori che possono oscurare i Post segnalati dagli altri utenti.
- un'estensione del metodo ***addReport(...)*** che oscura automaticamente i Post dopo un certo numero di segnalazioni, poi revisionati dagli utenti-amministratori.
- un controllo automatico del testo di un Post che lanci un'eccezione se questo contiene delle parole proibite utilizzando il metodo ***containing(...)*** della classe SocialNetwork.
- un controllo automatico del testo di un Post che sostituisca le parole proibite con degli asterischi '*****'.
- Una lista di utenti temporaneamente bannati dagli amministratori o automaticamente dopo un certo numero di segnalazioni che non possono pubblicare altri Post e non possono essere seguiti.