Siti web

Scritto dell'Appello di Programmazione Giugno 2019 (Java)

Introduzione

In questo esame vi chiederemo di scrivere un insieme di classi che riguardano una piattaforma di social networking chiamata *Facesheet*.

Classi

Descriviamo ora nei dettagli come realizzare le classi necessarie al progetto. Se lo ritenete, potete apportare cambiamenti alle segnature dei metodi proposti e anche – se lo ritenete – alla struttura delle classi, restando ovviamente all'interno dei requisiti esposti nel paragrafo introduttivo.

N.B. (1): Oltre ai metodi qui indicati, potete aggiungere ad ogni classe un valido metodo `toString()` e, dove lo ritenete, anche un metodo `equals` e un metodo `hashCode` (con le segnature appropriate e che soddisfino i contratti previsti).

N.B. (2): Nel seguito, useremo il termine *lista di X* per indicare un qualunque tipo che consenta di conservare delle sequenze di oggetti di tipo `X` : potete implementare una lista con un array (`X[]`) oppure usando ad esempio delle `ArrayList<X>` (pacchetto `java.util`).

Classe Utente

Un Utente è una persona che ha deciso di far parte di Facesheet. Ogni utente è caratterizzato da un nome, da una lista di amici (altri utenti), da una lista di post effettuati e da una lista di gruppi di cui l'utente fa parte. All'inizio tali liste sono vuote. Ogni utente può decidere se accettare o no messaggi da utenti che non sono amici. Scrivete una classe `Utente` con i seguenti costruttori e metodi:

- `Utente(String s, boolean accettaMessaggiDaNonAmici)` : crea un utente con nickname s; il secondo parametro decide se questo utente accetta di ricevere messaggi da utenti che non sono suoi amici.
- `void amico(Utente x)` : rende questo utente amico di `x` (e viceversa); non ha effetto se i due utenti sono già amici.
- `boolean amicoDi(Utente x)` : restituisce true se e solo se questo utente è amico di `x` .
- `void iscriviGruppo(Gruppo g)` : fa in modo che questo utente partecipi al gruppo indicato (non ha effetto se l'utente fa già parte di quel gruppo).
- `boolean faParteDi(Gruppo g)` : restituisce true se e solo se questo utente è parte di `g` .
- `int quantiAmici()` : restituisce quanti amici ha questo utente.
- `boolean accettaDaSconosciuti()` : dice se questo utente accetta messaggi dagli sconosciuti.

Classe Gruppo

Un Gruppo è un insieme di utenti creato con uno specifico scopo.

- `Gruppo(String nome, String descr)` : crea un gruppo (vuoto) con un dato nome e una certa descrizione.
- `void aggiungi(Utente x)` : aggiunge l'utente al gruppo (non ha effetto se l'utente fa già parte del gruppo).
- `boolean contiene(Utente x)` : restituisce true se e solo se l'utente fa parte del gruppo.
- `int numeroUtenti()` : restituisce il numero di utenti di questo gruppo.

Classe Facesheet

Questa classe rappresenta la piattaforma sociale. Dopo averne creato un'istanza, ogni volta che viene creato un utente o un gruppo viene aggiunto alla piattaforma con gli appositi metodi sotto descritti.

- `Facesheet()` : crea la piattaforma.
- `void aggiungi(Utente x)` : aggiunge un nuovo utente alla piattaforma.
- `void aggiungi(Gruppo g)` : aggiunge un nuovo gruppo alla piattaforma.
- `lista_di_Utente utenti()` : restituisce la lista degli utenti.
- `lista_di_Gruppo gruppi()` : restituisce la lista dei gruppi.
- `lista_di_Utenti foaf(Utente x)` : restituisce una lista con gli utenti che sono amici di amici di `x` (friend-of-a-friend).
- `lista_di_Utenti suggerisci(Utente x)` : restituisce una lista di utenti da suggerire a `x` come nuovi amici: si tratta dei suoi foaf e inoltre

degli utenti che appartengono agli stessi gruppi a cui `x` appartiene. Vanno esclusi, ovviamente, tutti gli utenti che sono già amici di `x`.
- `void espandi(Utente x, int k)`: prende i primi `k` suggerimenti del metodo `suggerisci` per l'utente `x` e li aggiunge come amici all'utente `x` (se i suggerimenti sono meno di `k`, li aggiunge tutti).

Classe Messaggio

Questa classe consente agli utenti di scambiarsi messaggi (simili a delle e-mail).

- `Messaggio(Utente da, Utente a, String titolo, String testo, Messaggio inRisposaA) throws IllegalArgumentException`: crea un messaggio da un certo utente a un altro utente con un dato titolo e testo; il messaggio può essere in risposta a un altro messaggio, altrimenti l'ultimo argomento è `null`. Solleva un'eccezione nei seguenti casi:
  - se l'utente `a` non è amico di `da` e inoltre non accetta messaggi dagli sconosciuti
  - se `inRispostaA` non è `null` ma non è un messaggio da `a` a `da`
- `Utente mittente()`: dice chi è il mittente di questo messaggio.
- `Utente destinatario()`: dice chi è il destinatario di questo messaggio.
- `Messaggio precedente()`: restituisce il messaggio precedente (cioè quello di cui questo è risposta), oppure `null` se questo messaggio non è in risposta a niente.
- `int lunghezzaThread()`: restituisce la lunghezza del thread di messaggi; 1 se questo messaggio non è in risposta a niente, 2 se questo messaggio è una risposta a un messaggio che non era in risposta a niente, ecc.