

**Università di Pisa-Dipartimento di Informatica**  
**Corso di Laurea in Informatica**  
**Primo progetto intermedio di Programmazione II**  
**a.a. 2020-21**

## **Introduzione**

Il progetto di programmazione II ha come obbiettivo la realizzazione di un componente software di supporto alla gestione e l'analisi di una rete sociale, per far ciò è stato implementato quanto segue.

## **La classe Post**

La classe post viene realizzata come implementazione dell'interfaccia "Post\_interface" per rappresentare un generico post pubblicato su una rete sociale, in particolare per la rappresentazione sono state utilizzate le seguenti variabili d'istanza:

- Id, è un intero che identifica univocamente un'istanza della classe Post. L'id di ogni post viene inizializzato tramite un metodo all'interno della classe "SocialNetwork" che garantisce univocità. Viene contrassegnata come final in quanto non modificabile.
- Authour, è una stringa che identifica il nome utente dell'autore del Post. Viene contrassegnata come final in quanto non modificabile.
- Text, è una stringa di lunghezza compresa tra 1 e 140 che identifica il testo del Post. Viene contrassegnata come final in quanto non modificabile.
- Timestamp, è una variabile di classe Timestamp, identifica la data e l'ora della creazione del Post. Viene contrassegnata come final in quanto non modificabile.
- Likes, è una LinkedList di stringhe che contiene i nomi degli utente che hanno messo like al post.

All'interno della classe post sono stati implementate le funzionalità richieste dalla consegna tramite la creazioni di diversi metodi tra cui i metodi di tipo get: getId, getAuthor, getText, getTimeStamp e getLikes che restituiscono gli elementi specificati dal nome del metodo. Inoltre è stato implementato il metodo addLikes che permette l'inserimento di un like all'istanza Post aggiungendo l'utente che ha inserito il like all'interno della lista Likes. All'interno di quest'ultimo metodo sono stati aggiunti dei controlli per verificare che l'utente che vuole inserire il like non sia l'autore del post e non abbia già messo mi piace, queste però non lanciano eccezioni in quanto l'azione non viene ritenuta un errore, ma viene ignorata.

## **La classe SocialNetwork**

La classe SocialNetwork viene realizzata come implementazione dell'interfaccia "SocialNetwork\_interface" per rappresentare una rete sociale.

Le variabili di istanza utilizzate sono:

- UsersPosts, è una struttura dati di tipo Map<String,Set<Post>> che identifica l'insieme degli utenti iscritti all'interno della rete sociale con l'insieme dei loro post. Viene inizializzata con un nuovo utente al momento della sua iscrizione alla rete sociale, implicando che gli utenti possono non avere un post ed essere comunque parte della rete.

- `UsersFollowing`, è una struttura dati di tipo `Map<String,Set<String>>` che identifica l'insieme degli utenti iscritti all'interno della rete sociale con l'insieme degli utenti da loro seguiti. Viene inizializzata con un nuovo utente al momento della sua iscrizione alla rete sociale e viene aggiornata quando l'utente inserirà un like ad un altro utente inserendo quest'ultimo all'interno della struttura.
- `Posts`, è una struttura dati di tipo `Map<Integer, Post>` che identifica una tabella che associa l'id ad un post, è stata implementata per rendere più efficiente la ricerca di post.
- `IDCount`, è un intero utilizzato per dare Id univoci ai post che vengono creati.
- `Soglia`, è un intero utilizzato all'interno del metodo `influencers`. Inizialmente è uguale ad 1 all'aumentare del numero di utenti registrati nella rete questo aumenta come " $(\text{NumeroUtenti}/2)-3$ ", si implementano inoltre dei controlli per far sì che la soglia non sia mai minore di 1.

La classe contiene vari metodi tra cui quelli presenti nella specifica "`SocialNetwork_interface`", in particolare si riportano le implementazioni dei metodi più rilevanti:

`GuessFollowers(List<Post> ps)`, restituisce i follower degli utenti autori dei post passati per input, dove per followers di un utente si intendono gli utenti che hanno messo like ai suoi post tra quelli passati per input. Viene fatto un controllo sui post in input per controllare se essi appartengono ad i post creati all'interno dell'istanza rete sociale, se così non fosse viene lanciata un'eccezione "`NoPostException`".

`Influencers(Map<String,Set<String>> followers)` restituisce gli utenti che possono essere definiti "influencers". Per essere definiti come influencers, si deve avere un numero di followers maggiore del numero di following e un numero di followers maggiore della variabile di istanza "soglia".

`createUser(String username)` permette la registrazione di un nuovo utente, in particolare, aggiorna le due variabili di istanza "`UsersPosts`" e "`UsersFollowing`" aggiungendo il nuovo utente. All'interno del metodo inoltre è stato implementato un controllo per evitare utenti duplicati.

`CreateNewPost(String author, String text)` permette la creazione di un nuovo post aggiornando le due variabili di istanza "`Posts`" e "`UserPosts`" e chiamando il costruttore della classe `Post`.

`Like(int id,String username)`, viene implementato per aggiungere dei controlli riguardanti la rete sociale al metodo `addLikes` della classe `post`, in particolare vengono controllate l'esistenza del post con id passato da input e che l'utente che prova a mettere like sia stato registrato nella rete sociale. Inoltre aggiorna la variabile di istanza "`UsersFollowing`" con le relazioni di following risultanti dall'azione like.

`GetUnId()`, è un metodo private che restituisce un intero unico, utilizzato per garantire l'unicità degli id dei `Post` nella rete.

Altri metodi che sono stati implementati sono i metodi getters `GetAllPosts`, `GetFollowing`, `GetSoglia` che restituiscono gli elementi specificati dal nome del metodo e gli altri metodi richiesti dalla consegna del progetto.

## La classe `CheckSocialNetwork`

Questa classe viene proposta come un'estensione gerarchica del tipo di dato SocialNetwork che permette la segnalazione di contenuti offensivi presenti nella rete sociale, infatti si introduce una variabile di istanza ReportedPosts che ha al suo interno un'associazione tra i post e gli utenti che hanno segnalato quel post. La segnalazione avviene tramite il metodo "ReportPost(String user, Post ps)" che aggiorna la variabile di istanza con il post e l'user che lo ha segnalato.

Inoltre è stato introdotto un metodo RemoveReportedPost(Post ps) per rimuovere il post dai post segnalati.

## MyExceptions

Sono state introdotte tre nuove eccezioni checked che estendono la classe Exception per catturare situazioni anomale all'interno del programma, in particolare:

- DuplicateUserException per controllare che non sia registrato due volte lo stesso utente.
- NoPostException per controllare che non si utilizzino i metodi della classe SocialNetwork su post non registrati sulla rete sociale così evitando inconsistenze.
- NoUserFoundException per controllare che non si utilizzino i metodi della classe SocialNetwork con utenti che non sono stati registrati sulla rete sociale così evitando inconsistenze.

## TestSet

TestSet è la classe contenente il main, viene utilizzata per poter provare i metodi implementati nelle classi illustrate precedentemente.

Si crea una rete sociale con il metodo creaRete() e su di essa vengono proposti due test, il primo che simula l'utilizzo di tutti i metodi all'interno della classe SocialNetwork e il secondo che simula delle situazioni anomale che hanno bisogno di essere gestite dalle eccezioni.

Infine si crea una nuova rete sociale per simulare, tramite il terzo test, i metodi all'interno della classe CheckSocialNetwork.