Analisi modificatori

registraUtente(String username) throws NullPointerException , SocialException

Questo è il metodo utilizzato per iscrivere un utente alla rete sociale.

Questo metodo, uno volta evocato su un oggetto che rispetta l'invariante di rappresentazione, continua a rispettare l'invariante di rappresentazione perché:

- Controllo se i parametri sono diversi da Null e proseguo solo in quel caso (non avrò nelle varie Map riferimenti uguali a Null)
- Controllo se la stringa è vuota, così non avrò mai utenti registrati con username uguale alla stringa vuota
- Iscrivo un utente alla rete seguendo la semantica che abbiamo adottato, ovvero aggiungiamo una nuova corrispondenza alla Map utentiSeguiti fra username e set vuoto: in questo modo assicuro che il dominio della nostra funzione di astrazione è costituito da tutti e soli gli utenti registrati

Questi controlli permettono di modificare la rappresentazione della rete sociale continuando a rispettare l'invariante di rappresentazione.

likeIt(String username, int idPost) throws NullPointerException, SocialException

Questo metodo viene utilizzato per mettere un like ad un post nella rete sociale e conseguentemente seguire una persona.

Questo metodo, uno volta evocato su un oggetto che rispetta l'invariante di rappresentazione, continua a rispettare l'invariante di rappresentazione perché:

- Controllo se i parametri sono diversi da Null e proseguo solo in quel caso (non avrò nelle varie Map riferimenti uguali a Null)
- Solo un utente iscritto può mettere like ad un post (controllo di appartenenza di username alla Map utentiSeguiti)
- Posso mettere like solamente a post che appartengono alla rete, ovvero solo ai post scritti da persone registrate alle rete: in questo modo assicuro che un utente può seguire solo utenti registrati e che, di conseguenza, il codominio della nostra funzione di astrazione è una coppia in cui il primo elemento è un sottoinsieme degli utenti registrati
- Quando username mette il like al post automaticamente segue l'autore di quel post: questo permette di rispettare l'assunzione che tutti gli utenti che sono seguiti da un utente_u sono tutti e soli quelli a cui ha messo like ad un post

Questi controlli permettono di modificare la rappresentazione della rete sociale continuando a rispettare l'invariante di rappresentazione.

removeLike(String username, int id) throws NullPointerException, SocialException

Questo metodo viene utilizzato per rimuovere il like ad un post. Particolare attenzione va fatta perché rimuovere un like significa anche rimuovere le relazioni che da esso scaturiscono.

Questo metodo, uno volta evocato su un oggetto che rispetta l'invariante di rappresentazione, continua a rispettare l'invariante di rappresentazione perché:

- Controllo se i parametri sono diversi da Null e proseguo solo in quel caso (non passerò ai metodi valori uguali a null)
- Chiamo il metodo removeLike() del post che rimuove il like di username dal post se egli aveva messo like: in tal senso non serve controllare se username era iscritto perché solo un utente iscritto può mettere like, quindi, in caso username non sia iscritto, un eccezione sarà sollevata perché ovviamente username non ha messo like al post.
- Quando si rimuove un like si elimina la relazione che scaturisce da quel like: di conseguenza se utente_u rimuove il like ad un post di utente_v e il like rimosso era l'unico like che utente_u aveva messo ad un post di utente_v, utente_u non seguirà più utente_v (viene rimosso utente_v dal set individuato da utente_u nella map utentiSeguiti).
- Al contrario la relazione di utente_u segue utente_v viene mantenuta perché vi è un altro like di utente_u a un post di utente_v che determina tale relazione: questo è fondamentale per rispettare l'osservazione secondo cui tutti gli utenti che hanno messo like ad un post conterranno nella Map followers (quindi sono utenti iscritti), nel corrispettivo set associato all'username dell'utente, l'autore del post al quale hanno messo like.

Questi controlli permettono di modificare la rappresentazione della rete sociale continuando a rispettare l'invariante di rappresentazione.

removePost(int id) throws SocialException

Questo è il metodo utilizzato per rimuovere un like dalla rete sociale.

Questo metodo, uno volta evocato su un oggetto che rispetta l'invariante di rappresentazione, continua a rispettare l'invariante di rappresentazione perché:

- Applico una removeLike() ripetuta per ogni utente che aveva messo like al post per eliminare tutte le relazioni che scaturivano dai like del post.
- Rimuovo il post da postGlobali e di conseguenza anche dal set di post scritti dall'autore del post in postUtente

Questi controlli permettono di modificare la rappresentazione della rete sociale continuando a rispettare l'invariante di rappresentazione.

createPost(String author, String text) throws NullPointerException, SocialException

Questo metodo permette il caricamento di un post non precedentemente istanziato.

Questo metodo, uno volta evocato su un oggetto che rispetta l'invariante di rappresentazione, continua a rispettare l'invariante di rappresentazione perché:

- Controllo se i parametri sono diversi da Null e proseguo solo in quel caso (non avrò nelle varie Map riferimenti uguali a Null)
- Può postare solo un utente iscritto (controllo di appartenenza di username alla Map utentiSeguiti)
- Crea il post se e solo se i parametri passati rispettano IR di post
- Ogni inserimento del post in postGlobali implica un inserimento in postUtente e viceversa: il primo inserimento con chiave id; il secondo nel set associato alla chiave post.getAuthor() (nel caso l'utente non avesse mai scritto un post creo uno corrispondenza nella map postUtente tra l'autore del post e un set contenente solamente il post)

Questi controlli permettono di modificare la rappresentazione della rete sociale continuando a rispettare l'invariante di rappresentazione

loadpost(Post ps) throws NullPointerException , SocialException

Permette l'inserimento di un post già istanziato nella rete sociale.

Il trattamento di questo caso è più complicato perché il post in questione è stato manipolato al di fuori della nostra rete sociale quindi un inserimento senza controlli potrebbe portare a non rispettare IR.

Questo metodo, uno volta evocato su un oggetto che rispetta l'invariante di rappresentazione, continua a rispettare l'invariante di rappresentazione perché:

- Controllo se i parametri sono diversi da Null e proseguo solo in quel caso (non avrò nelle varie Map riferimenti uguali a Null)
- Solo utente registrato può caricare un post (controllo se l'autore del post è registrato)
- Non tutti i like del post saranno validi: considereremo validi solo i like di persone iscritte in modo da preservare l'assunzione secondo cui il dominio della nostra funzione di astrazione è costituito da tutti e soli gli utenti registrati (se non controllassi effettivamente quali sono gli utenti registrati aggiungerei nella Map utentiSeguiti, nuovi utenti che seguono l'autore del post senza passare dalla registrazione)
- Creo una copia del post passando come lista dei like la lista dei post validi generata sopra
- Aggiungo il post creato alla Map postGlobali e postUtente con un meccanismo uguale a quello di createPost

Questi controlli permettono di modificare la rappresentazione della rete sociale continuando a rispettare l'invariante di rappresentazione

L'utilizzo dei tre metodi sopra descritti permette manipolare la nostra rete sociale rispettando l'invariante di rappresentazione e la semantica della funzione di astrazione:

- Manipolo la Map followers solo con i metodi likeIt, registraUtente, removeLike e loadPost che rispettano IR
- Manipolo la Map postGlobali e la Map postUtenti con i metodi createPost, loadPost e removePost che rispettano IR