Esame di Programmazione 2 - Unipi
Primo Progetto – Sessione Autunnale 2019

Maria Vitali, 548154

Overview

Il progetto consiste della progettazione e nell'implementazione di una collezione `DataBoard<E extends Data>`, un contenitore di oggetti generici che estendono il tipo di dato `Data`.
La collezione si comporta come uno spazio per la memorizzazione e visualizzazioni di dati di vario tipo.
Ogni dato presente nella bacheca ha associato la categoria del dato. Il proprietario della bacheca può definire le proprie categorie e stilare una lista di contatti (amici) a cui saranno visibili i dati per ogni tipologia di categoria.
Gli amici possono visualizzare i dati, che possono essere modificati solamente dal proprietario della bacheca.
Gli amici possono inoltre associare un "like" al contenuto condiviso.
Le varie componenti del progetto, presenti nel file zip allegato, sono le seguenti:

- Data.java (interface)
- Book.java (class)
- DataBoard.java (interface)
- MyBoard1.java (class), prima implementazione richiesta
- MyBoard2.java (class), seconda implementazione richiesta
- WrongPasswordException.java (class)
- EmptyDataBoardException.java (class)
- Tester.java (main class)

Data

✔ **Interfaccia**: `interface Data`
Definisce un tipo di dato astratto Data.

✔ **Implementazione**: `public class Book (implements Data)`
Implementa l'interfaccia Data. Il tipo Book rappresenta un libro, che è identificato da 4 valori: codice del libro (`int`), titolo (`String`), autore (`String`), anno di pubblicazione (`int`).

DataBoard

✔ **Interfaccia** : `interface DataBoard<E extends Data>`
Definisce la collezione `DataBoard`, contenitore di oggetti che estendono il tipo `Data`.

✔ **Implementaz. 1**: `public class MyBoard1<E extends Data> (implements DataBoard)`
La prima implementazione di DataBoard utilizza `HashMap` e `ArrayList` per l'organizzazione degli elementi della Board.
In particolare:

- 1 `ArrayList` che contiene tutte le categorie (`String`)
- 1 `HashMap` che mappa categorie (`String`) → a liste di dati (`List<E>`, dove E è il tipo generico del dato tale che `E extends Data`)
- 1 `HashMap` che mappa categorie (`String`) → a liste di amici che hanno accesso a quella categoria (`List<String>`)
- 1 `HashMap` che mappa dati (`E`) → a liste di amici che hanno messo like a quel dato (`List<String>`)

All'implementazione dell'interfaccia sono stati aggiunti i seguenti metodi di supporto:

- `private String getCategory(Data data)`, restituisce la categoria del dato data
- `public int getLikes(Data d)`, restituisce il numero di likes di un dato
- `private boolean hasAccess(String friend, String category)`, restituisce true se friend ha accesso alla categoria category, false altrimenti
- `private void sortLists(List<E> dataList, List<Integer> countersList)`, ordina la lista di dati dataList in base al numero di likes corrispondente

- un metodo **`public void display`** per ogni struttura della DataBoard, per controllare lo stato della Board quando necessario.

Per la specifica e l'implementazione di tutti i metodi, si rimanda ai file DataBoard.java e MyBoard1.java

✔ **Implementaz. 2**: `public class MyBoard2<E extends Data> (implements DataBoard)`
La seconda implementazione di DataBoard utilizza **`ArrayList`** per l'organizzazione degli elementi della Board.
In particolare:
- 1 **`ArrayList categories`** che contiene tutte le categorie (**`String`**)
- 1 **`ArrayList data`** che contiene i dati (**`E`**)
- 1 **`ArrayList dataCategories`** che contiene le categorie (**`String`**) associate ai dati. Ogni elemento di indice i di **`dataCategories`** contiene la categoria associata al dato di indice i di **`data`**.
- 1 **`ArrayList friends`** che contiene liste di amici (**`List<String>`**) che hanno accesso alle varie categorie. Ogni elemento di indice i di **`friends`** contiene la lista di amici associata alla categoria di indice i di **`categories`**.
- 1 **`ArrayList likesForData`** che contiene liste di amici (**`List<String>`**) che rappresentano i likes a un particolare dato. Ogni elemento di indice i di **`likesForData`** contiene la lista di amici che ha messo like al dato di indice i di **`data`**.

All'implementazione dell'interfaccia sono stati aggiunti i seguenti metodi di supporto:
- **`public int getLikes(Data d)`**, restituisce il numero di likes di un dato
- **`private void sortLists(List<E> dataList, List<Integer> countersList)`**, ordina la lista di dati dataList in base al numero di likes corrispondente
- un metodo **`public void display`** per ogni struttura della DataBoard, per controllare lo stato della Board quando necessario.

Per la specifica e l'implementazione di tutti i metodi, si rimanda ai file DataBoard.java e MyBoard2.java

Test: HOW TO

Per testare il codice da terminale, spostarsi dentro la directory dove sono presenti i file estratti dal .zip e:
- **`$ javac Tester.java`**
- **`$ java Tester`**

Saranno mostrati sul terminale le operazioni testate e i rispettivi risultati di tutti i metodi di una delle due implementazioni di DataBoard, divisi in blocchi distinti. Ogni blocco corrisponde al test di un particolare metodo.
Per testare il funzionamento dell'altra implementazione:
1. Aprire il file Tester.java
2. Modificare la dichiarazione della variabile dataBoard, cambiando la riga commentata a seconda del risultato desiderato:

   TEST IMPLEMENTAZIONE 1
   ```
   MyBoard1<Book> dataBoard = new MyBoard1<>(pass);
   //MyBoard2<Book> dataBoard = new MyBoard2<>(pass);
   ```
   TEST IMPLEMENTAZIONE 2
   ```
   //MyBoard1<Book> dataBoard = new MyBoard1<>(pass);
   MyBoard2<Book> dataBoard = new MyBoard2<>(pass);
   ```

Per testare il funzionamento delle eccezioni di un particolare metodo m:
1. Aprire il file Tester.java
2. A ogni metodo corrisponde un blocco di test all'interno del codice.
   ```
   /*
   //ECCEZIONI DEL METODO m
   …
   */
   ```
   Una volta trovato il blocco del metodo m che ci interessa, cancellare "/*" e "*/" all'inizio e alla fine del codice di test delle eccezioni di m.