Tema 1 POO

ACSBook - Mini-platformă de Socializare

Responsabili: Laur Neagu, Andreea Duțulescu

Introducere

Scopul acestei teme este dezvoltarea unei platforme de socializare simplificată pentru a simula interacțiunile utilizatorilor prin intermediul claselor și obiectelor. Această temă vă va ajuta să înțeleageți și să aplicați conceptele deja învățate ale programării orientate pe obiect (OOP), precum encapsularea, interfețele și abstractizarea.

Scheletul de cod furnizat include și un set de teste publice prin care puteți să vedeți, în timp ce dezvoltați, procentul de acoperire al soluției voastre raportat la cerințele temei. Vă recomandăm să studiați testele publice furnizate pentru a vedea formatul așteptat al procesării din vectorul args[]!

Această temă abordează o metodologie de tip Test-Driven Development (TDD)[1], prin care aveți la dispoziție "așteptările" din partea software-ului, iar voi va trebui să „aliniați" dezvoltarea acestuia.

Reguli generale

1. Nu este permis să modificați numele fișierului sursă unde este metoda main(), folosiți numele furnizat în schelet, și anume Tema1.java
2. Nu este permis să modificați nimic (linii de cod, fișiere, foldere) ce se află în directorul src/test, din scheletul de cod.
3. Nu este permis să implementați tema folosind EXCLUSIV generatoare de cod (Github Copilot / ChatGPT / etc.). La verificarea de plagiat, se va măsura similaritatea codului vostru cu generatoarele de cod.
4. Vă puteți instala și configura Gradle (nu este obligatoriu)[2], și rula aceste teste pe mediile voastre local (rulând comanda: gradle test). La încărcarea codului pe GitHub Classroom (rulând comanda: git push), aceste teste vor rula automat și veți vedea punctajul. Important de menționat că aceste teste publice sunt doar un set parțial de teste redactate pentru această temă. Tema include și un set de teste private ce se vor rula ulterior.

Descriere problemă

Tema își propune implementarea unei Platforme de Socializare simplă. Pentru a ușura testarea, utilizatorii acestei aplicații se vor autentifica în aplicație la orice apel din sistem, în afară de apelul de creare de utilizator.

Utilizatorii autentificați în sistem vor putea: **crea postări** (ce vor avea un conținut doar de tip text), **vizualiza postările altor utilizatori**, **da like unei postări**, **lăsa comentariu unei postări** (ce vor avea un conținut doar de tip text), **da like unui comentariu**, și **urmări alte persoane**. Postările trebuie să afișeze numărul de like-uri și comentarii, cât și numărul de like-uri per comentariu.

[1] https://testdriven.io/test-driven-development
[2] https://gradle.org/

Implementați clasele Utilizator, Postare și Comentariu cu atributele și metode adecvate. Asigurați-vă că datele sunt încapsulate corespunzător și că modificatorii de acces sunt utilizați pentru a restricționa accesul la membrii claselor.

Implementați interfața Likeable, o interfață care definește comportamentul obiectelor care pot fi apreciate (de exemplu, postări și comentarii). Aceasta ar trebui să includă o metodă pentru a aprecia un obiect.

Utilizați abstractizarea pentru a ascunde detaliile de implementare atunci când este necesar, făcând clasele mai modulare și ușor de întreținut.

Reguli implementare temă

- Nu poți face nici o operațiune (în afară de cea de creare utilizator) dacă nu ești autentificat.
- Nu poți crea un utilizator folosind un username deja existent în baza de date.
- Nu îți poți da like singur unei postări sau unui comentariu create de către tine.
- Nu poți da mai mult de un like unei postări sau unui comentariu (doar dacă îi dați unlike în prealabil).
- Nu poți da follow de mai multe ori aceleiași persoane succesiv (doar dacă îi dați unfollow în prealabil).
- Nu poți vedea postările unui utilizator pe care nu îl urmărești.
- Nu poți șterge postările sau comentariile altui utilizator. Notă – vei putea șterge doar comentariile altui utilizator din postările create de tine!
- O postare ștearsă nu va mai fi accesibilă pentru vizualizare, alături de comentariile și like-urile asupra acelei postări.
- Un comentariu șters nu va mai fi accesibil pentru vizualizare, alături de numărul de like-uri asupra acelei postări.
- Nu poți vedea lista de persoane urmărite de către un utilizator (Lista Following, vezi mai jos), ci doar pentru utilizatorul curent autentificat. Cu toate astea, vei putea vedea lista de persoane care urmăresc un utilizator (Lista Followers, vezi mai jos).

Tema va fi implementată sub forma unei aplicații Java de tip consolă, și va accepta o serie de argumente în linie de comandă ce vor trebui interpretate.

Următoarele comenzi vor trebui procesate în tema voastră. Documentația completă a acestor comenzi, cu răspunsurile așteptate în funcție de apel, este disponibilă în documentul Tema1-DocumentațieComenzi.pdf.

1. **Creare utilizator**
2. **Creare postare**
3. **Ștergere postare**
4. **Follow utilizator**
5. **Unfollow utilizator**
6. **Like unei postări**

7. **Unlike unei postări**
8. **Like unui comentariu**
9. **Unlike unui comentariu**
10. **Listă postări a persoanelor urmărite, ordonate descrescator după dată**
11. **Listă postări per utilizator, ordonate descrescator după dată**
12. **Detalii postare (text și număr de like-uri)**
13. **Listă comentarii postare și număr de like-rui, ordonate descrescator după dată**
14. **Comentează postare**
15. **Șterge comentariu postare**
16. **Listă Following per utilizator**
17. **Listă Followers per utilizator**
18. **Top 5 Most Liked posts în platformă**
19. **Top 5 Most Commented posts**
20. **Top 5 Most Followed users**
21. **Top 5 Most Liked users în platformă (suma din postări + comentarii)**
22. **Curăță toate datele din platformă**

* *Acest apel va fi folosit pentru a curăța datele istorice (utilizatori, întrebări, chestionare, soluții) existente în sistem. În practică, va trebui să curățați fișierele unde veți salva aceste date.*

Pentru citirea din fișier, regăsiți mai jos un snippet de cod prin care puteți face asta:

```
try (BufferedReader br = new BufferedReader(new FileReader(("myfile.txt"))) {
    String line;
    while ((line = br.readLine()) != null) {
       // process the line
    }
}
```

Pentru scriere (mod append) în fișier, regăsiți mai jos un snippet de cod prin care puteți face asta:

```
try (FileWriter fw = new FileWriter("myfile.txt", true);
    BufferedWriter bw = new BufferedWriter(fw);
    PrintWriter out = new PrintWriter(bw)) {

    out.println("the text");
    //more code
    out.println("more text");
    //more code
} catch (IOException e) {
    //exception handling left as an exercise for the reader
}
```

Vă recomandăm să stocați persistent informația pentru creare utilizatori, creare postări, acțiuni de follow, like, comentariu la postare în fișiere de tip text separate (se poate folosi formatul CSV – Comma Separated Value[3]).

[3] https://en.wikipedia.org/wiki/Comma-separated_values

Spre exemplu, pentru utilizatori, puteți crea un fișier users.csv, unde să aveți pe fiecare linie informația în următorul format:

Username,Password

Test1,test1234!

Test2,test2222

În mod similar, puteți stoca informația și pentru celelate date necesare.

Criterii de Evaluare

- Încapsularea corectă a datelor în cadrul claselor.
- Demonstrarea interacțiunii între obiecte, cum ar fi crearea de postări, aprecierea lor și comentarea.
- Utilizarea corectă a abstractizării pentru a ascunde detaliile de implementare.

Punctaj

DOMENIU	TASK	PUNCTAJ	TOTAL
Test	***Suită de teste rulată automat***	80 p	80 PUNCTE
Cod	***Folosirea conceptelor de POO studiate*** ***Folosirea structurilor de date studiate*** ***Lizibilitate cod, comentarii (acolo unde este cazul)***	5 p 5 p 5 p	15 PUNCTE
README	***Redactarea unui fișier Readme în care să se detalieze implementarea temei (200-500 cuvinte)***	5 p	5 PUNCTE

Bonus

Ce alte cazuri limită ați mai trata în această aplicație (minim 3 cazuri, a se descrie în README)?
Cum ați refactoriza comenzile și răspunsurile din aplicație (minim 3 propuneri, a se descrie în README)?
– 5 PUNCTE

Notițele autorilor: Chiar dacă răspunsurile primite din fiecare comandă sunt formatate precum JSONs[4] în tabelele anterioare, ele vor fi trimise de către voi, de fapt, ca string-uri, formatate în stil JSON (exemplu: '{ "status" : "error", "message" : "You need to be authenticated"}').

Totodată, pentru a vă maximiza punctajul în cadrul acestei teme, încercați să acoperiți cât mai multe dintre cazurile întoarse de fiecare comandă, și din lanțuri de comenzi succesive.

[4] https://www.json.org/json-en.html