

Tema - VideosDB

- Responsabili: Florin Mihalache, Maria Leoveanu, Radu Pavel, Radu Pătrășcoiu, Rares-Ştefan Epure, Ionela Nicuță, Adriana Drăghici
- Deadline: 29.11.2020 - 23:55 **fără depunțări de întârziere pe 30.11 și pe 1.12**
- Deadline hard: 6.12.2020 - 23:55
- Data publicării: 8.11.2020
- Data ultimei modificări: 12.11.2020

Obiective

În cadrul acestei teme veți implementa o platformă simplificată ce oferă informații despre filme și despre seriale. Utilizatorii pot să primească recomandări personalizate pe baza preferințelor.

Principalul scop al temei este de a vă familiariza cu scrierea codului în mod orientat pe obiect prin folosirea de concepte precum moștenirea, agregarea, polimorfismul și abstractizarea. De asemenea veți putea să vă exersați programarea în Java.

În cadrul acestui curs punem accentul pe calitatea codului, atât ca design, cât și ca aspect. De aceea, pentru toate temele vă încurajăm să folosiți tool-ul de analiză statică checkstyle, cu care vă puteți verifica respectarea code style-ului oficial al limbajului Java.

Descriere

Platforma pe care veți implementa simulează acțiuni pe care le pot face utilizatorii pe o platformă de vizualizare de filme: ratings, vizualizare film, căutări, recomandări etc.

Entitățile pe care le veți modela vor fi:

- Video
 - De două tipuri: filme și seriale (shows). Diferența dintre ele este că serialele au sezoane.
 - Toate videourile au în comun titlu, an lansare, unul sau mai multe genuri (e.g. comedie, thriller)
 - Sezoanele unui serial au asociat un număr, durata întregului sezon și un rating.
 - Filmele au durată și rating
- User (utilizator)
 - Are două categorii: standard și premium
 - Are videouri favorite și videouri vizualizate

Datele pentru aceste entități sunt încărcate din fișierele JSON oferite ca intrare în teste. Ele sunt tinute într-un **Repository**.

În secțiunea Detalii de implementare vom oferi exemple pentru fiecare entitate și acțiunile efectuate pe ele.

Utilizatorii pot realiza următoarele trei tipuri de **acțiuni**: comenzi, query-uri și recomandări.

Execuția temei

1. Încarcă datele citite din fișierul de test, în format JSON, în obiecte.
2. Se primesc secevențial acțiuni (comenzi, queries sau recomandări) și se execută pe măsură ce sunt primite, rezultatul lor având efect asupra Repository-ului
3. După execuțarea unei acțiuni, se afișează rezultatul ei în fișierul JSON de ieșire
4. La terminarea tuturor acțiunilor se termină și execuția programului.

Comenzi

Acestea reprezintă abilitatea unui utilizator de a realiza acțiuni directe, fiind de 3 tipuri diferite.

- **Favorite** - adaugă un video în lista de favorite videos ale aceluia user, dacă a fost deja vizionat de user-ul respectiv.
- **View** - vizualizează un video prin marcarea lui ca văzut. Dacă l-a mai văzut anterior, se incrementează numărul de vizualizări ale acelui video de către user. Atunci cand se vizualizeaza un serial, se vizualizeaza toate sezoanele.
- **Rating** - oferă rating unui video care este deja văzut (la seriale se aplică la pentru fiecare sezon în parte(spre deosebire de vizionare, unde se face pe tot serialul)).
 - Ratingul diferă în funcție de tip - pentru seriale este pentru fiecare sezon în parte.
 - Ratingul poate fi dat o singură dată de către un utilizator. Pentru seriale poate fi o singură dată pe sezon.

Query-urile

Acestea reprezintă căutări globale efectuate de utilizatori după actori, video-uri și utilizator. Rezultatele acestor query-uri sunt afișate ca output al testului.

Un query conține tipul de informație căutată: actor/video/user, criteriul de sortare și diversi parametri. În secțiunea Implementare aveți exemple pentru toate felurile de queries. Query-urile conțin ca parametru și ordinea sortărilor, dacă sortarea să se facă crescător sau descrescător. Criteriul de sortare secundar este ordinea alfabetică descrescătoare/crescătoare în funcție de ordinea de la primul criteriu.

Pentru actori:

- **Average** - primii N actori (N dat în query) sortați după media ratingurilor filmelor și a serialelor în care au jucat. Media este aritmetică și se calculează pentru toate videoclipurile(cu ratingul total diferit de 0) în care a jucat.
- **Awards** - toți actorii cu premiile menționate în query. Atenție, trebuie ca acei actori să fi primit toate premiile cerute, nu doar pe unele din ele. Sortarea se va face după numărul de premii al fiecărui actor, după ordinea precizată în input.
- **Filter Description** - toți actorii în descrierea cărora apar toate keywords-urile (case insensitive) menționate în query. Sortarea se va face după ordinea alfabetică a numelor actorilor, după ordinea precizată în input.

Pentru video-uri:

- **Rating** - primele N video-uri sortate după rating. Pentru seriale rating-ul se calculează ca media tuturor sezoanelor, cu condiția ca cel puțin un sezon să aibă rating, cele fară rating având 0. Nu se vor lua în considerare video-uri care nu au primit rating.
- **Favorite** - primele N video-uri sortate după numărul de apariții în listele de video-uri favorite ale utilizatorilor
- **Longest** - primele N video-uri sortate după durata lor. Pentru seriale se consideră suma duratelor sezoanelor.
- **Most Viewed** - primele N video-uri sortate după numărul de vizualizări. Fiecare utilizator are și o structură de date în care ține cont de câte ori a văzut un video. În cazul sezoanelor se ia întregul serial ca și număr de vizualizări, nu independent pe sezoane.

Administrativ

- Regulament
- Echipă
- Orar
- Indicații pentru activitățile online
- Recomandări cod
- Indicații pentru teme

Cursuri

- Cursuri seria CA
- Cursuri seria CD

Laboratoare

- Lab 01 - Java Basics
- Lab 02 - Constructor și referințe
- Lab 03 - Agregare și moștenire
- Lab 04 - Static, Final, Singleton
- Lab 05 - Abstracțizare
- Lab 06 - Clase interne
- Lab 07 - Overriding, Overloading & Visitor pattern
- Lab 08 - Colectii
- Lab 09 - Design Patterns
- Lab 10 - Design Patterns
- Lab 11 - Genericitate
- Lab 12 - Java8
- Lab 13 - Exceptii

Temă

- Tema - VideosDB
- Etapa 1 - Sistem energetic

Teste

- Teste grila

Resurse utile

- Instalație IntelliJ Idea
- Activare IntelliJ Idea
- Demo proiect IntelliJ Idea
- Tutorial GUI în IntelliJ Idea
- Tutorial checkstyle

Alte resurse

- Laborator recapitulare
- Exerciții vechi
- POO și Java
- JUnit
- Organizarea surselor și controlul accesului
- Tutorial I/O
- JSON & Jackson
- Double Dispatch - scurt tutorial
- Reflection

Arhiva Teme

- 2019-2020
- 2018-2019
- 2017-2018
- 2016-2017
- 2015-2016
- 2014-2015
- 2013-2014

Table of Contents

- Tema - VideosDB
 - Obiective
 - Descriere
 - Execuția temei
 - Execuția temei
 - Comenzi
 - Queries
 - Recomandări
 - Detalii pentru implementare
 - Scheletul de cod
 - Exemple date de intrare
 - Entități
 - Comenzi
 - Queries
 - Recomandări
 - Diverse detalii
 - Rulare
 - Evaluare
 - Checkstyle
 - Upload temă
 - Resurse și linkuri utile

Pentru utilizatori:

- **Number of ratings** - primii N utilizatori sortați după numărul de ratings pe care le-au dat (practic cei mai activi utilizatori)

Precizare: Dacă numarul de răspunsuri este mai mic decât N, atunci se vor returna toate. Rezultatul este de forma: "Query result: [X]", unde X este o listă de elemente, care poate să conțină 0 elemente!

Recomandările

Acestea reprezintă căutări după video-uri ale utilizatorilor. Ele sunt particularizate pe baza profilului acestora și au la bază 5 strategii.

Strategiile de recomandare:

Pentru toți utilizatorii:

- **Standard** - întoarce primul video nevăzut de utilizator din baza de date, neexistând al doilea criteriu.
- **Best unseen** - întoarce cel mai bun video nevizualizat de acel utilizator. Toate video-urile sunt ordonate crescător după rating și se alege primul din ele, al doilea criteriu fiind ordinea de apariție din baza de date.

Doar pentru utilizatorii *premium*:

- **Popular** - întoarce primul video nevizualizat din cel mai popular gen (video-urile se parcurg conform ordinii din baza de date). Popularitatea genului se calculează din numărul de vizualizări totale ale videoclipurilor de acel gen. În cazul în care toate videoclipurile din cel mai popular gen sunt vizionate de utilizator, atunci se trage la următorul gen cel mai popular. Procesul se rela pana se găsește primul videoclip care nu a fost vizionat din baza de date.
- **Favorite** - întoarce videoclipul care e cel mai des întâlnit în lista de favorite (care să nu fie văzut de către utilizatorul pentru care se aplică recomandarea) a tuturor utilizatorilor, al doilea criteriu fiind ordinea de apariție din baza de date. Atenție! Videoclipul trebuie să existe în cel puțin o listă de videoclipuri favorite ale utilizatorilor.
- **Search** - **toate videoclipurile nevăzute de user dintr-un anumit gen, dat ca filtru în input.** Sortate este în ordine crescătoare după rating, al doilea criteriu fiind numele.

În cadrul recomandărilor (fără Search Recommendation), al doilea criteriu de sortare este ordinea din baza de date (adică ordinea de apariție în câmpul "movies"/"shows" din database).

Doar în cadrul Search Recommendation al doilea criteriu de sortare este numele.

Atunci când nu se poate întoarce niciun video se afisează "XRecommendation cannot be applied!", altfel "XRecommendation result: ", unde X este numele strategiei.

Detalii pentru implementare

Scheletul de cod

În rezolvarea temei va fi nevoie de folosirea unor obiecte pentru stocarea și maparea datelor primite în format JSON. Scheletul temei vă oferă mai multe clase și enum-uri utile pentru rularea temei, citirea și parsarea testelor. Acestea se regăsesc în cadrul scheletului astfel:

- Clase de tip Enum : `ActorsAwards`, `Genre`. Acestea conțin informații despre tipurile de premii, respectiv toate genurile de videouri posibile.
- Clasele de Input din cadrul pachetului `fileio` : Acestea se vor folosi pentru parsarea datelor de input. Astfel preluată toate informațiile necesare pentru a crea propriile clase pentru rezolvarea temei.
- Clasa `Test` în interiorul căreia se poate testa individual rezolvarea, pe un anumit fișier de input.
- Clasa `Main` care este punctul de intrare pentru logica de rezolvare a temei.

Pentru a putea scrie în fișier se pot folosi metodele din clasa `Writer`:

- `closeFile` - scrie în fișier toate datele din `JSONArray`-ul dat ca parametru și închide fișierul
- `writeFile` populează un `JSONArray` cu date după formatul de input. Id este id-ul acțiunii și message este mesajul rezultat în cadrul acțiunii id.

Pentru a putea scrie în fișier este nevoie de popularea unui singur `JSONArray`.

Scheletul conține și un fișier **README** cu detalii despre rulare și fișierele oferite.

Exemple date de intrare

Entități

Utilizator

[Click pentru exemplu ↗](#)

Actor

[Click pentru exemplu ↗](#)

Video - film

[Click pentru exemplu ↗](#)

Video - serial

[Click pentru exemplu ↗](#)

Comenzi

Adaugă video-ul *The Circle* în lista de **videouri favorite** ale utilizatorului *madUnicorn3*

[Click pentru exemplu de comanda favorite ↗](#)

Adaugă video-ul *False Identity* în lista de **videouri vizualizate** ale utilizatorului *madUnicorn3*

[Click pentru exemplu de comanda view ↗](#)

Adaugă rating 9 pentru sezonul 1 al show-ului *The Haunting of Hill House* în **lista de ratings** pentru show-urile utilizatorului *kindLocust2*

[Click pentru exemplu de comanda rating ↗](#)

Queries

Oferă primii 17 **actori** sortați în ordine crescătoare în funcție de **media** acestora.

[Click pentru exemplu ↗](#)

Oferă o listă cu **actorii** ce au câștigat **premiile BEST_PERFORMANCE și BEST_SUPPORTING_ACTOR** sortate în ordine crescătoare.

[Click pentru exemplu ↗](#)

Oferă o listă cu **actorii** ce au în **descriere** cuvintele cheie *actress* și *producer* sortați în ordine crescătoare.

[Click pentru exemplu ↗](#)

Întoarce o listă cu primele 6 **filme** care sunt din anul 2019 și au genul *Action* sortate crescător după **rating**.

[Click pentru exemplu ↗](#)

Întoarce o listă cu primele 9 **filme** favorite din anul 1994 cu genul *Romance* sortate crescător după numărul de *zorritii* în lista de *filme favorite* ale utilizatorilor.

Pe exemplu în lista de filme vor fi sortate după lungimea lor.

Click pentru exemplu ↗

Întoarce o listă cu primele 8 **filme** din anul 2017 cu genul **Comedy** sortate crescător după **lungimea** lor.

Click pentru exemplu ↗

Întoarce o listă cu primele 7 **filme** din anul 2017 cu genul **Drama** sortate descrescător după **numărul de vizualizări**.

Click pentru exemplu ↗

Întoarce o listă cu primii 10 **utilizatori** sortați crescător după **numărul de ratings** pe care le-au dat.

Click pentru exemplu ↗

Recomandări

Întoarce primul **video nevăzut** de utilizatorul lyingRice4.

Click pentru exemplu ↗

Întoarce **cel mai bun** video nevizualizat de utilizatorul solidEagle6.

Click pentru exemplu ↗

Întoarce **video-ul** cel mai vizionat din **cel mai popular** gen pentru utilizatorul culturedCurlew5.

Click pentru exemplu ↗

Întoarce **filmul** care e **cel mai des întâlnit** în lista de favorite pentru utilizatorul aboardMackere15.

Click pentru exemplu ↗

Întoarce o listă cu toate **filmele nevăzute** de utilizatorul finickyZebra8 din genul **Action & Adventure**.

Click pentru exemplu ↗

Diverse detalii

- Rating - e un double. Folosiți Double.compare pt comparare
- Se vor realiza obiecte custom pentru fiecare entitate necesară, toate datele fiind parsate de clasele ce se regăsesc în pachetul fileio. (ex. ActionInputData)
- Am folosit termenul "lista de " pentru filmele vizualizate, favorite etc. Este la latitudinea voastră însă ce structură de date alegeți să folosiți pentru a ține aceste date. Mai multe detalii despre structurile de date din Java găsiți în [laboratorul Colectii](#)

Rulare

Rularea temei se va realiza exclusiv prin intermediul clasei Main, de acolo fiind apelat și checkstyle-ul. Toate testele se regăsesc în directorul **test_db** din cadrul scheletului temei.

Evaluare

Punctajul constă din:

- 80p implementare - trecerea testelor
- 10p coding style (vezi checkstyle)
- 5p README clar, concis, explicații axate pe design (flow, interacțiuni)
- 5p folosire git pentru versiunea temei



Pe pagina [Indicații pentru teme](#) găsiți indicații despre scrierea readme-ului și depunctările generale pentru teme

Depunctările pentru **designul și organizarea codului** se vor scădea din punctajul testelor. Dacă vor apărea depunctări specifice temei în momentul evaluării, nemenționate pe pagina cu depunctări generale, ele se vor incadra în limitele de maxim 15 pentru design, 5p pentru readme. Dacă tema nu respectă cerințele, sau are zero design OOP atunci se pot face depunctări suplimentare.

Folosirea git pentru versiunea va fi verificată din folderul .git pe care trebuie să îl includeți în arhiva temei. Punctajul se va acorda dacă ați făcut minim 3 commit-uri relevante și cu mesaj sugestiv.

Teste temei au punctaj diferențiat pe dificultatea lor:

- 21 teste de tip comanda unică - 2p fiecare
- 5 teste de fail - 4 au 2p, 1 are 3p
- 9 teste complexe - 3p fiecare

Bonusuri: La evaluare, putem oferi bonusuri pentru design foarte bun, cod bine documentat dar și pentru diverse elemente suplimentare alese de voi.



Temele vor fi testate împotriva plagiului. Orice tentativă de copiere va duce la **anularea punctajului** de pe parcursul semestrului și **repetarea materiei** atât pentru sursă(e) cât și pentru destinație(i), fără excepție.

Checkstyle

Unul din obiectivele temei este învățarea respectării code-style-ului limbajului pe care îl folosiți. Aceste convenții (de exemplu cum numiți fișierele, clasele, variabilele, cum indentati) sunt verificate pentru temă de către tool-ul [checkstyle](#).

Pe pagina de [Recomandări cod](#) găsiți câteva exemple de coding style.

Dacă numărul de erori depistate de checkstyle depășește 30, atunci punctele pentru coding-style nu vor fi acordate. Dacă punctajul este negativ, *acesta se trunchiază la 0*.

Exemple:

- punctaj_total = 100 și nr_erori = 200 ⇒ nota_finala = 90
- punctaj_total = 100 și nr_erori = 29 ⇒ nota_finala = 100
- punctaj_total = 80 și nr_erori = 30 ⇒ nota_finala = 80
- punctaj_total = 80 și nr_erori = 31 ⇒ nota_finala = 70

Upload temă

Arhiva pe care o veți urca pe [VMChecker](#) va trebui să conțină în directorul rădăcină:

- fisierul README
- folder-ul src cu pachetele și cu fișierele .java
- folderul .git

Resurse și linkuri utile

- [Schelet de cod](#)
- [Indicații pentru teme](#)

▪ Recomandări coding style & javadoc

poo-ca-cd/teme/tema.txt · Last modified: 2020/11/26 07:34 by ion_radu.patrașcu

Old revisions

Media Manager Back to top