Tudor Maria - Elena, 324 CC

Tema a avut un grad de dificultate mediu, iar timpul alocat pentru rezolvare a fost de cinci zile.

Dupa ce am implementat clasele din cerinta, o mare parte din munca a fost parsarea fisierelor JSON (in Parser.java). Am folosit biblioteca json-simple recomandata in cerinta temei, care a fost usor de folosit o data ce am inteles mecanismul. Am parcurs fiecare fisier astfel, folosind-ma in majoritatea timpului de doua functii din aceasta biblioteca containsKey() si get(). De asemenea, pentru instantierea Userilor a fost nevoie sa implementez Factory pattern-ul, dar si Builder Pattern pentru instantierea clasei de tip Information din interiorul clasei User. Cel mai complicat lucru la partea de citire din fisiere JSON a fost parsarea componentelor de tip List sau SortedSet.

In clasa Admin, am adaugat metode pentru crearea unui user precum addUser, generateUniqueUsername, generateUniqueNumber, generateStrongPassword, dar si o metoda pentru stergerea userilor.

In clasa Contributor, exista doua metode pentru cereri. In clasa Production, pe langa metodele uzuale (getteri, setteri, toString), exista metode care se ocupa cu rating-urile, precum si o metoda care actualizeaza valorea averageRating dupa ce un rating este adaugat sau sters de un utilizator.

In clasa Regular, exista metode care se ocupa de cereri, precum si o metoda pentru a adauga rating-urile in listele fiecarei productii, pentru ca utilizatorii Regular sunt singurii care pot adauga rating-uri.

In clasa Staff sunt implementate metodele din Staff Interface, dintre care resolveUserRequests are nevoie de niste imbunatatiri, precum si niste metode ajutatoare pentru a gasi un actor sau o productie doar stiind numele/titlul acestora.

Cele 3 clase Strategy, sunt folosite la implementarea Strategy pattern, pentru actualizarea experientei utilizatorilor, oferind diferite scoruri in functie de actiunile lor in aplicatie.

Clasa User implementeaza metode pentru adaugarea in lista de favorite a actorilor/productiilor si stergerea acestora, precum si metode pentru actualizarea experientei.

In clasa IMDB este dat drumul propriu zis aplicatiei, folosind o singura instanta conform Singleton Pattern. Metoda run parseaza fisierele JSON, dupa care autentifica utilizatorul si afiseaza meniul in functie de tipul acesteia, logica codului mutandu-se in clasa Flow. Aici se gaseste si clasa statica RequestsHolder ce se ocupa de cererile destinate tuturor adminilor.

Clasa FLow contine cele 3 meniuri in functie de tipul de utilizator si alte metode ajutatoare pentru diferitele functii: searchActors si searchProductions pentru cautarea in functie de anumite criterii a actorilor si productiilor din sistem, printProductionsByGenre, printProductionsByRating si printProductions pentru printarea tururor productiilor in functie de anumite criterii, printActorsByName si printActors pentru printarea tururor actorilor in functie de anumite criterii,

createRequest si deleteRequest pentru gestionarea cererilor, addReview si deleteReview pentru gestionarea rating-urilor, updateProduction si updateActor pentru actualizarea datelor, addProductionInSystem, enterSeasons si enterProductionsForActors pentru implementarea functiei de adaugarea a productiilor si actorilor in sistem.

AccountType, Genre, ProductionType, RequestType sunt enum-uri, ExperienceStrategy, StaffInterface si RequestsManager sunt interfete, iar InformationIncompleteException si InvalidCommandException sunt clase pentru interpretarea erorilor

Celelalte clase implementate si pe care nu le-am mentionat aici contin de obicei membrii din cerinta, unul sau doi constructori, o metoda toString pentru printarea membrilor, getteri si setteri.

Din pacate, nu am avut timp sa implementez si interfata grafica, asa ca implementarea pentru tema aceasta este toata pentru lucru in terminal.

In concluzie, am invatat nou in tema asta in special despre Design Pattern-uri si despre lucrul cu fisiere JSON, iar pe viitor sper ca voi reusi sa implementez si o interfata grafica.