README.txt
Implementarea temei mi-a luat foarte mult, cerintele destul de vagi,
un catalog electronic poate sa varieze in foarte multe feluri, eu am ales
cateva limitari, printre care:
- un asistent -- are o singura grupa corespunzatoare, un asistent nu poate preda la
mai multe grupe
- un student poate primi o singura nota la materie, compusa din cele 2 valori de tip
Double

Lipsurile mele: nu am implementat design pattern-ul de memento, stiu ca trebuie de facut
copie la arrayList-urile de note, si de salvat, fizic nu mi-o mai ajuns timpul
Nu am inteles exact logica la mediator, am incercat ceva, in care am o clasa intermediara care
preia date si decide ce actiune va executa.
Frame-ul de Course este incomplet, mai ramasesera 3 butoane la care puteam face implementarea,
nu obligatoriu complicat, dar destul de lung procesul.

Observatie: Nu am mai folosit fisierul de Test dat, aveam eu unul deja destul de complex.
Pentru a testa folosesc 2 fisiere text, in folderul manageInput se regaseste input.txt, acesta
contine toti utilizatorii de tip useri, iar pentru profesori asistenti la ce materii predau.
Pentru gradesInput.txt se regasesc notele puse de asistenti si de profesori, acestea sunt preluate de
scoreVisitor, sunt transformate in tuple-uri si dupa se aplica metoda visit si se adauga notele in catalog.
Fisierul ignore.txt contine o copie la celelalte 2 fisiere, ca sa nu se
piarda formatul in care scriu input-ul.

Pentru partea de testare am ales sa implementez mai multe clase bazate pe interfata Tester care contine 2 metode
cea de execute() care asigura logica programului, iar cea de printResults, care face printuri si se asigura ca
lucrurile sunt implementate corespunzator, Functiile de tip execute() de la fiecare clasa de tip Tester trebuie sa
se execute in ORDINE altfel nu functioneaza programul corespunzator. Ordinea este :
TestInputManagement -- se asigura ca informatiile sunt luate din fisierul input.txt si se formeaza clasele.
TestGradingSystem -- se adauga notele in catalog
TestStrategyPassingCondition -- se testeaza toate strategiile, pentru a ne asigura ca acestea functioneaza.
TestMemento - nu am apucat sa il implementez
Logica lor pleaca din fisierul Test care se afla in src/testAllFunctionalities/testUnits/Test acolo ii main-ul.

AppUsers contine clasele de tip User pe care le-am implementat.

Folderul de manageInput ma ajuta sa citesc din fisier, clasa ArrangeInputIntoCatalog este o clasa intermediara
care permite adaugarea in catalog a unui User atunci cand toate datele specifice acestuia sunt citite de la
tastatura si a putut fi format. TeacherInputReceiver, AssistantInputReceiver creaza Userii aferenti.
Clasa StudentInputReceiver nu permite creearea studentilor pana cand nu sunt creati ambii parinti,
clasa ParentInputReceiver fiind clasa interna lui StudentInputReceiver, mi s-a parut mai logic asa.
Pana la urma ce sens are sa existe un adult random pe un catalog scolar?? daca nu este adaugat din cauza copilului
acestuia.

UniSystem este folderul care contine toate clasele specifice sistemului de invatamant universitar. Cele 2 tipuri
de cursuri si clasa abstracta. Clasa Grade, Group care esxtinde TreeSet, clasa cea mai importanta Catalog si
o clasa intermediara pe care o folosesc pentru a-mi usura lucrurile SearchSystem, o folosesc pe aceasta ca
sa dau parametrii si sa obtin din Catalog ceea ce ma intereseaza de exemplu, dau un nume si prenume si obtin
studentul/profesorul/ asistentul pe care il caut.

Folderul Utilities contine ScoreVisitor-ul, si designPattern-ul de strategy cu tot ceea ce intereseaza, de
asemenea se regasesc Notificarile pe care le primesc parintii atunci cand odrasla lor primeste o nota.
Ambii parinti sunt abonati la catalog, ambii parinti primesc notificare, la fiecare nota adaugata.