Tema 2 POO

Responsabili: Andreea Dutulescu, Laur Neagu
Concepte: Colecții, Genericitate [1], Excepții, I/O

Introducere

Tema are ca scop implementarea unui sistem informatic pentru gestiunea și repartizarea studenților la materiile opționale din facultate. Astfel, sistemul va fi pus la dispoziție secretariatului pentru gestionarea mediilor studenților, preferințelor de repartizare și specificului cursurilor.

Funcționalități

Sistemul informatic va trebui să ofere implementări pentru următoarele entități și un program principal care să citească comenzile și să facă acțiunile necesare: Secretariat (care gestionează întreg fluxul, conține baze de date cu studenții și cursurile aferente, răspunde la interogări și realizează repartizarea), Student și Curs.

Descrierea entităților

Student - entitate de tip Student ce va fi modelată printr-o clasă. Studentul este definit de numele său, ulterior i se va atribui o medie. Student va servi drept clasă părinte pentru două tipuri de studenți (student de master și student de licență). Aceast aspect va fi modelat prin moștenire.
Curs - entitatea de tip Curs va fi modelată printr-o clasă. Cursul va fi definit de denumirea lui și capacitatea maximă. În plus, va avea o colecție care la final va conține toți studenții înrolați la acel curs. În cadrul unui curs se pot înrola doar studenții de la respectivul program de studiu (fie doar studenți de master, fie doar studenți de licență). Astfel, clasa de tip Curs va fi parametrizată folosind genericitate astfel încât colecția aferentă să admită entități care moștenesc clasa Student, dar să nu permită asignarea unui student de master în colecția unui curs ce admite studenți de licență.
Secretariat - Secretariatul este entitatea principală care centralizează bazele de date cu studenți și cursuri. Modul de implementare al funcționalităților clasei este la latitudinea dezvoltatorului, însă clasa Secretariat trebuie să poată răspunde la toate comenzile tratate în programul principal (în clasa **Main** se va interacționa doar cu metodele Secretariatului, nu și cu metodele/atributele claselor Student și Curs, nu se vor menține colecții sau baze de date). Toate comenzile necesare sunt citite dintr-un fișier.

Descrierea comenzilor

Comenzile vor fi citite și parsate în cadrul clasei **Main**. Funcția `public static void main(String[] args)` va primi ca parametru numele aferent testului ce urmează să fie executat. Astfel, folderul aferent testului va avea denumirea parametrului, iar fișierul care conține comenzile se va afla în folderul aferent și va avea denumirea parametrului și extensia `.in`. De exemplu, pentru parametrul `00-test`, folderul averent va avea calea "`src/main/resources/00-test`", iar fișierul cu comenzi va avea calea "`src/main/resources/00-test/00-test.in`". Va trebui să creați și să scrieți rezultatul comenzilor ce necesită afișare în fișierul "`src/main/resources/00-test/00-test.out`", iar fișierul de referință, pe care nu aveți voie să îl modificați, se va afla în

“src/main/resources/00-test/00-test.ref”. În cadrul folderului, vor mai exista și alte fișiere prefixate cu “note_”, pe care le veți folosi pentru a citi notele studenților.

Comanda	Descriere
adauga_student - <program_de_studii> - <nume_student> Ex: adauga_student - master - Anna	Adăugarea unui student în baza de date a secretariatului. La adăugare, studentul nu are o medie sau un curs asignat, acestea urmând a fi setate ulterior. În funcție de tipul de program de studiu, se va crea o entitate de student master sau student licență. Numele unui student ar trebui să fie unic, astfel că, în cazul adăugării unui student cu nume duplicat, indiferent de programul de studii, Secretariatul trebuie să arunce o excepție definită de voi, care va fi tratată într-un bloc try-catch în Main. În acest caz, studentul nu va fi adăugat în baza de date și se va afișa mesajul Student duplicat: nume_student
adauga_curs - <program_de_studii> - <nume_curs> - <capacitatea_maxima> Ex: adauga_curs - licenta - POO - 5	Adăugarea unui curs în baza de date a secretariatului. La adăugare, cursul nu are încă niciun student asignat, acest lucru urmând să se facă la următoarele comenzi. În funcție de tipul programului de studiu, obiectul de tip curs creat va fi parametrizat cu tipul de student pe care îl acceptă. Pe lângă acest lucru, cursului i se va seta numele și capacitatea maximă de studenți pe care o poate accepta.
citeste_mediile	Secretariatul citește mediile din fișierele prefixate cu note_ din folderul aferent testului. Numărul fișierelor poate varia de la un test la altul, astfel că funcționalitatea se va implementa pentru citirea tuturor fișierelor cu note din folderul aferent, indiferent de numărul acestora. Un fișier cu note conține pe fiecare linie <nume_student> - <media_generala> și toate aceste informații trebuie centralizate, ca în final, entităților de tip Student să li se asigneze media lor. Ex: Carol - 9.55
posteaza_mediile	Secretariatul va posta mediile studenților pentru a putea fi inspectate de aceștia în vederea corectării anumitor note greșite. Astfel, în fișierul .out, se va afișa, separat pe câte o linie, studenții și mediile lor în ordinea descrescătoare a mediilor (și în caz de egalitate, în ordinea alfabetică a studenților). De câte ori apare comanda de postare de note, trebuie afișate notele actualizate, eventual modificate în urma unor contestații. Ex:

	Carol - 9.55 Anna - 8.6 …
contestatie - <nume_student> - <noua_medie> Ex: contestatie - Anna - 8.9	În cazul sesizării unor nereguli, studenții pot solicita o contestație a mediei generale, pe care secretariatul trebuie să o ia în calcul și să o soluționeze. Astfel, se va actualiza în baza de date și în cadrul entității corespunzătoare de student noua medie. La o eventuală postare a mediilor ulterioară, nota studentului va fi cea din cadrul contestației.
adauga_preferinte - <nume_student> - <curs1> - <curs2> - <curs…> … Ex: adauga_preferinte - Anna - POO - SO - SD	Comanda de adăugare preferințe va apărea după soluționarea tuturor contestațiilor și va presupune menținerea unei liste cu prioritățile fiecărui student în parte. Astfel, studenții își fac cunoscută prioritizarea cursurilor opționale la care doresc să fie repartizați.
repartizeaza	După adăugarea tuturor preferințelor studenților, Secretariatul trebuie să facă repartizarea studenților la cursurile opționale, ținând cont de media acestora și preferințele lor. Astfel, studenții sunt ordonați în funcție de medie. Pe rând, începând cu primul student, se va încerca repartizarea acestuia la primul curs din lista sa de preferințe. Dacă încă nu a fost atinsă capacitatea maximă la acel curs, studentul va fi asignat acolo și se va trece mai departe. Altfel, se va lua următorul curs din lista sa de preferințe și se va testa dacă există loc. În situația în care la un curs s-a atins capacitatea maximă, dar studentul ce urmează să fie asignat are aceeași medie ca cea mai mică medie existentă la acel curs, el va fi asignat acolo, în ciuda depășirii capacității maxime. Puteți consulta fișierele de test pentru mai multe exemple în ceea ce privește repartizarea. După repartizare, toți studenții vor avea un curs asignat. Se garantează că repartizarea se poate face pentru toți studenții.
posteaza_curs - <nume_curs> Ex: posteaza_curs - POO	În cadrul acestei comenzi, se va afișa în fișierul .out detalii despre cursul cerut și studenții asignați, în ordine alfabetică, sub următorul format: <nume_curs> (<capacitatea_maxima>) <nume_student> - <media> <nume_student> - <media> …
posteaza_student - <nume_student>	În cadrul acestei comenzi, se va afișa în fișierul .out detalii despre studentul cerut,

Ex: `posteaza_student - Anna`	sub următorul format: `Student <ciclul_de_studii>: <nume_student> - <media> - <nume_curs_asignat>`

*** Pentru a putea urmări mai ușor lucrurile afișate în fișierul `.out`, înainte de afișarea elementelor cerute de o anumită comandă, se va afișa pe un rând separat șirul de caractere `***`, pentru o delimitare vizuală a outputului comenzilor.

Punctaj

*Se cere folosirea exclusivă a Java Collections. Nu se admit tablouri standard.

50 de puncte - Suita de teste
20 de puncte - Folosirea colecțiilor eficiente din punct de vedere al complexității de execuție, potrivite pentru operațiile efectuate
5 puncte - Folosirea genericității în cadrul clasei Curs.
10 puncte - Definirea, aruncarea și tratarea corectă a excepției pentru 2 studenți cu același nume.
10 puncte - Folosirea conceptelor de POO studiate (encapsulare, abstractizare, moștenire și polimorfism)
5 puncte - Readme detaliat în care să se specifice motivul alegerii unui anumit tip de colecție pentru un anume task
10 puncte - Bonus: Modelați funcționalitatea în care, dacă un student nu reușește să fie asignat la niciunul din cursurile de pe lista de priorități, să fie asignat la un alt curs care mai are loc. Adăugați un test și fișierele aferente pentru această funcționalitate. Asignarea trebuie să fie deterministă (nu random, pentru a putea verifica corectitudinea prin teste).

[1] https://docs.oracle.com/javase/tutorial/java/generics/types.html