PROIECT FINAL
JAVA 1 Associate

Proiectul are ca scop gestionarea activităților zilnice dintr-o singură bibliotecă. Din această bibliotecă se pot împrumuta o varietate de cărti. Orice carte are un cod unic, un titlu, un autor, un gen, un număr de pagini. De asemenea, se știe dacă o carte este sau nu disponibilă (este momentan împrumutata sau nu de altcineva). Există mai mulți clienți, identificați prin cod, nume, număr de cărți împrumutate de-a lungul timpului, o dată de retur pentru cartea împrumutată curent (de ex un string sub forma zi/lună/an); dacă nu există, acest câmp va fi gol. Un client poate să împrumute o singură carte la un anumit moment. Clienții acestei biblioteci sunt de două tipuri: studenti, ce au în plus definită facultatea, anul de studiu și profesori, ce au în plus o materie predată. Nu pot să existe doi clienți cu același nume.

Se cere realizarea unei aplicații care are urmatoarele cazuri de utilizare:

1. adaugaStudent – adaugă un student în evidența clienților. Se va arunca excepția NumeDejaExistentException dacă există deja un client al bibliotecii cu același nume

2. adaugaProfesor - adaugă un profesor în evidența clienților. Se va arunca excepția NumeDejaExistentException dacă există deja un client al bibliotecii cu același nume

3. afiseazaClienti – afișează toți clienții bibliotecii în ordinea în care au fost înregistrați

4. afiseazaStudenti – afișează doar studenții care sunt clienti ai bibliotecii

5. adaugaCarte – adaugă o nouă carte în bibliotecă

6. afiseazaCarti – afișează toate cărțile bibliotecii în ordinea în care au fost adăugate

7. afiseazaCartiDisponibile – afișează doar cărțile ce pot să fie împrumutate acum

8. cautaCarte – afișează detaliile cărtii căutate (adica valorile atributelor/implementarea toString)

9. filtreazaCartiDupaGen – se vor afișa doar cărțile ce aparțin unui anumit gen

10. sorteazaCarti – afișează cărțile ordonate dupa numărul de pagini în ordine crescătoare

11. sorteazaClienti – afișează clienții ordonați după numele lor, lexicografic

12. celMaiFidelCititor – afișează datele clientului care a împrumutat cele mai multe cărți

13. imprumutaCarte – se va împrumuta una dintre cărti; se va seta o data de retur; nu se va putea împrumuta o carte ce nu este disponibilă (nu există în bibliotecă sau e deja împrumutată), caz în care se va arunca o excepție ce va fi și tratată -CarteIndisponibilaException

14. returneazaCarte – se va înapoia cartea

15. stergeCarte – nu va mai exista cartea în bibliotecă

16. stergeClient – se va elimina clientul din evidență

17. exit – închide aplicația

Pentru fiecare caz de utilizare dintre cele de mai sus, aplicația poate primi de la tastatură următoarele comenzi:

1. “adaugaStudent <<atribute specifice>>” – unde <<atribute specifice>> reprezintă valori specifice atributelor unui student – 5p

2. “adaugaProfesor <<atribute specifice>>”– unde <<atribute specifice>> reprezintă valori specifice atributelor unui profesor – 5p

3.” afiseazaClienti” – 4p

4. “afiseazaStudenti” - 4p

5. “adaugaCarte <<atribute specifice>>”– unde <<atribute specifice>> reprezintă valori specifice atributelor unei cărți – 5p

6. “afiseazaCarti” – 4p

7. “afiseazaCartiDisponibile” – 4p

8. “cautaCarte <<titlu>>” – unde << titlu >> reprezintă titlul cautat – 3p

9. “filtreazaCartiDupaGen <<gen>>” – unde <<gen>> reprezintă genul dupa care filtrez cărțile –3p

10. “sorteazaCarti” – 3p

11. “sorteazaClienti” – 3p

12. “celMaiFidelCititor” – 3p

13. “imprumutaCarte <<cod, numeClient>>” – unde <<cod>> reprezintă codul cărții de împrumutat si <<numeClient>> este numele clientului care imprumuta cartea – 4p

14. “returneazaCarte <<cod, numeClient >>” – unde <<cod>> reprezintă codul cărții de returnat si <<numeClient>> este numele clientului care returneaza cartea – 4p

15. “stergeCarte <<titlu>>”- unde «titlu» reprezintă numele cărții care va fi ștearsă–4p

16. “stergeClient <<nume>>” - unde «nume» reprezintă numele clientului care va fi șters –4p

17. “exit” – 3p

Bonus:

1. Se va realiza un fir de execuție care va rula în fundal și va afișa o dată la 50 de secundenumărul cartilor din biblioteca – 5p

2. Se va implementa un caz de utilizare verificareIstoricCarte care va afisa câți proprietari a mai avut înainte cartea respectiva. Puteți să și afisați care au fost aceștia, în funcție de modaliatea de rezolvare pe care o veți aborda. – 5p

3. Se va implementa metoda arePenalitati ce va verifica daca s-a depășit data de depunere. Formatul comenzii este acesta :

   "arePenalitati <<nume>>" – unde <<nume>> reprezintă numele clientului pentru care se face verificarea – 5p

Despre implementare:

Se va respecta principiul encapsulării claselor – 5p
Se va folosi Singleton Pattern unde este nevoie – 5p
Se vor folosi expresii lambda acolo unde este posibil – 5p
Se va implementa equals si toString in clasele unde este nevoie – 5p
Design-ul claselor va fi în conformitate cu cerința – 5p
Se vor respecta principiile OOP și standardele de scriere a codului; - 10p

Observatii :

Se acorda punctaje partiale.
Se corecteaza solutia chiar daca prezinta erori de compilare.
Se considera ca toate comenzile introduse de la tastatura sunt corecte. (nu este necesar sa fie realizate validari)

Mult succes!