Tema POO - 2013-2014
Catalog şcolar

Scopul acestei teme reprezintă implementarea unei interfețe grafice ce permite gestiunea unui catalog centralizator pentru un liceu.
Punctaj: 2p din nota finală (2.5 cu bonus)
Termen Predare: duminica 12 ianuarie 2013, ora 23:55
Atenție!
Tema se va uploada pe curs.cs doar până la această dată, urmând ca la ultimul laborator (săptămâna 14), fiecare student să-şi prezinte personal tema la grupa lui, preferabil pe laptopul personal pentru a evita eventualele probleme de rulare!! Se va rula aplicația din arhiva trimisă - pentru aceasta e bine să vă păstrați această arhivă!

Arhitectura:

Interfeţe

1. **IAdministrator** - interfaţa cu operaţiile pe care le poate executa un Administrator
2. **IProfesor** - interfaţa cu operaţiile pe care le poate executa un Profesor
3. **ISecretar** - interfaţa cu operaţiile pe care le poate executa un Secretar
4. **IElev** - interfaţa cu operaţiile pe care le poate executa un Elev

Clase abstracte

1. **Utilizator** – clasa ce reprezintă persoanele ce vor folosi catalogul.

Clase concrete (instantiabile)

1. **Elev** (moşteneşte **Utilizator**, implementează **IElev**)– clasa ce reprezintă subiectul principal al catalogului.
2. **Profesor** (moşteneşte **Utilizator**, implementează **IProfesor**)– instanţele acestei clase vor putea modifica notele şi absenţele din catalog pentru fiecare elev.
3. **Secretar** (moşteneşte **Utilizator**, implementează **ISecretar**)– va adăuga la catalog elevi, îi va edita sau îi va şterge
4. **Administrator** (moşteneşte **Utilizator**, implementează **IProfesor şi ISecretar**) - are şi drept de adăugare şi şterge utilizatori pe lângă toate celelalte drepturi pe care le au un Profesor şi un Secretar
5. **Clasă** - reprezintă o clasă de elevi având un identificator (9A, 12C etc), elevii, materiile şi catalogul.
6. **SituaţieMaterieBază** - reprezintă situaţia şcolară a unui Elev la o anumită Materie
7. **SituaţieMaterieCuTeză** (moşteneşte **SituaţieMaterieBază**) - are aceeaşi funcţionalitate de bază ca SituaţieMaterieBază, dar este folosită în cazul unei materii cu teză. Teza va reprezenta un sfert din medie.
8. **Absenţă** (***clasă internă*** a clasei **SituaţieMaterieBază**) - reprezintă o absenţă a unui elev

9. **Materie** - conţine informaţiile despre o materie din curriculum
10. **Catalog** - conţine situaţia şcolară a tuturor elevilor dintr-o clasă
11. **Centralizator** - centralizează toate clasele de elevi din şcoală, precum şi celelalte liste de utilizatori.

Descriere interfete - schelet *minimal*

IAdministrator:

- Metode pentru :
  - adăugarea unui nou utilizator
  - ştergerea unui utilizator
  - listarea utilizatorilor

IProfesor:

- Metode pentru:
  - Listarea elevilor unei clase
  - Ordonarea elevilor unei clase după un criteriu specificat
  - Vizualizarea/Editarea situaţiei şcolare a unui elev.
    - adăugare note noi
    - modificare (incheiere) medii
    - adăugarea unei absenţe
    - modificarea statusului unei absenţe - marcare drept motivată/nemotivată

  Obs. Listele cu elevi vor putea fi ordonate după mai multe criterii (nume, media de la o materie, media generală, numărul de absenţe).

ISecretar:

- Metode pentru:
  - Adăugarea/Stergerea/Editarea unei clase
  - Adăugarea/stergerea/editarea unei materii pentru un profesor
  - Adăugarea/Stergerea/Editarea elevilor/materiilor unei clase
  - Calcularea mediei generale

IElev:

- Metode pentru:
  - afişarea situaţiei şcolare

Descriere clase - schelet *minimal*

Utilizator:

- Date:
  - numeUtilizator
  - parolă

    - nume
    - prenume
- Metode:
    - toString()

Se va putea loga la aplicaţia catalog folosind un nume de utilizator şi o parolă.
În funcţie de tipul utilizatorului: elev/profesor/secretar/administrator funcţionalităţile şi interfaţa vor fi diferite.

Profesor:

- Date:
    - Materia la care predă.

Elev:

- Date:
    - CNP
    - data naştere

Își poate vizualiza situația școlară detaliată: date personale, note, medii, absențe (interfața IElev).

Clasă:

- Date:
    - idClasa
    - elevii
    - materiile
    - catalogul
- Metode pentru:
    - adăugarea unui elev la o clasă
    - ștergerea unui elev dintr-o clasă

Materie:

- Date:
    - nume
    - număr de ore pe săptămână
    - are/nu are teză

SituaţieMaterieBază:

- Date:
    - un obiect de tip Materie
    - 2 liste cu notele pe semestre
    - mediile pe cele 2 semestre
    - listă cu absenţele
- Metode pentru:
    - calcularea mediei materiei
    - adăugare notă
    - adăugare absenţă
    - modificare absenţă

SituaţieMaterieCuTeză:

- Date:

    - nota de la teză
- Metode pentru:
    - calcularea mediei materiei

Absenţă:

- Date:
    - status(motivată, nemotivată, nedeterminat)
    - data

Catalog:

- Date:
    - dicţionar cu elevii şi situaţia şcolară asociată fiecărei materii (dicţionar de dicţionare cheie = Elev, valoare = dicţionar Materie - SituaţieMaterieBază)

Centralizator:

- Date:
    - listele tuturor utilizatorilor
    - o listă cu toate clasele din acea şcoală
    - un dicţionar cu materiile, profesorul asociat fiecărei materii şi fiecărei clase dicţionar - (cheie: Materie, valoare: dicţionar - (cheie:Clasă, valoare: Profesor))

Toate câmpurile claselor trebuie să fie private. Pe lângă aceste date puteţi adăuga orice altceva consideraţi util!!

Toate clasele trebuie să aibă metoda toString implementată!!

Interfaţa grafică - Java Swing

Pentru gestiunea catalogului, veţi crea o interfaţă grafică folosind Java Swing care trebuie să cuprindă următoarele elemente:

- pagină de logare: 2 câmpuri pentru numeUtilizator si parolă
- să apară permanent după logare numele de utilizator şi rolul utilizatorului (adică: elev, profesor, secretar sau administrator)
- un meniu din care utilizatorul să poată alege ceea ce doreşte să facă

Operații minimale:

Elev:

- pagină cu informaţiile detaliate ale elevului logat

Profesor:

- pagină din care profesorul să poată alege una din clasele la care predă. Alegerea implică afişarea unei liste în care să apară elevii din clasa aleasă (numele, media de la fiecare materie, numărul de absenţe/materie, numărul total de absenţe şi media generală). Aici profesorul poate să ordoneze elevii după criteriile menţionate anterior.
- pagină cu informaţiile despre un elev din care poate să editeze notele şi să gestioneze absenţele acestuia numai de la materia lui.

Secretar:

- pagină de adăugare/ştergere/editare a unei clase
- pagină de adăugare/ştergere/editare a unei materii pentru un profesor

Pentru o clasă:

- pagină de adăugare a unui elev
- pagină asemănătoare celei ce apare unui profesor cu lista tuturor elevilor, dar secretarul să aibă şi posibilitatea să şteargă un elev, nu doar să-i editeze datele personale
- pagină de editare a datelor personale ale elevului (nu poate edita note/absenţe, ci doar datele personale)

Administrator:

- pagină de creare utilizator: numeUtilizator, parolă şi rolul utilizatorului (elev, profesor, secretar, administrator)
- pagină cu o listele tuturor utilizatorilor din care să se poată şterge/adăuga/edita un utilizator
- toate paginile pe care le au un Secretar şi un Profesor

BONUS: Se va acorda bonus pentru o interfață grafică intutitivă şi complexă, frumos realizată :-) care pune la dispoziţie toate operaţiile implementate la arhitectură!!!

Detalii de implementare

- ➢ *Se va utiliza **tipizarea** colecţiilor folosite (utilizaţi ArrayList<Elev>, iar nu ArrayList).*
- ➢ *Clasa Centralizator se va implementa folosind **Singleton pattern**. [0]*
- ➢ *Adăugarea de utilizatori de către administrator se va face folosind **Factory pattern** . [1]*
- ➢ *Trebuie să aveţi o listă cu utilizatori reţinută într-un **fişier** în care să verificaţi dacă credenţialele furnizate la logare de un potenţial utilizator sunt corecte!*
- ➢ *Listele elevi/materii etc. trebuie reţinute şi ele în nişte fişiere, iar la pornirea aplicației **datele din acestea să fie încărcate**, iar la închiderea aplicației, modificările să fie **salvate** tot în acele fişiere!*
- ➢ *Pentru realizarea interfeței grafice a aplicației se vor folosi cât mai multe din facilitățile şi componentele Swing - panel-uri (ex. JTabbedPane), componente MVC - JComboBox, JList, JTable etc, precum şi tot ce poate să facă interfața cât mai atractivă!*

Punctaj (20 puncte + 5puncte Bonus)

(12p) Implementarea ierarhiei de clase cu toate câmpurile şi metodele necesare implementării funcţionalităţilor menţionate. Din care:

(2p) Implementare cu *Singleton pattern*

(2p) Implementare cu *Factory pattern*

(8p) Realizarea interfaței grafice a aplicației.

(5p) BONUS: Design reuşit interfață grafică!

Alte observații

➢ Se vor defini două package-uri (pachete):
- Pachetul “**liceu**“ va conține clasele şi interfețele specificate în enunțul temei.
- Clasele ce afişează interfețele grafice se vor defini în pachetul “**grafic**”.

Vor fi depunctate cu **10p** temele care definesc toate clasele in acelasi pachet!

➢ Validarea implementării arhitecturii se va realiza cu un checker pentru a verifica faptul ca ați respectat condițiile impuse: doar variabile private, fără static etc.

➢ Pentru testare veți trimite aplicația cu cel puțin o entitate din fiecare clasă deja creată (cel puțin o clasă de elevi, câțiva elevi în clasă, câteva materii, câțiva profesori, elevi, secretari etc.) Reamintim că datele aplicației (de la o rulare la alta) vor fi păstrate în fişiere.

➢ Temele sunt **INDIVIDUALE**. Copierea de la alți colegi se va sancționa cu **punctaj 0** atât pentru cel care a copiat cât şi pentru cel de la care s-a copiat. Copierea de pe Internet atrage de asemenea punctaj 0 pe temă pentru cel care a copiat.

➢ Tema o veți trimite într-o arhivă .zip de forma **grupa_NUME_Prenume.zip** care va conține:
- un folder SURSE ce conține doar sursele Java
- un folder POIECT ce conține proiectul NetBeans sau Eclipse
- un fişier README în care veți specifica numele, grupa, gradul de dificultate al temei, timpul alocat, modul de implementare şi alte obervații dacă există. Lipsa fişierului README duce la o **depunctare de 10p**.

Responsabili temă:

Cosmin Didii (cosmin.didii@gmail.com)
Taygun Agiali (taygunagiali@gmail.com)
Alexandra Vieru (alexandra.vieru@cti.pub.ro)

Resurse

[0] Singleton pattern
http://www.tutorialspoint.com/java/java_using_singleton.htm
http://howtodoinjava.com/2012/10/22/singleton-design-pattern-in-java/
http://en.wikipedia.org/wiki/Singleton_pattern
[1] Factory pattern
http://www.oodesign.com/factory-pattern.html
http://howtodoinjava.com/2012/10/23/implementing-factory-design-pattern-in-java/
http://en.wikipedia.org/wiki/Factory_method_pattern