LABORATOR 5 – Arhitecturi pentru sisteme software

Obiectiv – Înţelegerea arhitecturilor cu invocare implicită.

Sistemul de bază – aplicaţie pentru înregistrarea studenţilor la cursuri – dată sub formă de cod sursă.

Activitate:

1. Modificarea codului existent şi analiza implicaţiilor.
2. Analizarea arhitecturii sistemului, a modificărilor acesteia şi a implicaţiilor.

DETALII:

Funcţionalitatea sistemului de bază.

Funcţia de bază a sistemului este înregistrarea studenţilor la cursuri. Sistemul oferă suport pentru înregistrarea unui student la cursuri şi răspunde la o serie de interogări, cum ar fi listarea cursurilor la care studentul este înregistrat.

Pentru realizarea acestei funcţionalităţi sistemul menţine 2 liste : (1) lista studenţilor şi (2) lista cursurilor. O instanţă a clasei `Student` este un obiect ce menţine 2 liste interne : (1) lista cursurilor absolvite de student şi (2) lista cursurilor la care studentul este înscris.
O instanţă a clasei `Course` este un obiect ce reprezintă un curs şi păstrează o listă internă cu studenţii înregistraţi la cursul respectiv.

Fişierele de intrare sunt : `Studenti.txt` ce conţine o listă de studenţi şi `Cursuri.txt` ce conţine o listă de cursuri.
Fişierul `Studenti.txt` este orientat pe câmpuri şi foloseşte spaţiul ca separator.
Câmpul „Balanţă cont” conţine suma de bani de care dispune studentul.
Fiecare linie conţine informaţiile unui singur student şi are următoarea structură:

ID student	Nume şi prenume	Specializarea	Balanţă cont	Lista cursurilor absolvite
100234009	Popescu Ion	IS	3	13567 23324 21701

Fişierul `Cursuri.txt` este, de asemenea, orientat pe câmpuri separate prin spaţii.
Fiecare linie conţine o intrare corespunzătoare unui curs, cu următoarele câmpuri :

Număr curs	Zile	Ora început	Ora sfârşit	Instructor	Titlu curs
17655	JO	1800	1930	Mindruta	Arhitecturi software

Sistemul oferă o interfaţă rudimentară menu-text cu următoarele opţiuni pentru utilizator:

(1) Lista studenti – listarea studenţilor din sistem; informaţiile sunt preluate din fişierul `Studenti.txt`.

(2) Lista cursuri – listarea cursurilor din sistem; informaţiile sunt preluate din fişierul `Cursuri.txt`.

(3) Lista studenti înregistrati la un curs – sistemul cere utilizatorului să introducă ID-ul unui curs; sunt listaţi studenţii înregistraţi la cursul respectiv.

(4) Lista cursurilor la care este înregistrat un student – sistemul cere utilizatorului să introducă ID-ul unui student; listează cursurile la cere este înregistrat studentul respectiv.

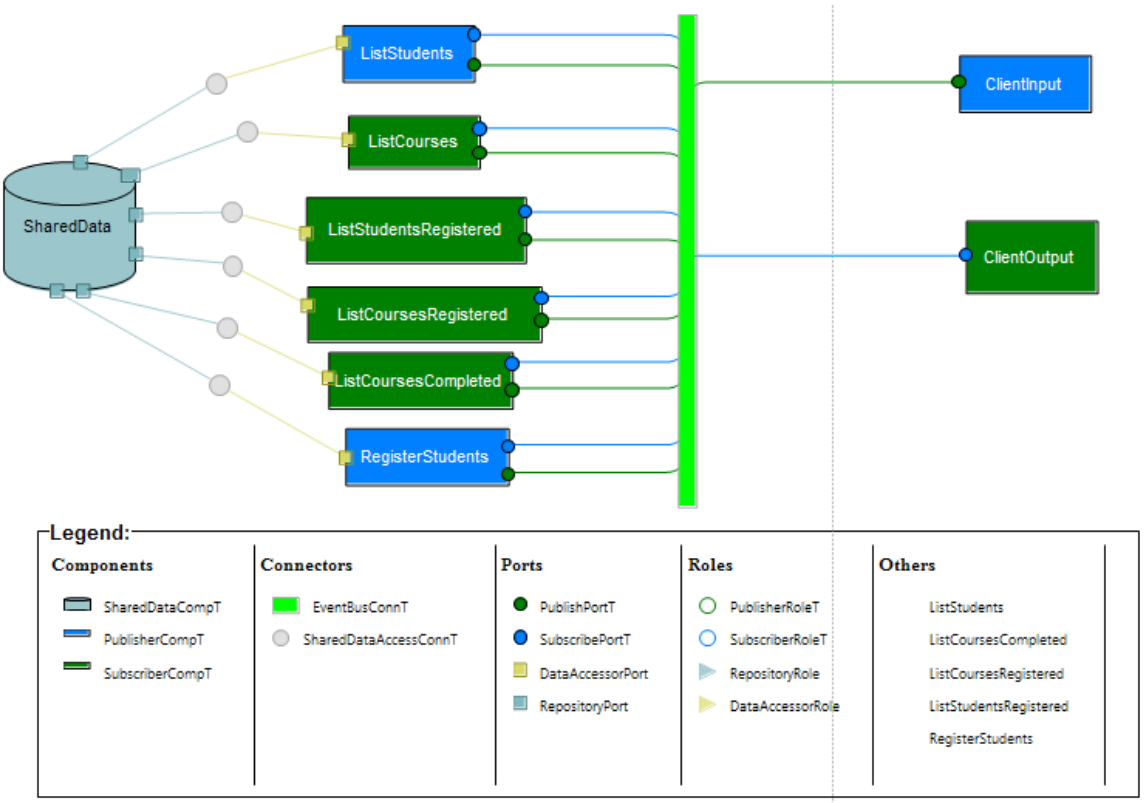
(5) Lista cursurilor absolvite de un student – sistemul cere ID-ul studentului; listează cursurile absolvite de acesta.

(6) Înregistrare student la un curs – sistemul cere ID student şi ID curs; sistemul adaugă cursul la lista de cursuri la care studentul s-a înregistrat şi adaugă studentul la lista studenţilor înregistraţi la curs; sistemul verifică conflictele de duplicare şi de planificare înainte de a face înregistrarea.

(X) Exit . Terminarea execuţiei programului.

Arhitectura sistemului de bază

Sistemul de bază are o arhitectură cu invocare implicită, ale cărei componente sunt implementate ca obiecte. Diagrama arhitecturii este reprezentată în figura următoare.



Funcţionalitatea sistemului este partiţionată, fiecare parte fiind încapsulată într-una dintre componente. Componentele fie difuzează (broadcast) evenimente de tip cereri pentru servicii către magistrala de evenimente, fie ascultă magistrala de evenimente pentru a oferi servicii altor componente. Observaţi că unele componente doar trimit notificări la magistrală (announce), altele doar recepţionează evenimente de la magistrală (listen), dar există componente care şi transmit şi recepţionează evenimente. O componentă de date partajate memorează starea partajată, formată din obiecte de tip `Student` şi `Course`. Aceasta este accesată de anumite componente prin conectori de acces la date de tip point-to-point (există 6 astfel de conectori în sistemul de bază).

Implementarea sistemului de tip „publish-subscribe" este realizată prin utilizarea claselor Java `Observer` şi `Observable`. Sistemul este iniţializat de către clasa `SystemMain`. Sistemul de bază conţine următoarele clase:

`SystemMain.java` – conţine metoda `main()` şi crează structura sistemului prin instanţierea tuturor componentelor şi lansarea componentei `ClientInput`.

`ClientInput.java` – prezintă menu-ul principal şi difuzează cererile de servicii către alte componente, pe baza intrărilor de la utilizator.

`ClientOutput.java` – Subscrie (se abonează) la şi recepţionează notificări de tip „afişează". Conţinutul acestor notificări este afişat pe consola utilizator.

`DataBase.java` – Oferă acces la listele de studenţi şi de cursuri. Oferă, de asemenea, metode pentru înregistrarea studenţilor la cursuri.

`EventBus.java` – Arhitectura cu invocare implicită este implementată utilizând clasele Java `Observer` şi `Observable`. Această clasă oferă componentelor suport pentru lucrul cu evenimente în regim de invocare implicită (înscriere interes pentru evenimente şi anunţare de notificări referitoare la apariţia de evenimente).

`CommandEventHandler.java` – Implementarea mecanismulului general de gestionare a evenimentelor.

XXX`Handler.java` – Implementarea unei componente care tratează un anumit eveniment de menu.

`Student.java` – Reprezintă un student.

`Course.java` – Reprezintă un curs.

De asemenea, sunt date ca model două fişiere cu date de intrare:
`Cursuri.txt` – fişier cu lista cursurilor
`Studenti.txt` – fişier cu lista studenţilor.

Compilarea şi execuţia sistemului

- Condiţii prealabile: J2SE (min.1.4.2) instalat.

- Compilarea fişierelor sursă Java:

```
...>javac *.java
```

- Pornirea sistemului:

```
...>java SystemMain Studenti.txt Cursuri.txt
```

TEMA

Partea 1 : Modificări la sistemul existent

Utilizaţi sistemul existent ca bază pentru crearea unui nou sistem cu capabilităţi suplimentare conform cerintelor de mai jos.
Va trebui să creaţi modelul arhitectural în Acme pentru noul sistem.
Va trebui să implementaţi cerinţele pentru noul sistem.

Obs.
1. Includeţi toate comentariile relevante.
2. Modelul arhitectural şi implementarea trebuie să fie sincronizate.

ATENŢIE !!!
Pe parcursul efectuări modificărilor păstraţi de câte ori este posibil alinierea la stilul cu invocare implicită. Modificări cum ar fi adăugarea de noi evenimente, adăugarea de noi componente şi schimbarea evenimentelor pe care le ascultă sau le generează o componentă păstrează stilul cu invocare implicită. Modificări la infrastructură, adăugarea de noi tipuri de conectori sau modificări la componentele existente conduc la riscul ca noul sistem să devieze de la arhitectura cu invocare implicită. Să ne amintim că se poate implementa un anumit stil peste o infrastructură corespunzătoare unui alt stil, cum ar fi implementarea unui stil din familia call-return utilizând evenimente. Cu fiecare dintre modificări, analizaţi dacă schimbarea sistemului este făcută „în spiritul" invocării implicite. Dacă nu, explicaţi deviaţiile, de ce le-aţi făcut şi consecinţele alegerilor pe care le-aţi făcut.

Cerinţele pentru sistemul nou

Se va păstra întreaga funcţionalitate a sistemului de bază, la care vor adăuga următoarele extensii:

1. Extindeţi sistemul cu suport pentru jurnalizare, prin adăugarea unei noi componente. Toate ieşirile transmise către ecran vor fi scrise şi într-un fişier de jurnalizare (log file). Analizaţi mai multe variante de implementare şi precizaţi care necesită codificare minimă şi care se potriveşte cel mai bine cu stilul arhitectural cu invocare implicită.
2. Adăugaţi sistemului o nouă capabilitate a.î. să se poată anunţa când un curs este suprasolicitat, fără a restricţiona însă următoarele înscrieri la cursul respectiv.
3. Extrageţi verificarea conflictelor de înscriere la cursuri din componenta `RegisterStudentHandler` şi plasaţi-o într-o componentă separată.

Comportamentul sistemului observabil de către utilizator nu trebuie să se modifice.

4. Creaţi o componentă prin care studentului i se va percepe o taxă odată cu înscrierea la un curs. În realizarea acestei componente se va ţine cont de următoarele constrângeri:
   - Dacă studentul nu are suficienţi bani în cont, atunci nu i se permite înscrierea la curs.
   - Dacă înregistrarea eşuează datorită conflictelor de orar a cursurilor, taxa nu va fi percepută.

   În general, va trebui să ţineţi cont de faptul că este posibilă anularea unei înscrieri sau a unei taxe dacă apare o problemă.

Tema scrisă.

1. Creaţi o descriere arhitecturală (i.e. diagrame) a sistemului modificat. Creaţi o vedere a sistemului din perspectivă dinamică. Creaţi o altă vedere ce reprezintă structurile statice de cod ale sistemului. Precizaţi ce tip de analiză se poate face pe fiecare dintre aceste vederi. Adăugaţi câte o explicaţie textuală pentru elementele din fiecare vedere. Adăugaţi orice vedere şi orice explicaţie textuală consideraţi că este necesară pentru a asigura claritatea şi neabiguitatea descrierii sistemului modificat.
2. Instrucţiuni cu modul de realizare şi de execuţie a sistemului modificat. Aceste instrucţiuni trebuie să fie suficiente pentru ca urmărindu-le să pot ajunge la rezultatul cerut.
3. Pentru noul sistem, asociaţi elementele arhitecturale (i.e. componente, conectori, porţi, roluri, legături (bindings), ierarhii) cu elementele de implementare corespunzătoare (i.e. variabile, metode, clase, fişiere). Consideraţi că dacă există elementele pentru care nu găsiţi o corespondenţă clară, aceasta semnifică faptul că ceva este greşit în arhitectură sau în implementare? Discutaţi motivele pentru care pot să apară aceste nepotriviri.

Partea 2 : Analiza sistemului – continuare la temă scrisă.

4. Discutaţi orice deviere de la stilul cu invocare implicită pur, identificată fie în sistemul original fie în cel modificat, atât din punct de vedere al proiectului arhitectural cât şi relativ la implementarea acestuia.
5. Descrieţi modul în care modificările făcute au afectat arhitectura sistemului. Discutaţi deciziile de proiectare critice, alternativele de proiectare pe care le-aţi considerat, şi justificările alegerilor pe care le-aţi făcut. Realizaţi această discuţie în termeni de concepte arhitecturale; explicitaţi atributele de calitate ce au ghidat realizarea arhitecturii şi priorităţile lor relative în luarea deciziilor arhitecturale.
6. Pe baza experienţei voastre, ce tipuri de modificări sunt uşor sau dificil de realizat în arhitecturile cu invocare implicită în comparaţie cu alte stiluri studiate în laboratoarele anterioare. Explicaţi de ce, utilizând exemple.
7. Presupuneţi că un sistem cu invocare implicită funcţionează prea lent, astfel încât pentru a creşte performanţa decideţi replicarea uneia dintre componente. Credeţi că pot să a apară condiţii de competiţie? Descrieţi cauzele posibile de apariţie a condiţiilor de competiţie. Descrieţi soluţii potenţiale pentru prevenirea acestei probleme şi comparaţi avantajele lor relative. Este acest tip de problemă inerent sistemelor cu invocare implicită?
8. Descrieţi pe scurt modificările ce ar fi necesare pentru a transforma sistemul nou într-un sistem distribuit, cu invocare implicită. Evaluaţi dificultatea realizării acestor modificări. Credeţi că se poate păstra stilul cu invocare implicită?

ANEXA
Model în detaliu al codului