LABORATOR 4 – Arhitecturi pentru sisteme software

Obiectiv – Înţelegerea familiei de stiluri arhitecturale call-return şi a relaţiei cu implementările OO. Experimentarea cu o arhitectură în 3 trepte (3-tiered) şi o strategie de implementare OO.

Sistemul de bază – aplicaţie pentru înregistrarea studenţilor la cursuri – dat 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. Un `Student` este un obiect ce menţine 2 liste interne : (1) lista cursurilor absolvite de student şi (2) lista cursurilor la care studentul este înscris.

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. Fiecare linie conţine un singur student şi are următoarea structură:

ID student	Nume şi prenume	Specializarea	Lista cursurilor absolvite
100234009	Popescu Ion	IS	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	Secţiune	Zile	Ora început	Ora sfârşit	Instructor	Titlu curs
17655	A	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.

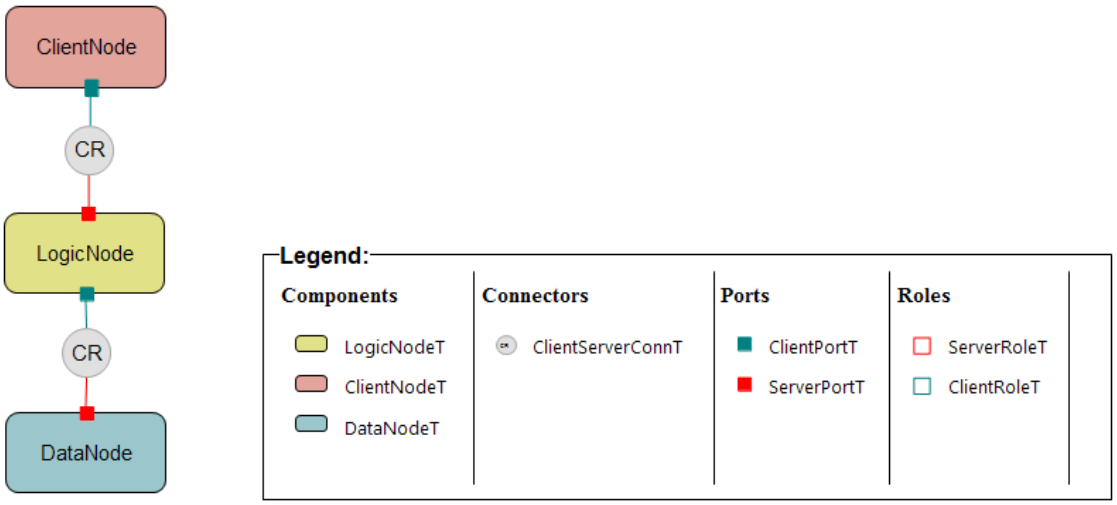
(5) Lista cursurilor absolvite de un student – sistemul cere ID-ul studentului; listează cursurile absolvite de acesta.

(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 în trei trepte, care este implementată (la nivel de detaliu) ca sistem orientat obiect. Cele trei trepte ale arhitecturii sunt reprezentate în figura următoare.



Fiecare treaptă este implementată ca un set de obiecte care oferă funcţionalitatea treptei. Conectivitatea între trepte este obţinută prin invocare de metodă la distanţă (RMI). Sistemul este implementat în Java şi cuprinde următoarele fişiere:

`Client.java` – conţine metoda `main()` pentru nodul `Client` din treapta client.

`Logic.java` – conţine metoda `main()` pentru nodul `Logic` din treapta cu logica aplicaţiei şi oferă acces de nivel înalt la baza de date.

`RILogic.java` – oferă interfaţa RMI pentru clasa `Logic`.

`Data.java` – conţine metoda `main()` pentru nodul `Data` din treapta datelor şi oferă acces la listele cu studenţi şi cu cursuri.

`RIData.java` – oferă interfaţa RMI pentru clasa `Data`.

`Student.java` – clasa ce reprezintă un student.

`Course.java` – clasa ce reprezintă un curs.

Suplimentar sunt date următoarele fişiere utilitare:
`BuiltSystem.bat` – fişier de comenzi ce compilează fişierele sursă şi construieşte stub şi skeleton RMI.
`StartSystem.bat` – fişier de comenzi ce lansează `rmiregistry` şi porneşte cele trei trepte.

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
```

- Crearea stub şi skeleton pentru RMI:

```
...>rmic Data
...>rmic Logic
```

Fişierul `BuildSystem.bat` conţine comenzile de mai sus.

Ca rezultat se vor crea fişierele `Data_Skel.class, Data_Stub.class, Logic_Skel.class, Logic_Stub.class`.

Acestea sunt fişiere produse de `rmic` (RMI compiler) şi utilizate pentru a invoca via RMI metode din fişierele de interfaţă.

- Sistemul este “construit” din următoarele fişiere:

`rmiregistry.exe` – permite înregistrarea serviciilor RMI şi oferă utilizatorilor acestora informaţii despre ce servicii sunt disponibile; este parte componentă a JRE (java runtime environment).
`Data.class` – construieşte treapta sistemului responsabilă cu gestionarea listelor de cursuri şi de studenţi.

`Logic.class` - construieşte treapta sistemului responsabilă cu servirea cererilor venite din treapta client; preia datele brute despre studenţi şi cursuri din treapta datelor şi oferă servicii (relativ complexe).

`Client.class` – construieşte treapta client a sistemului, care preia intrări de la utilizator şi afişează rezultatele pe terminal.

Pornirea sistemului se face astfel:

Lansare `rmiregistry`:
```
...>start rmiregistry
```

Lansarea treptei datelor
```
...>start java Data Studenti.txt Cursuri.txt
```

Lansarea treptei logicii aplicaţiei
```
...>start java Logic
```

Lansarea treptei Client
```
...>java Client
```

Toate aceste comenzi sunt grupate în fişierul `StartSystem.bat`.

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.

Cerinţele pentru sistemul nou

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

1. Distribuiţi treptele a.î. clientul să se execute pe o maşină separată. Treapta logică şi treapta datelor pot fi amplasate pe aceeaşi maşină. (Indicaţie: Citiţi specificaţia Java API pentru metoda `java.rmi.Naming.bind()`.)
2. Adăugaţi un nou client în treapta client a.î. cei doi clienţi să poată accesa simultan sistemul pentru înregistrare studenţi. Atenţie la posibilitatea apariţiei problemelor de competiţie privind accesul concurent la o resursă comună: va trebui să oferiţi suport pentru realizarea excluderii mutuale. De asemenea,

precizaţi clar la nivelul arhitecturii dacă adăugaţi un nou rol la conectorul existent sau creaţi un nou conector.

3. Adăugaţi o nouă facilitate care va realiza jurnalizare (log) prin memorare într-un fişier a istoricului interacţiunilor utilizatorilor. Conţinutul fişierului trebuie să poată fi citit utilizînd un editor de texte. Puteţi alege ce format doriţi, informaţiile furnizate pentru fiecare înregistrare fiind: identificare client, tipul comenzii utilizator şi o marcă de timp. Se presupune că există sute de clienţi activi şi că treapta logică şi treapta datelor nu pot fi oprite.

Tema scrisă.

1. Descrierea arhitecturală, din perspectivă C&C, a noului sistem : diagramele Acme şi legendele ataşate, codul Acme, explicaţiile textuale necesare înţelegerii corecte a arhitecturii.
2. Creaţi cel puţin o vedere modulară a sistemului (perspectiva statică) utilizând o metodă de reprezentare pe care aveţi libertatea să o alegeţi.
3. Instrucţiuni cu modul de realizare şi cu modul de execuţie a sistemului modificat. Aceste instrucţiuni trebuie să fie suficiente pentru ca urmărindu-le să pot ajunge la rezultatul cerut.

Partea 2 : Analiza sistemului – continuare la temă scrisă.

4. Evaluaţi şi discutaţi dificultatea relativă a realizării fiecărei modificări în raport cu celelalte modificări, pe baza experienţei voastre. Pentru fiecare modificare:
   - evaluaţi gradul de dificultate în raport cu celelalte modificări.
   - analizaţi în ce măsură gradul de dificultate este corelat cu faptul că stilul call-return avantajează modificarea respectivă.
   - analizaţi în ce măsură gradul de dificultate este corelat cu faptul că infrastructura a fost mai potrivită pentru modificarea respectivă în raport cu celelalte modificări.
5. Referitor la adăugarea jurnalizării, justificaţi alocarea noii funcţionalităţi la diferite trapte. Specificaţi dacă există alternative rezonabile la soluţia propusă de voi şi precizaţi motivele alegerii pe care aţi făcut-o.
6. Analizaţi relaţia dintre vederea C&C şi vederile modulare asupra arhitecturii sistemului. Există o corespondenţă clară între elementele arhitecturale din perspectivă dinamică şi elementele implementării ?
7. Discutaţi avantajele şi dezavantajele utilizării şablonului în trepte pentru această aplicaţie. Discutaţi diferenţa dintre treptele din arhitectura sistemului şi maşinile pe care se execută aceste trepte. Ce ar putea influenţa arhitectul în alocarea unei trepte la o anumită maşină ? În particular, ce anume ar putea influenţa alegerea locurilor unde se execută treapta logică şi treapta datelor ?

OBS: Pentru următoarele întrebări nu se cere şi implementarea.

8. Fie următorul text, generator al unui set de cerinţe noi:
   „Unii clienţi ar prefera să primească datele returnate în format XML în loc de format textual. De asemenea, se zvoneşte că în viitor va trebui ca sistemul să suporte clienţi bazaţi pe web”.

Consideraţi cel puţin două variante pentru obţinerea acestor capabilităţi. Analizaţi comparativ cele două soluţii.

9. Utilizând sistemul modificat ca bază pentru analiză, explicaţi cum îi puteţi îmbunătăţi securitatea. De exemplu, aţi putea considera utilizarea unei comunicări de date criptate între trepte pentru a preveni scurgerile de informaţii şi verificarea contului şi parolei utilizator pentru a preveni accesele nedorite. Schiţaţi arhitectura unui astfel de sistem şi răspundeţi la următoarele întrebări:

- Ce impact ar avea asupra arhitecturii modificările necesare securizării sistemului ?
- Discutaţi modificările arhitecturale, motivaţiile şi elementele implicate în securizarea sistemului.

Dacă credeţi că sistemul nu poate fi făcut mai sigur, justificaţi-vă opinia.

ANEXA

Model în detaliu al codului