Serializare

Tema 2

Responsabil: Caramizaru Horea

1. Enunt

Se considera un graf neorientat in care fiecare Nod se gaseste in unul dintre tipurile NodA, NodB, NodC.

Cele 3 clase (NodA, NodB, NodC) sufera modificari astfel incat avem 3 versiuni ale acestora. (ex. pentru clasa NodA avem versiunea 1,2 si 3)

Fiecare Nod contine **numele** nodului (string) si respectiv o modaliate de a mentine nodurile adiacente.

La iteratia 1 a celor 3 clase (NodA1, NodB1, NodC1) nodurile adiacente sunt mentinute folosind o lista (LIST) de adiacenta.

La iteratia 2 (NodA2, NodB2, NodC2) nodurile adiacente sunt mentinute folosind un vector (VECTOR)

La iteratia 3 (NodA3, NodB3, NodC3) nodurile sunt mentinute folosind un Set (SET)

Cele 3 clase (LIST / VECTOR / SET) vor fi implementate de catre studenti.

Pe acest graf se pot face urmatoarele operatii:

- Adaugare nod : Add NodA „Nume Nod” lista nume noduri adiacente
- Stergere nod : Del NodB „Nume Nod”
- Adaugare muchie : AddM „Nume nod” „Nume nod”
- Stergere muchie : DelM „Nume nod” „Nume nod”

Pentru a putea executa operatii se „incarca” in prealabil setarile care specifica versiunea de noduri utilizate de fiecare nod: Settings NodA NodB NodC

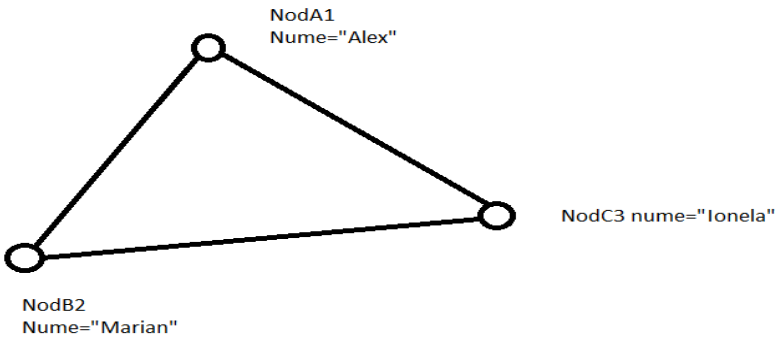
Ex. : Settings 1 2 1

Ordinea operatiilor:

Initial graful este gol si nu avem nicio setare activa. Prima comanda „incarca” o setare. Ulterior urmeaza o insiruire de comenzi aplicate grafului care se va termina cu o comanda de serializare de forma : Serialize Nume nod Nume fisier

Ex: Serialize „Ana” AnaSerializat.log

Pentru graful format din 3 noduri avem:



Serialize Alex

```
<Object class="NodA" Version="1" id="0">
    <Nume>Alex</Nume>
    <LIST>
        <Object class="NodB" Version="2" id="1">
                <Nume>Marian</Nume>
                <VECTOR>
                        <Reference class="NodA" Version="1" id="0">
                        <Object class="NodC" Version="3" id="2">
                                <Nume>Ionela</Nume>
                                <SET>
                                        <Reference class="NodB" Version="2" id="1">
                                        <Reference class="NodA" Version="1" id="0">
                                </SET>
                        </Object>
                </VECTOR>
        </Object>
    </LIST>
</Object>
```

Exista mai multe modalitati de a serializa; exemplul de mai sus foloseste o parcurgere in adancime asociata grafului. Nu sunteti limitati la o anumita ordine.

In urma serializarii nu neaparat toate nodurile vor fi salvate (pot exista noduri care nu au drum spre nodul de la care se incepe serializarea)

Pentru:

<Object class="NodA" Version="1" id="0">

„id” reprezinta un unic identificator pentru instanta serializata.

Avem { class, Version, id } ca si atribute.

„Reference” indica faptul ca acel camp indica catre un obiect care deja a mai fost serializat; atributul id specific indica acelas id ca si obiectul care a mai fost serializat.

Puteti sa adaugati atribute suplimentare daca acestea ajuta procesul de serializare / deserializare.

Deserializarea se executa la aparitia comenzii: Deserialize Nume Fisier

Ex: Deserialize AnaSerializat.log

Ordinea versiunilor celor 3 clase este $1 < 2 < 3$ de unde rezulta ca daca incercam sa „incarcam” o versiune mai noua de clasa a unei instante fata de setarile curente o problema ar trebui sa apara. In cazul in care incercam sa incarcam o versiune mai veche atunci un proces de adaptare al respectivei versiune are loc.

Lista operatiile de „cast” precizate mai sus vor fi salvate in fisierul: Deserialize_NumeFisier_CAST.log in formatul urmator:

OK/Fail cast NodA/NodB/NodC Nume from Version=”nr” to Version=”nr”

Ex:

OK cast NodA Iulia from Version=”1” to Version=”2”

Fail cast NodB Ionela from Version=”3” to Version=”2”

In cazul in care apare o operatie Fail, pe langa adaugarea mesajului, acel obiect va fi instantiat folosind versiunea curenta din setari.

Exemplu de operatii:

Settings 1 2 1

Add NodA Alex

Add NodB Marian Alex

Add NodC Ionela Marian Alex

Serialize Alex Alex_Fisier.txt

Settings 2 2 1

Deserialize AlexFisier.txt

La linia 5 are loc serializarea grafului curent iar un posibil rezultat al serializarii este exemplul de XML dat mai sus.

In cazul in care apare o operatie de deserializare graful mentinut in memorie este distrus.

2. Constrângeri

- Camptul **Nume** e format doar din litere si este identificator unic
- **Pentru realizarea temei nu vor fi utilizate librarii care fac facila operatia de serializare/deserializare.**
- Pentru realizarea temei trebuie folosită o versiune de Java nu mai recentă de Java 7
- numele fisierului de intrare se va da ca argument in linia de comanda (primul parametru)
- Pe langa implementarea efectiva, va trebui sa generati si un javadoc pentru clasele create, folosind comentarii in cod si generarea automata oferita de Netbeans / Eclipse, precum si un readme in care să explicati deciziile luate in implementarea temei si problemele intampinate.
- va trebui să aveti si un makefile cu o regula “run” care sa ruleze clasa si care contine metoda ”main” cat si o regula “build”.

3. Punctaj

Punctajul pe tema va consta din:

- 50% teste
- 40% coding style / design
- 10% readme + JavaDoc