Universitatea de Vest din Timişoara
Departamentul de Informatică Aritoni Ovidiu

Şabloane de Proiectare 2010-2011
Laborator R-2

Modelare XML

Scopul acestui laborator este de a construi un framework pentru operarea cu documente XML. Acest framework va fi utilizat în cadrul prototipului pentru evaluarea funcţiilor reale.
Composite, Visitor, Command, Singleton, Adapter,

1 Modelarea fişierelor XML

Un document XML poate fi văzut ca un arbore ale cărui noduri sunt elemente XML iar frunzele sunt atribute ale nodului respectiv. Un element poate conţine 0 sau n elemente, şi 0 sau n atribute. Un atribut este caracterizat printr-un cuplu (identificator, valoare). Un element este caracterizat prin numele tag-ului sau/şi un continut. Un element particular este identificat fie printr-o expresie de cale (la fel ca şi în UNIX), fie printr-un atribut particular ‚ID' a cărui valoare este unică. Mai jos este un exemplu de document XML şi reprezentarea sa arborescenta.

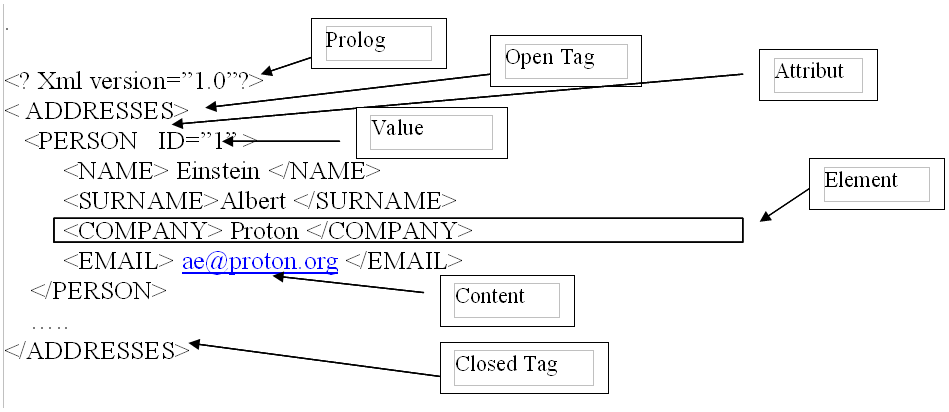

Figure 1: Structura unui document XML

Figure 2: Structura arborescentă a unui document XML

Documentul XML este alcătuit dintr-un sngur tag, numit tag-ul root. Acest tag este compus la rândul său din alte tag-uri, iar acestea pot fi compuse din alte tag-uri. Avem aşadar o structură de tip Composite, în modelarea fişierelor XML.
De asemenea asupra acestei structuri de tip Composite va trebui să aplicăm noi modificări.
Mai jos se află două programe Java.
Primul program încarcă dintr-un fişier XML o funcţie într-un arbore funcţional.

```
1  package xml;
2  import java.io.BufferedReader;
3  import java.io.FileReader;
4  import java.io.IOException;
5  import functii.functiiElementare.Constanta;
6  import functii.functiiElementare.Necunoscuta;
7  import functii.functiiTrigonometrice.Cos;
8  import functii.functiiTrigonometrice.Sin;
9  import operatori.Cat; import operatori.Minus;
10 import operatori.Plus;
11 import operatori.Produs;
12 import operatori.Putere;
13 import arbori.Arbore;
14 import arbori.Nod;
15
16 public class LoadXML{
17  protected Arbore a;
18  protected Nod n;
19  protected FileReader fstream;
20  protected BufferedReader in;
21  protected Array[] lista;
22  protected int dim;
23
24  public LoadXML(String fisier) throws IOException{
```

```
25      fstream = new FileReader(fisier);
26      in = new BufferedReader(fstream);
27      lista = new Array[100];
28      read();
29      creareArbore();
30      in.close();
31      a = new Arbore(lista[0].getNod());
32    }
33
34    public Arbore getArbore(){
35      return a;
36    }
37
38    public void read() throws IOException{
39      String sir = in.readLine();
40      int index=0, i = -1;
41      double  valoare=0;
42      sir = in.readLine();
43
44      while (sir != null){
45        if (sir.indexOf('/') == -1) {
46            i++;
47        index = getNivel(sir);
48             if(sir.indexOf("Constanta")!=-1)
49          valoare=getValoare(sir);
50        sir = prelucrareSir(sir);
51        if (sir.equals("Plus"))
52          n = new Plus();
53        else if (sir.equals("Minus"))
54          n = new Minus();
55        else if (sir.equals("Produs"))
56          n = new Produs();
57        else if (sir.equals("Putere"))
58         n = new Putere();
59        else if (sir.equals("Cat"))
60         n = new Cat();
61        else if (sir.equals("Sin"))
62         n = new Sin();
63        else if (sir.equals("Cos"))
64         n = new Cos();
65        else if (sir.equals("Necunoscuta"))
66         n = new Necunoscuta();
67        else if (sir.equals("Constanta")){
68          n = new Constanta();
69          ((Constanta)n).setCt(valoare);
```

```
70        }
71        lista[i] = new Array(n, index);
72       } //end-if
73     sir = in.readLine();
74     }//end-while
75     dim = i;
76     System.out.println();
77     System.out.println();
78     }
79
80   public String prelucrareSir(String sir) throws IOException{
81     sir = sir.trim();
82     int index = 0, index1;
83     if ((index1 = sir.indexOf(' ')) != -1)
84        index = index1;
85     else
86        index = sir.indexOf('>');
87     String sir2 = sir.substring(1, index);
88     return sir2;
89   }
90
91   public int getNivel(String sir) throws IOException{
92     sir = sir.trim();
93     int index = 0, index1, index2;
94     if ((index1 = sir.indexOf("id=\"")) != -1)
95         index = index1;
96     else
97       System.out.println("Eroare: Format xml gresit");
98     index2 = sir.substring(index + 4).indexOf('\"');
99     sir = sir.substring(index + 4, index + 4 + index2);
100    return Integer.parseInt(sir);
101  }
102
103  public double getValoare(String sir){
104    sir = sir.trim();
105    int index = 0, index1, index2;
106    if((index1 = sir.indexOf("val=\"")) != -1)
107      index = index1;
108    else
109      System.out.println("Eroare: Format xml gresit");
110    index2 = sir.substring(index + 5).indexOf('\"');
111    sir = sir.substring(index + 5, index + 5 + index2);
112    return Double.parseDouble(sir);
113  }
114
```

```
115  public void creareArbore(){
116    int nivel, j, nivel2, nivel1;
117    for(int i = 0; i <= dim; i++){//start-for-0
118       nivel = lista[i].getIndex();
119       nivel1 = lista[i].getIndex();
120       if(!lista[i].getNod().getClass().getName().equals("functii
             .functiiElementare.Constanta")
121         && !lista[i].getNod().getClass().getName().equals("
             functii.functiiElementare.Necunoscuta"))
122         {//start-if-0
123                for(j = i;j < dim && (nivel2 = lista[j].getIndex
                   ()) >= nivel1;j++)
124          {//start-for-1
125            if (lista[j].getIndex() == nivel + 1){//start-if-1
126                if (lista[i].getNod().getFStang() == null)
127                   {//start-if-2
128                       String nume=lista[j].getNod().getClass().
                          getName();
129                   if(
130                       nume.equals("operatori.Plus")||
131                       nume.equals("operatori.Minus")||
132                       nume.equals("operatori.Produs")||
133                       nume.equals("operatori.Putere")||
134                       nume.equals("operatori.Cat")||
135                       nume.equals("functii.functiiElementare.
                          Constanta")||
136                       nume.equals("functii.functiiElementare.
                          Necunoscuta")||
137                       nume.equals("functii.functiiTrigonometrice
                          .Sin")||
138                       nume.equals("functii.functiiTrigonometrice
                          .Cos")
139                     )
140                    lista[i].getNod().setFStang(lista[j].getNod()
                       );
141                    else
142                      System.out.println("error la stanga:"+lista
                         [i].getNod().getClass().getName());
143                }//end-if-2
144                else{//start-else
145                  String nume=lista[j].getNod().getClass().
                     getName();
146                  if(
147                     nume.equals("operatori.Plus")||
148                     nume.equals("operatori.Minus")||
```

```
149                     nume.equals("operatori.Produs")||
150                     nume.equals("operatori.Putere")||
151                     nume.equals("operatori.Cat")||
152                     nume.equals("functii.functiiElementare.
                            Constanta")||
153                     nume.equals("functii.functiiElementare.
                            Necunoscuta")||
154                     nume.equals("functii.functiiTrigonometrice.
                            Sin")||
155                     nume.equals("functii.functiiTrigonometrice.
                            Cos"))
156                      lista[i].getNod().setFDrept(lista[j].getNod
                            ()
157                    );
158                    else
159                       System.out.println("error la dreapta: "+
                             lista[i].getNod().getClass().getName())
                             ;
160                  }//end-else
161               }//end-if-1
162               nivel1 = nivel2;
163            }//end-for-1
164           if(lista[j].getIndex() == nivel + 1){//start-if-3
165             if (lista[i].getNod().getFStang() == null)
166                {//start-if-4
167                String nume=lista[j].getNod().getClass().getName
                       ();
168                if(
169                     nume.equals("operatori.Plus") ||
170                     nume.equals("operatori.Minus")||
171                     nume.equals("operatori.Produs")||
172                     nume.equals("operatori.Putere")||
173                     nume.equals("operatori.Cat")||
174                     nume.equals("functii.functiiElementare.
                            Constanta")||
175                     nume.equals("functii.functiiElementare.
                            Necunoscuta")||
176                     nume.equals("functii.functiiTrigonometrice.
                            Sin")||
177                     nume.equals("functii.functiiTrigonometrice.
                            Cos")
178                  )
179                   lista[i].getNod().setFStang(lista[j].getNod()
                         );
180                else
```

```
181                     System.out.println("error la stanga: "+lista[i
                            ].getNod().getClass().getName());
182               }//end-if-4
183               else{
184                   String nume=lista[j].getNod().getClass().getName
                          ();
185                   if(
186                       nume.equals("operatori.Plus")||
187                       nume.equals("operatori.Minus")||
188                       nume.equals("operatori.Produs")||
189                       nume.equals("operatori.Putere")||
190                       nume.equals("operatori.Cat")||
191                       nume.equals("functii.functiiElementare.
                              Constanta")||
192                       nume.equals("functii.functiiElementare.
                              Necunoscuta")||
193                       nume.equals("functii.functiiTrigonometrice.
                              Sin")||
194                       nume.equals("functii.functiiTrigonometrice.
                              Cos"))
195                       lista[i].getNod().setFDrept(lista[j].getNod()
                              );
196                  else
197                    System.out.println("error la dreapta: "+lista[i
                          ].getNod().getClass().getName());
198                  }
199             }//end-if-3
200           }//end-if-0
201       }//end-for-0
202   }//end-method
203 }//end-class
```

Următorul program salvează "un arbore" funcţional sub forma unui fişier XML.

```
 1 import arbori.Nod;
 2 public class SaveXML{
 3 protected FileWriter fstream;
 4 protected BufferedWriter out;
 5
 6  public SaveXML(Nod a, String fisier) throws IOException{
 7    fstream= new FileWriter(fisier);
 8    out = new BufferedWriter(fstream);
 9    out.write(" <?xml version=\"1.0\" encoding=\"ISO-8859-1\" ?>\
         n ");
10    save(a,1);
11    out.close();
```

```
12   }
13   public void save(Nod a, int nivel) throws IOException{
14     int i;
15     for(i=1;i<=nivel;i++)
16           out.write("   ");
17     int index=(a.getClass().getName()).lastIndexOf('.');
18     String fin=a.getClass().getName().substring(index+1);
19     if(fin.equals("Constanta"))
20         out.write("<"+fin+" id=\""+nivel+"\" val=\""+((Constanta)a
               ).getCt()+"\" >\n");
21     else
22         out.write("<"+fin+" id=\""+nivel+"\" >\n");
23     if(a.getFStang()!=null)
24       save(a.getFStang(),nivel+1);
25     if(a.getFDrept()!=null)
26       save(a.getFDrept(),nivel+1);
27     for(i=1;i<=nivel;i++)
28       out.write("   ");
29     out.write("</"+fin+">"+"\n");
30    }
31 }
```

Scopul acestui laborator este de modifica cele două programe de mai sus, pentru a construi un sistem mai flexibil de dezvoltare.

Task1 Realizaţi o diagramă de clasă care să modeleze urmăatoarea frază: “Un tag XML poate fi compus din zero, unul sau mai multe tag-uri XML. Un document XML conţine un singur tag “rădăcină”. Ce şablon de proiectare se va folosi pentru a exprima faptul că un tag XML este organizat conform principiului compunerii recursive ? Ce şablon de proiectare se va utiliza pentru a indica că orice document XML are un unic nod rădăcină ?

Task2 Modifcaţi această diagramă de clasă, adăugând semantica următoarei fraze: “Un tag XML poate conţine unul sau mai atribute. Orice atribut al unui tag XML este caracterizat printr-un nume şi o valoare”.

Task3 Adăugaţi diagramei de clasă obţinute anterior elementele necesare care rezultă din analiza următoarei fraze: “Orice tag XML are un nume şi de asemenea o valoare. De exemplu tag-ul cu numele “name” şi valoarea “Enstein”: <name>Enstein</name>. De asemenea orice nod simplu, adică care nu este nod de tip rădăcină, are un nod părinte. Implementaţi în Java diagrama de clasă obţinută.

Task4 Utilizaţi şablonul de proiectare Command pentru a indica faptul că un tag XML poate efectua următoarele operaţii: adăugarea / ştergerea unui nod fiu/atribut. Aceste operaţii se pot efectua asupra unui atribut al nodului-tag curent, dar şi asupra unui nod-tag fiu al nodului curent.

Task5 Implementati şablonul de proiectare Visitor pentru a parcurge arborele de noduri-tag. Acest Visitor va fi utilizat în procesul de afişare a documentului XML.

Task6 Operaţia iplementată de Visitorul de mai sus va fi utilizată pentru a salva arborele funcţional într-un fişier XML. Utilizaţi şablonul de proiectare Adapter pentru a adapta acest vizitor construit la operaţia de salavare în fişier XML.

Task7 Dezvoltaţi un visitor care va inspecta nodurile arborelui funcţional şi va crea pornind de la acesta un arbore de taguri XML. Aşadar pentru a putea permite transformarea unui arbore funcţional într-un document XML se vor aplica doi visitori: primul care transformă arborele funcţional în arbore XML, şi alt vizitor "adaptat" care "afişează" acest arbore XML într-un document XML.

Task8 De asemenea va trebui construit un alt visitor care va construi arborele funcţional pornind de la documentul XML. Acest vizitor va inspecta arborele XML.

Task9 Creaţi o clasă care să modeleze un document XML. Aceasta va avea ca şi atribute locaţia din sistemul de fişiere unde se află stocat acel document XML, respectiv elementul rădăcină.

Task10 Utilizând şablonul de proiectare Command adăugaţi următoarele acţiuni care se pot efectua cu documente XML: creare/suprimarea unui element, crearea / suprimarea unui atribut, schimbarea valorii unui atribut sau element, deplasarea unui element în alt element, afişarea unui document XML la terminal, scrierea unui document XML

Task11

2 Maparea XML a funcţiilor reale

Prototipul de evaluare a funcţiilor reale dezvoltat în cadrul laboratorului trecut utilizează o reprezentare arborescentă pentru funcţiile reale. De exemplu pentru funcţia $f(x) = x + sin(x) \cdot tg(x/ln(x))$ vom avea următoarea reprezentare arborescentă: