```
1  /*
2   * Copyright DSTC Pty.Ltd. (http://www.dstc.com), Technische Universitaet Darmstadt
3   * (http://www.tu-darmstadt.de) and the University of Queensland (http://www.uq.edu.au).
4   * Please read licence.txt in the toplevel source directory for licensing information.
5   *
6   * $Id: XMLSyntaxError.java 749 2003-02-11 09:44:52Z peterbecker $
7   */
8  package net.sourceforge.toscanaj.util.xmlize;
9
10 @SuppressWarnings("serial")
11 public class XMLSyntaxError extends Exception {
12     public XMLSyntaxError(Throwable cause) {
13         super(cause);
14     }
15
16     public XMLSyntaxError() {
17         super();
18     }
19
20     public XMLSyntaxError(String reason, Throwable cause) {
21         super(reason, cause);
22     }
23
24     public XMLSyntaxError(String reason) {
25         super(reason);
26     }
27 }
```

```java
/*
 * Copyright DSTC Pty.Ltd. (http://www.dstc.com), Technische Universitaet Darmstadt
 * (http://www.tu-darmstadt.de) and the University of Queensland (http://www.uq.edu.au).
 * Please read licence.txt in the toplevel source directory for licensing information.
 *
 * $Id: XMLizable.java 549 2002-10-28 02:46:23Z peterbecker $
 */
package net.sourceforge.toscanaj.util.xmlize;

import org.jdom.Element;


/**
 * This serialization interface is used for the model. All
 * elements of the model support this interface and in addition have a
 * constructor that callls readXML.
 *
 * @todo separate this aspect of the code from the model. Turning the model into XML
 *     and back should not be part of the models interface but of some other classes,
 *     maybe using introspection and some kind of mapping information.
 */
public interface XMLizable {

    /**
     * Write this element as the content of elem.
     */
    public abstract Element toXML();

    /**
     * Read this element as the content of elem.
     */
    public abstract void readXML(Element elem) throws XMLSyntaxError;
}
```

```
1  package es.us.ccia.dummy.teoria;
2
3  /**
4   * Version 0.9:
5   *    20080807
6   *    − AdaptaciÃ³n del calculo de relaciones a RCC5
7   *
8   *    20080713
9   *    − CLASE PRINCIPAL DE LA APLICACION.
10  *      Aqui se encuentra el trozo de la ontologia
11  *      que va a ser nuestra teoria a estudiar.
12  *
13  *================================================
14  * TODO:
15  */
16
17
18 import java.util.ArrayList;
19 import java.util.Iterator;
20 import java.util.LinkedList;
21 import java.util.Stack;
22
23 import es.us.ccia.dummy.contexto.elementos.FCAAttribute;
24 import es.us.ccia.dummy.contexto.elementos.FCAElement;
25 import es.us.ccia.dummy.contexto.elementos.FCAObject;
26 import es.us.ccia.dummy.historial.Cambio;
27 import es.us.ccia.dummy.historial.H_Cambios8;
28 import es.us.ccia.dummy.historial.ListaCambios;
29 import es.us.ccia.dummy.razonador.*;
30 import es.us.ccia.dummy.restricciones.*;
31 //import es.us.ccia.dummy.teoria.movimientos.*;
32 import es.us.ccia.dummy.teoria.regiones.*;
33 import es.us.ccia.dummy.teoria.relaciones.*;
34 import es.us.ccia.rcc.RCC;
35
36
37
38 /***
39  *
40  * @author  Gonzalo A. Aranda Corral
41  * @version 0.9.0
42  * @since   13/07/2008
43  *
44  */
45 @SuppressWarnings("unchecked")
46 public class Teoria { // implements I_PosicionesRelativas {
47
48
49         // Argumentos a estudiar
50         private ArrayList          LClases;
51         private ArrayList          LIndividuos;
52         private Stack<ListaRelacionRCC> PilaRelacRCC;
53
54         private Stack<ListaElementos> LRegiones;
55
56         private LinkedList<ListaCambios> LCambios;
57
58
59
60         /***
61          * Constructor de la clase. Aqui se inicializa la teoria
62          *
63          * Hay que distinguir entre el contexto completo y los
64          * argumentos a tratar
65          *
66          *
67          */
68     public Teoria() {
69
70         // Son las clases de los argumentos a estudiar
71         this.LClases         = new ArrayList();
72                 this.LIndividuos     = new ArrayList();
73                 this.PilaRelacRCC    = new Stack<ListaRelacionRCC>();
74                 this.LRegiones       = new Stack<ListaElementos>();
75                 this.LCambios        = new LinkedList<ListaCambios>();
76
77                 addListaRegiones(new ListaElementos());
78                 addListaRelaciones(new ListaRelacionRCC());
79
80         }
81
82
```

```
83
84      //=======================================================
85      // *   METODOS DE CLASES
86      //=======================================================
87
88          public void insertarClase(FCAAttribute nombre) {
89                  LClases.add(nombre);
90          }
91
92          public void insertarClases(ArrayList<FCAAttribute> lista) {
93                  Iterator i = lista.iterator();
94                  while (i.hasNext()) {
95                          insertarClase((FCAAttribute)i.next());
96                  }
97          }
98
99          public ArrayList<FCAAttribute> getListaClases() {
100                 return this.LClases;
101         }
102
103         public void limpiarClases() {
104                 LClases.clear();
105         }
106
107         public int numClases() {
108                 return LClases.size();
109         }
110
111
112         //=======================================================
113     // *   METODOS DE INDIVIDUOS
114     //=======================================================
115         public void insertarIndividuo(FCAObject nombre) {
116                 LIndividuos.add(nombre);
117         }
118
119         public void insertarIndividuos(ArrayList<FCAObject> lista) {
120                 Iterator i = lista.iterator();
121                 while (i.hasNext()) {
122                         insertarIndividuo((FCAObject)i.next());
123                 }
124         }
125
126         public ArrayList<FCAObject> getListaIndividuos() {
127                 return this.LIndividuos;
128         }
129
130         public void limpiarIndiv() {
131                 LIndividuos.clear();
132         }
133
134         public int numIndividuos() {
135                 return LIndividuos.size();
136         }
137
138
139
140
141     //=======================================================
142     // *   R E L A C I O N E S     RCC8
143     //=======================================================
144         // Esto estÃ¡ desarrollado en forma de pila para poder
145         // DESHACER LOS CAMBIOS.
146         //
147
148         public void addListaRelaciones(ListaRelacionRCC lr) {
149                 PilaRelacRCC.push(lr);
150         }
151
152         public void insertarRelacionRCC(Relacion r) {
153                 PilaRelacRCC.peek().add(r);
154         }
155
156         public ListaRelacionRCC getListaRelacionesRCC() {
157                 return PilaRelacRCC.peek();
158         }
159
160         public void limpiarRelacionesRCC() {
161                 PilaRelacRCC.peek().clear();
162         }
163
164
```

```
165         //=========================================================
166     // *   PASO DE ARGUMENTOS(ELEGIDOS) A RELACIONES RCC
167     //=========================================================
168
169         //Desde Seleccion (Argumentos)
170         public void Argumentos_a_RCC() {
171
172                 // INICIALIZACION
173                 limpiarRelacionesRCC();
174
175
176                 // RELACIONES ENTRE TODOS. No es necesario separarlos
177                 // ya que el razonador distingue entre unas y otras.
178
179                 ArrayList total = new ArrayList(this.getListaClases());
180                 total.addAll(this.getListaIndividuos());
181
182                 Object[] clases = total.toArray();
183
184                 for (int i=0;i<clases.length;i++) {
185                         FCAElement a1 = (FCAElement)clases[i];
186                         for (int j=0;j<i;j++) {
187                                 FCAElement a2 = (FCAElement)clases[j];
188                                 this.insertarRelacionRCC(
189                                                 new Relacion(Razonador.getRelacionRCC(a1, a2),
190                                                                 a1.getNombre(),a2.getNombre())
    );
191                         }
192                 }
193         }
194
195
196         //=========================================================
197     // *   PASO DEL DIBUJO A RELACIONES RCC
198     //=========================================================
199         //DesdeDibujo
200 //      public void Dibujo_a_RCC8() {
201 //              ListaElementos listaTotal = getListaRegiones();
202 //              int longitud = listaTotal.size();
203 //              getListaRelacionesRCC().clear();
204 //
205 //                      for(int i=0;i<longitud−1;i++) {
206 //                      for(int j=i+1;j<longitud;j++) {
207 //                              Elemento e1 = (Elemento)listaTotal.get(i);
208 //                              Elemento e2 = (Elemento)listaTotal.get(j);
209 //                              Relacion rr = new Relacion(
210 //                                              Posiciones.PosicionRCC8(e1, e2),
211 //                                              e1.getArgumento(), e2.getArgumento());
212 //                              insertarRelacionRCC(rr);
213 //                      }
214 //              }
215
216 //      }
217
218
219     //=========================================================
220     // *   R E G I O N E S
221     //=========================================================
222         public void insertarRegion(String nombre,int x1,int y1,int x2,int y2) {
223                 Region t = new Region(nombre,x1,y1,x2,y2);
224                 ListaElementos el = LRegiones.peek();
225                 el.add(t);
226         }
227
228         public void insertarPunto(String nombre,int x1,int y1) {
229                 Punto t = new Punto(nombre,x1,y1);
230                 ListaElementos el = LRegiones.peek();
231                 el.add(t);
232         }
233
234         public ListaElementos getListaRegiones() {
235                 if (LRegiones.isEmpty())
236                         return null;
237                 else
238                         return this.LRegiones.peek();
239         }
240
241         public void limpiarRegiones() {
242                 ListaElementos el = LRegiones.pop();
243                 el.clear();
244                 this.LRegiones.push(el);
245         }
```

```
246
247
248         public void calcularRegiones() {
249                 if (getListaRelacionesRCC().size()>0) {
250
251                         //                      simplificarRelacionesRCC();
252                         limpiarRegiones();
253                         ResolverCHR res = new ResolverCHR();
254                         res.resolv();
255                 }
256         }
257
258         public void addListaRegiones(ListaElementos _le) {
259                 this.LRegiones.push(_le);
260         }
261
262         public void undoListaRegiones() {
263                 ListaElementos el = this.LRegiones.pop();
264                 if (LRegiones.isEmpty()) {
265                         LRegiones.push(el);
266                 }
267         }
268
269         public void nuevaListaRegiones() {
270                 this.LRegiones.push(new ListaElementos());
271         }
272
273         public void duplicarListaRegiones() {
274                 ListaElementos el = LRegiones.peek();
275
276                 ListaElementos nueva = new ListaElementos();
277                 for (Iterator it = el.iterator();it.hasNext();) {
278                         Elemento e = (Elemento)it.next();
279                         if (e.getTipo()== I_Elemento.PUNTO) {
280                                 nueva.add(new Punto((FCAObject)e.getArgumento(),e.getX1(),e.ge
    tY1()));
281                         } else {
282                                 nueva.add(new Region((FCAAttribute)e.getArgumento(),
283                                                 e.getX1(),e.getY1(),
284                                                 ((Region)e).getX2(),((Region)e).getY2()));

285                         }
286                 }
287
288
289                 this.LRegiones.push(nueva);
290         }
291
292
293
294         //=========================================================
295     // *   PROCEDIMIENTO DE SIMPLIFICACIONES
296     //=========================================================
297
298         public void simplificarRelacionesRCC() {
299                 PilaRelacRCC.peek().simplificarListaRCC();
300         }
301
302
303         //=========================================================
304     // *   PROCEDIMIENTO DE CAMBIOS
305     //=========================================================
306
307         public LinkedList<ListaCambios> getLCambios() {
308                 return this.LCambios;
309         }
310
311         public void addListaCambios(ListaCambios _c) {
312                 LCambios.add(_c);
313         }
314
315
316
317
318         public void undoCambiosRelaciones() {
319
320                 if (!LCambios.isEmpty())
321                         LCambios.removeLast();
322
323
324         }
325
```

```
326
327 //      Lo que hay que hacer ahora es poner algo en la ventana de cambios...
328
329
330         public void duplicarListaRelaciones() {
331                 ListaRelacionRCC el = PilaRelacRCC.peek();
332
333                 ListaRelacionRCC nueva = new ListaRelacionRCC();
334                 for (Iterator it = el.iterator();it.hasNext();) {
335                         Relacion e = (Relacion)it.next();
336                         nueva.add(new Relacion(e.getTipo(),e.getA(),e.getB()));
337                 }
338
339                 this.PilaRelacRCC.push(nueva);
340         }
341
342
343
344         @Override
345         public String toString() {
346
347                 StringBuffer devolver = new StringBuffer();
348                 devolver.append("==================================\n");
349                 devolver.append("= T E O R I A =\n");
350                 devolver.append("==================================\n");
351
352                 devolver.append("Lista de clases\n");
353
354                 Iterator c = this.LClases.iterator();
355                 while (c.hasNext()) {
356                         FCAAttribute clase = (FCAAttribute)(c.next());
357                         devolver.append(clase);
358                         devolver.append("\n");
359                 }
360
361                 devolver.append("\n");
362
363                 devolver.append("Lista de individuos\n");
364
365                 Iterator o = this.LIndividuos.iterator();
366                 while (o.hasNext()) {
367                         FCAObject ind = (FCAObject)(o.next());
368                         devolver.append(ind);
369                         devolver.append("\n");
370                 }
371
372                 devolver.append("==================================\n");
373
374
375
376                 return devolver.toString();
377         }
378
379
380
381 }
382
383
```

```
1  package es.us.ccia.dummy.contexto.entsal;
2
3  import java.io.File;
4  import java.io.FileWriter;
5  import java.io.IOException;
6
7  import es.us.ccia.dummy.contexto.Contexto;
8  import es.us.ccia.dummy.contexto.elementos.FCAAttribute;
9  import es.us.ccia.dummy.contexto.elementos.FCAElement;
10 import es.us.ccia.dummy.contexto.elementos.FCAObject;
11 import es.us.tad.Salida;
12
13 public class Grabar {
14
15
16
17
18         public static void formatoCXT(String fichero) {
19
20                 StringBuffer texto = new StringBuffer("");
21
22
23                 // PRimero ESCRiBIR ETIQUETAS
24                 Object[] et1 = Contexto.getAttributes().toArray();
25                 for(int j=0;j<et1.length;j++) {
26                         texto.append(((FCAElement)et1[j]).getNombre() + "\t");
27                 }
28                 texto.append("\n");
29
30
31                 Object[] ob1 = Contexto.getObjects().toArray();
32                 for (int i=0;i<ob1.length;i++) {
33
34                         // Escribir el objeto
35                         FCAObject objeto = (FCAObject)ob1[i];
36                         texto.append(objeto.getNombre() + "\t");
37
38                         for(int j=0;j<et1.length;j++) {
39                                 FCAAttribute etiqueta = (FCAAttribute)et1[j];
40                                 if (Contexto.existsRelation(etiqueta, objeto)) {
41                                         texto.append("1\t");
42                                 } else {
43                                         texto.append("0\t");
44                                 }
45
46                         }
47                         texto.append("\n");
48                 }
49                 String resultado = texto.toString();
50
51                 File f = new File(fichero);
52                 FileWriter fw;
53                 try {
54                         fw = new FileWriter(f,false);
55                         fw.write(resultado);
56                         fw.close();
57                 } catch (IOException e) {
58                         e.printStackTrace();
59                 }
60
61                 Salida.Imprimirln("Contexto grabado en formato CXT");
62         }
63
64
65
66
67
68
69
70
71
72
73
74
75 }
```

```
1  package es.us.ccia.dummy.contexto.entsal;
2
3  import java.util.StringTokenizer;
4
5  import javax.swing.JOptionPane;
6
7  import del.icio.us.Delicious;
8  import del.icio.us.beans.Post;
9  import del.icio.us.beans.Tag;
10 import es.us.ccia.dummy.contexto.Contexto;
11
12
13 public class Importar {
14
15
16         private Delicious del;
17
18         private String  usuario;
19         private String  clave;
20
21
22         public Importar(String _usuario, String _clave) {
23 ////     Delicioso d = new Delicioso("jaalonso","Wu_Wei_08");
24 ////     Delicioso d = new Delicioso("garanda","libertad32");
25                 this.usuario = _usuario;
26                 this.clave   = _clave;
27         }
28
29
30         public boolean importar() {
31                 boolean devolver = true;
32
33                 try {
34                         sacarDelicious();
35
36                 } catch (Exception e) {
37                         devolver = false;
38 /*
39                         Poner un mensaje para cuando no se pueda importar desde
40                         Delicious
41
42
43
44                         Hay que probar a recuperar el contexto de Jalonso...
45                         y grabarlo dentro de un fichero.
46
47
48                         Crear un usuario para las pruebas de Delicious..
49                         con 6 etiquetas y 10 sitios
50
51
52                         Mirar como hacer el tema de las preferencias...
53                         de guardarlas y recuperarlas.
54
55
56                         Poner las preferencias, me imagino como estaticas...
57                         y que el tipo de razonamiento sea un argumento para la
58                         mayoria de las cosas
59
60
61                         Otra cosa a hacer... es el razonamiento del dibujo... incremental.
62                         lo que ya esta pintado no se vuelve a recalcular... por lo que
63                         podemos ganar un mundo en efectividad
64
65
66                         * deseables:
67                                 el tema del RCC8... con la frontera, que hay que analizar
68                                 a todos los demas
69
70                                 podemos limitar el razonamiento de dummy a RCC5, ya que el
71                                 razonamiento sobre conceptos creo que es mas sencillo.
72                                 dos conceptos estan EC cuando interseccionan SOLOpor debajo
73                                 del soporte.
74
75                                 Otro tema es el TPP(A,B) es cuando existe un concepto C talque
76                                 EC(A,C) y EC(B,C)
77                                 Esto significa, ver si lo hacemos antes de empezar a razonar..
   .
78                                 o lo hacemos on the fly!
79
80                         En principio, aunque le pongamos como parametro... TODO RCC5
```

```
81
82
83
84                          Otra cosa en la que podria pensar... es
85
86                            EMPEZAR A ESCRIBIR!!...
87
88
89                               LA TESIS...
90
91
92                         OTRA COSA.. es los dos CAEPIAS!!
93
94
95                         pensar tambien.. como podriamos integrar la creacion de reticulos...
96
97
98
99                         Pero sobre todo.. TERMINAR DUMMY PERFECTAMENTE!!
100        */
101
102
103                          System.out.println("Some problems were detected importing data.");
104                 }
105             return devolver;
106         }
107
108
109
110         /*****************************************
111          *
112          * Comienzo de los metodos privados
113          *
114          *****************************************/
115         private void sacarDelicious() throws Exception {
116                 Post[]  objetos;
117                 Tag[]   etiquetas;
118
119             /** CONEXION CON DELICIOUS **/
120                 del = new Delicious(this.usuario,this.clave);
121
122
123             /** EXTRAER ETIQUETAS **/
124             Object [] t = del.getTags().toArray();
125             etiquetas = new Tag[t.length];
126             for(int i=0; i<t.length; i++) {
127                 etiquetas[i]=(Tag)t[i];
128                 Contexto.createAttribute(etiquetas[i].getTag());
129             }
130
131             /** ESTABLECER RETRASO **/
132
133             /** EXTRAER OBJETOS **/
134                 Object [] o = del.getAllPosts().toArray();
135             objetos = new Post[o.length];
136             for(int i=0; i<objetos.length; i++) {
137                 objetos[i]=(Post)o[i];
138                 Contexto.createObject(objetos[i].getDescription());
139
140                 String et1 = objetos[i].getTag();
141                         StringTokenizer st = new StringTokenizer(et1," ");
142                         while(st.hasMoreTokens()) {
143                                 Contexto.addRelation(objetos[i].getDescription(), (String)st.n
    extToken());
144                         }
145             }
146
147         }
148
149 }
```

```
1  package es.us.ccia.dummy.contexto.entsal;
2
3  import java.util.*;
4
5  import es.us.ccia.dummy.contexto.Contexto;
6
7
8  @SuppressWarnings("unchecked")
9  public class Entrada {
10
11
12         public static void analizarContenido(String contenido) {
13                 String cabecera = contenido.substring(0, contenido.indexOf("\n"));
14                 String cuerpo   = contenido.substring(contenido.indexOf("\n")+1);
15
16                 Contexto.inicializar("por defecto");
17
18                 /***  SACAR LAS ETIQUETAS  ***/
19                 LinkedList Letiq = new LinkedList();
20                 StringTokenizer st = new StringTokenizer(cabecera,"\t");
21                 while (st.hasMoreElements()) {
22                         String etiqueta = (String)st.nextElement();
23                         Contexto.createAttribute(etiqueta);
24                         Letiq.add(etiqueta);
25                 }
26
27                 /***  SACAR LOS OBJETOS  ***/
28                 StringTokenizer st2 = new StringTokenizer(cuerpo,"\n");
29                 while (st2.hasMoreElements()) {
30
31                         String linea = st2.nextToken();
32                         if (linea.trim().length() > 0) {
33
34                                 StringTokenizer st3 = new StringTokenizer(linea,"\t");
35                                 // Este es el nombre del objeto
36                                 String objeto = st3.nextToken();
37                                 Contexto.createObject(objeto);
38
39                                 int iEtiq = 0;
40
41                                 while (st3.hasMoreElements()) {
42                                         String cadena = (String)st3.nextElement();
43                                         if (cadena.trim().length()>0) {
44                                                 int elemento = Integer.parseInt(cadena);
45                                                 if (elemento == 0) {
46
47                                                 } else {
48                                                         String etiqueta = (String)Letiq.get(iE
   tiq);
49                                                         Contexto.addRelation(etiqueta, objeto)
   ;
50                                                 }
51                                         }
52                                         iEtiq++;
53                                 }
54                         }
55
56                 }
57
58 //              System.out.println("Contexto cargado: " + Contexto.getName() + "\n"
59 //                              + Contexto.imprimir());
60
61         }
62
63
64 }
```

```
1  package es.us.ccia.dummy.contexto.elementos;
2
3  import java.util.LinkedList;
4
5  public class FCAAttribute extends FCAElement {
6
7          private LinkedList<FCAObject> Lista;
8
9          public FCAAttribute(String _nombre, int _pos) {
10                 super(_nombre, _pos);
11                 this.Lista    = new LinkedList<FCAObject>();
12
13         }
14
15
16         public void setRelacion(FCAObject _elemento) {
17                 this.Lista.add(_elemento);
18         }
19
20
21         public LinkedList<FCAObject> getRelaciones() {
22                 return (LinkedList<FCAObject>) this.Lista.clone();
23         }
24
25         public boolean isRelacionado(FCAObject _e) {
26                 return this.Lista.contains(_e);
27         }
28
29
30
31
32
33
34
35
36
37 }
```

```
1  package es.us.ccia.dummy.contexto.elementos;
2  import java.util.LinkedList;
3
4  import org.jdom.Element;
5
6  import net.sourceforge.toscanaj.util.xmlize.XMLSyntaxError;
7  import net.sourceforge.toscanaj.util.xmlize.XMLizable;
8
9  public abstract class FCAElement implements XMLizable {
10
11         private String                Nombre;
12         private int                   Posicion;
13 //      private int                   tipo;
14         private String                NombreVar;
15
16
17         public FCAElement(String _nombre, int _pos) {
18                 this.Nombre   = _nombre;
19                 this.Posicion = _pos;
20                 this.NombreVar = "var" + Posicion;
21         }
22
23         public String getNombre() {
24                 return this.Nombre;
25         }
26
27         public void setNombre(String _nombre) {
28                 this.Nombre = _nombre;
29         }
30
31 //      public void setRelacion(FCAElement _elemento) {
32 //              this.Lista.add(_elemento);
33 //      }
34 //
35 //
36 //      public LinkedList<FCAElement> getRelaciones() {
37 //              return this.Lista;
38 //      }
39 //
40 //      public boolean isRelacionado(FCAElement _e) {
41 //              return this.Lista.contains(_e);
42 //      }
43
44
45         public int getPosicion() {
46                 return Posicion;
47         }
48
49         public void setPosicion(int posicion) {
50                 this.Posicion = posicion;
51         }
52
53
54         public String getNombreVar() {
55                 return this.NombreVar;
56         }
57
58         public void setNombreVar(String _nvar) {
59                 this.NombreVar = _nvar;
60         }
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
```

```
83
84
85
86
87          @Override
88          public String toString() {
89                  return Nombre;
90          }
91
92          @Override
93          public boolean equals(Object obj) {
94                  if (obj instanceof FCAElement) {
95                          return Nombre.equalsIgnoreCase(((FCAElement) obj).getNombre());
96                  } else
97                          return false;
98          }
99
100         public void readXML(Element elem) throws XMLSyntaxError {
101                 // TODO Auto-generated method stub
102
103         }
104
105         public Element toXML() {
106                 // TODO Auto-generated method stub
107                 return null;
108         }
109
110 }
```

```
1  package es.us.ccia.dummy.contexto.elementos;
2
3  import java.util.LinkedList;
4
5  public class FCAObject extends FCAElement {
6
7          private LinkedList<FCAAttribute> Lista;
8
9          public FCAObject(String _nombre, int _pos) {
10                 super(_nombre, _pos);
11                 this.Lista = new LinkedList<FCAAttribute>();
12         }
13
14         public void setRelacion(FCAAttribute _elemento) {
15                 this.Lista.add(_elemento);
16         }
17
18
19         public LinkedList<FCAAttribute> getRelaciones() {
20                 return this.Lista;
21         }
22
23         public boolean isRelacionado(FCAAttribute _e) {
24                 return this.Lista.contains(_e);
25         }
26
27
28 }
```

```
1  package es.us.ccia.dummy.contexto;
2
3  import java.util.*;
4
5  import es.us.ccia.dummy.contexto.elementos.FCAAttribute;
6  import es.us.ccia.dummy.contexto.elementos.FCAElement;
7  import es.us.ccia.dummy.contexto.elementos.FCAObject;
8
9
10 @SuppressWarnings("unchecked")
11 public class Contexto {
12
13         private String Name;    // Nombre del Contexto
14         private Hashtable<String,FCAAttribute> Atributos;
15         private Hashtable<String,FCAObject> Objetos;
16         private int NElementos;
17         private Vector<FCAElement> FCAVector;
18
19         private static Contexto contexto;
20
21
22
23         public static void inicializar(String _name) {
24                 contexto = new Contexto(_name);
25         }
26
27
28
29
30
31         /**
32          * Creador del contexto
33          * @param _name Le pasamos el nombre del contexto
34          */
35         private Contexto(String _name) {
36                 Name       = _name;
37                 Atributos  = new Hashtable<String,FCAAttribute>();
38                 Objetos    = new Hashtable<String,FCAObject>();
39                 NElementos = 0;
40                 FCAVector  = new Vector<FCAElement>();
41         }
42
43         /**
44          * Recupera la coleccion de lo s nombres de los atributos
45          * @return
46          */
47         public static Collection getAttributes() {
48
49                 Collection devolver = new LinkedList();
50
51                 for (Iterator i = contexto.Atributos.values().iterator();i.hasNext();) {
52                         devolver.add(((FCAAttribute)i.next()).getNombre());
53                 }
54
55
56
57                 return devolver;
58         }
59
60         /**
61          * Recupera la coleccion de los nombres de los objetos
62          * @return
63          */
64         public static Collection getObjects() {
65
66                 Collection devolver = new LinkedList();
67
68                 for (Iterator i = contexto.Objetos.values().iterator();i.hasNext();) {
69                         devolver.add(((FCAObject)i.next()).getNombre());
70                 }
71
72                 return devolver;
73         }
74
75         /**
76          * Recupera el nombre del Contexto
77          * @return
78          */
79         public static String getName() {
80                 return contexto.Name;
81         }
82
```

```
83
84          /**
85           * Crea un atributo o etiqueta y lo aÃ±ade al contexto
86           * @param _nombre
87           */
88          public static void createAttribute(String _nombre) {
89                  FCAAttribute e = new FCAAttribute(_nombre,contexto.NElementos++);
90                  contexto.FCAVector.add(e);
91                  contexto.Atributos.put(_nombre,e);
92          }
93
94          /**
95           * Crea un objeto y lo aÃ±ade al contexto.
96           * No aÃ±ade valores.
97           * @param _nombre
98           */
99          public static void createObject(String _nombre) {
100                 FCAObject e = new FCAObject(_nombre,contexto.NElementos++);
101                 contexto.FCAVector.add(e);
102                 contexto.Objetos.put(_nombre,e);
103         }
104
105
106         /**
107          * A&amp;ntilde;ade una relacion entre entre _n1 y _n2
108          * @param _n1
109          * @param _n2
110          * @return devuelve True si la relacion se ha podido
111          */
112         public static boolean addRelation(String _at,String _ob) {
113
114                 if (Contexto.existsAttribute(_at)) {
115                         FCAAttribute n1 = (FCAAttribute)contexto.Atributos.get(_at);
116                         FCAObject    n2 = (FCAObject)contexto.Objetos.get(_ob);
117                         return Contexto.addRelation(n1, n2);
118                 } else {
119                         FCAObject    n1 = (FCAObject)contexto.Objetos.get(_at);
120                         FCAAttribute n2 = (FCAAttribute)contexto.Atributos.get(_ob);
121                         return Contexto.addRelation(n2, n1);
122                 }
123
124         }
125
126         /**
127          * A&amp;ntilde;ade una relacion entre entre _n1 y _n2
128          * @param _n1
129          * @param _n2
130          * @return devuelve True si la relacion se ha podido
131          */
132         public static boolean addRelation(FCAAttribute _at,FCAObject _ob) {
133                 boolean devolver = false;
134
135                 if ((_at !=null) && (_ob != null)) {
136                         _at.setRelacion(_ob);
137                         _ob.setRelacion(_at);
138                         devolver = true;
139                 }
140
141                 return devolver;
142         }
143
144
145         /**
146          * Comprueba si existe el atributo de nombre _n1
147          * @param _n1
148          * @return
149          */
150         public static boolean existsAttribute(String _n1) {
151                 return contexto.Atributos.containsKey(_n1);
152         }
153
154         /**
155          * Comprueba si existe el objeto de nombre _n1
156          * @param _n1
157          * @return
158          */
159         public static boolean existsObject(String _n1) {
160                 return contexto.Objetos.containsKey(_n1);
161         }
162
163         /**
164          * Recupera el atributo de nombre _n1
```

```
165             * @param _n1
166             * @return
167             */
168            public static FCAAttribute getAttribute(String _n1) {
169                    return (FCAAttribute) contexto.Atributos.get(_n1);
170            }
171
172            /**
173             * Recupera el objeto _n1 por su nombre
174             * @param _n1
175             * @return
176             */
177            public static FCAObject getObject(String _n1) {
178                    return (FCAObject) contexto.Objetos.get(_n1);
179            }
180
181            /**
182             * Recupera el Elemento(Atributo u objeto) de numero _n1
183             * @param _n1
184             * @return
185             */
186            public static FCAElement getElement(int _n1) {
187                    return (FCAElement)contexto.FCAVector.get(_n1);
188            }
189
190            public static boolean isAttribute(String a) {
191                    return contexto.Atributos.get(a) != null;
192            }
193
194            public static boolean isObject(String a) {
195                    return contexto.Objetos.get(a) != null;
196            }
197
198
199
200            public static boolean existsRelation(String _at, String _ob) {
201
202                    FCAAttribute a = getAttribute(_at);
203                    FCAObject o    = getObject(_ob);
204
205                    return Contexto.existsRelation(a, o);
206            }
207
208
209            public static boolean existsRelation(FCAAttribute _at, FCAObject _ob) {
210
211                    if ((_at!=null)&&(_ob!=null)){
212                            return _at.isRelacionado(_ob);
213                    } else
214                            return false;
215
216            }
217
218
219
220            public static LinkedList<FCAObject> get_ObjetosClase(FCAAttribute nombre) {
221                    return contexto.getObjetosClase(nombre);
222            }
223
224            private LinkedList<FCAObject> getObjetosClase(FCAAttribute nombre) {
225                    return nombre.getRelaciones();
226            }
227
228
229
230            public static LinkedList<FCAAttribute> get_ClasesObjeto(FCAObject nombre) {
231                    return contexto.getClasesObjeto(nombre);
232            }
233
234            private LinkedList<FCAAttribute> getClasesObjeto(FCAObject nombre) {
235                    return nombre.getRelaciones();
236            }
237
238
239            /**
240             * Esto es para imprimir de una forma decente
241             * @return devuelve la cadena a imprimir.
242             */
243            public static String imprimir() {
244    //              String devolver="\t";
245
246    //              /** Escribir las etiquetas **/
```

```
247 //              Object[] ve = Contexto.getAttributes().toArray();
248 //
249 //              for(int i=0; i<ve.length; i++) {
250 //                      devolver += ve[i] + "\t";
251 //              }
252 //              devolver += "\n";
253 //
254 //              Object[] ob = Contexto.getObjects().toArray();
255 //              for(int j=0; j<ob.length; j++) {
256 //                      String obj = (String)ob[j];
257 //                      devolver += obj + "\t";
258 //
259 //                      for(int i=0; i<ve.length; i++) {
260 //
261 //                              String etiq = (String)ve[i];
262 //
263 //                              if (Contexto.existsRelation(etiq, obj)) {
264 //                                      devolver += "X\t";
265 //                              } else {
266 //                                      devolver += "\t";
267 //                              }
268 //
269 //                      }
270 //                      devolver += "\n";
271 //
272 //              }
273
274 //
275 //              /***
276 //               * Vamos a tratar de escribirlo de una forma alternativa...
277 //               *
278 //               *  − ponemos en cada objeto las etiquetas que tiene.
279 //               */
280 //
281                 StringBuffer devolver= new StringBuffer();
282 //
283 //              Object[] ob = Contexto.getObjects().toArray();
284 //              for(int j=0; j<ob.length; j++) {
285 //                      String obj = (String)ob[j] + ": ";
286 //                      devolver.append(obj);
287 //
288 //                      FCAObject oo = Contexto.getObject(obj);
289 //
290 //                      if (oo != null) {
291 //                              Collection relaciones = oo.getRelaciones();
292 //                              for (Iterator i=relaciones.iterator();i.hasNext();) {
293 //                                      String et = ((FCAAttribute)i.next()).getNombre();
294 //                                      devolver.append(et + ", ");
295 //                              }
296 //
297 //                              devolver.append("\n");
298 //                      } else {
299 //                              devolver.append("objeto nulo\n");
300 //                      }
301 //
302 //              }
303 //
304
305
306 //              for(int i=0; i<ve.length; i++) {
307
308
309                 Enumeration e = contexto.Atributos.elements();
310
311                 while (e.hasMoreElements()) {
312                         FCAAttribute gg = (FCAAttribute)e.nextElement();
313                         devolver.append(gg.getNombre() + ":");
314
315
316                         if (gg.getRelaciones() != null) {
317                                 Iterator ob = gg.getRelaciones().iterator();
318                                 while (ob.hasNext()) {
319                                         FCAObject obj = (FCAObject)ob.next();
320                                         devolver.append(obj.getNombre() + ",");
321                                 }
322
323
324
325                         } else {
326                                 devolver.append("no tiene relaciones.");
327                         }
328                         devolver.append("\n");
```

```
329
330
331                     }
332
333
334
335
336 //                  devolver.append(contexto.Atributos.toString() + "\n");
337 //                  devolver.append(contexto.Objetos.toString() + "\n");
338
339
340
341
342
343
344
345
346
347
348
349
350
351                     return devolver.toString();
352             }
353
354 }
```

```
1  package es.us.ccia.dummy.restricciones.analizador;
2
3  /**
4   * Version 0.9:
5   *    20080713
6   *    − Grupos lexicos. Hemos aÃ±adido el guion bajo al
7   *      grupo de las letras, para que las pueda reconocer
8   *      como parte de los nombres.
9   *
10  *================================================
11  * TODO:
12  *    − Borrar lo que sobra... que es mucho.
13  */
14
15 /***
16  *
17  * @author  Gonzalo A. Aranda Corral
18  * @version 0.9.0
19  * @since   13/07/2008
20  *
21  */
22 public class Grupos {
23
24
25
26         /************************************************
27          *
28          * GRUPOS LEXICOS DE LOS LETRAS
29          *
30          ************************************************/
31         public boolean es_L(char x){
32                 return ((x>='a') && (x<='z'))
33                         || ((x>='A') && (x<='Z'))
34                         || (x=='_');
35         }
36
37
38
39         /************************************************
40          *
41          * GRUPOS LEXICOS DE LOS DELIMITADORES
42          *
43          ************************************************/
44         public boolean es_del(char x) {
45                 return
46                         es_del_sinsignificado(x) ||
47                         es_fininstruccion(x) ||
48                         es_operador(x) ||
49                         es_llave(x) ||
50                         es_operadorrelacional(x) ||
51                         es_parentesis(x) ||
52                         es_corchete(x) ||
53                         es_dospuntos(x) ||
54                         es_punto(x) ||
55                         es_corte(x) ||
56                         es_coma(x) ||
57                         es_Comilla(x);
58         }
59
60         public boolean es_del_sinsignificado(char x) {
61                 return (x =='\n') || (x ==' ') || (x =='\t') || (x =='\r');
62         }
63         public boolean es_fin_linea(char x) {
64                 return (x =='\n');
65         }
66
67         public boolean es_fininstruccion(char x) {
68                 return (x ==';');
69         }
70
71         public boolean es_Comilla(char x){
72                 return (x=='"');
73         }
74
75
76
77
78
79
80         /************************************************
81          *
82          * GRUPOS LEXICOS DE LOS OPERADORES
```

```
83              *
84              ************************************************/
85
86      public boolean es_operador(char x) {
87              return es_operadormas(x) || es_operadormenos(x) || es_operadordiv(x) || es_operadormult(x) || es_operadorexpo(x);
88      }
89              public boolean es_operadormas(char x) {
90                      return (x ==’+’);
91              }
92              public boolean es_operadormenos(char x) {
93                      return (x ==’−’);
94              }
95
96              public boolean es_operadormult(char x) {
97                      return (x ==’*’);
98              }
99
100             public boolean es_operadordiv(char x) {
101                     return (x ==’/’);
102             }
103
104             public boolean es_operadorexpo(char x) {
105                     return (x ==’^’);
106             }
107
108     /************************************************
109      *
110      * GRUPOS LEXICOS DE LOS OPERADORES RELACIONALES
111      *
112      ************************************************/
113     public boolean es_operadorrelacional(char x) {
114             return
115             es_operadormayor(x) ||
116             es_operadormenor(x) ||
117             es_operadorigual(x);
118     }
119
120     public boolean es_operadormayor(char x) {
121             return (x ==’>’);
122     }
123
124     public boolean es_operadormenor(char x) {
125             return (x ==’<’);
126     }
127
128     public boolean es_operadorigual(char x) {
129             return (x ==’=’);
130     }
131
132
133
134
135     /************************************************
136      *
137      * GRUPOS LEXICOS DE LAS LLAVES Y BLOQUES
138      *
139      ************************************************/
140
141     public boolean es_llave(char x) {
142             return es_llaveabierta(x) || es_llavecerrada(x);
143     }
144             public boolean es_llaveabierta(char x) {
145                     return (x ==’{’);
146             }
147
148             public boolean es_llavecerrada(char x) {
149                     return (x ==’}’);
150             }
151
152     public boolean es_parentesis(char x) {
153             return es_parentesisabierto(x) || es_parentesiscerrado(x);
154     }
155             public boolean es_parentesisabierto(char x) {
156                     return (x ==’(’);
157             }
158
159             public boolean es_parentesiscerrado(char x) {
160                     return (x ==’)’);
161             }
162
163
164     public boolean es_corchete(char x) {
```

```
165                     return es_corcheteabierto(x) || es_corchetecerrado(x);
166             }
167                     public boolean es_corcheteabierto(char x) {
168                             return (x ==’[’);
169                     }
170
171                     public boolean es_corchetecerrado(char x) {
172                             return (x ==’]’);
173                     }
174
175
176
177             public boolean es_punto(char x){
178                     return (x ==’.’);
179             }
180
181
182
183
184     /**
185      *
186      *
187      * GRUPOS LEXICOS DE LOS NUMEROS
188      *
189      *
190      *
191      *
192      */
193             public boolean es_N(char x){
194                     return (x>=’0’) && (x<=’9’);
195             }
196
197             public boolean es_NH(char x){
198                     return (this.es_N(x)) || ((x>=’a’) && (x<=’f’)) || ((x>=’A’) && (x<=’F’));
199             }
200
201             public boolean es_NB(char x){
202                     return ((x>=’0’) && (x<=’1’));
203             }
204
205             public boolean es_NO(char x){
206                     return ((x>=’0’) && (x<=’7’));
207             }
208
209             public boolean es_CERO(char x){
210                     return (x ==’0’);
211             }
212
213             public boolean es_sepBIN(char x){
214                     return (x ==’x’) || (x ==’X’);
215             }
216
217             public boolean es_sepHEX(char x){
218                     return (x ==’h’) || (x ==’H’);
219             }
220
221             public boolean es_sepOCT(char x){
222                     return (x ==’o’) || (x ==’O’);
223             }
224
225             public boolean es_simboloE(char x){
226                     return (x ==’e’) || (x ==’E’);
227             }
228
229             public boolean es_signodecimal(char x){
230                     return (x ==’.’);
231             }
232
233             public boolean es_dospuntos(char x){
234                     return (x ==’:’);
235             }
236
237             public boolean es_coma(char x){
238                     return (x ==’,’);
239             }
240
241             public boolean es_guion(char x){
242                     return (x ==’−’);
243             }
244
245             public boolean es_corte(char x){
246                     return (x ==’!’);
```

```
247                 }
248
249
250
251 }
```

```
1  package es.us.ccia.dummy.restricciones.analizador;
2  /**
3   * Version 0.9:
4   *    20080713
5   *    − ConstrucciÃ³n del automata de reconocimiento.
6   *      Esta basado en el completo de Compiladores y
7   *      se puede mirar un poco mas a fondo.
8   *
9   *================================================
10  * TODO:
11  *    − Optimizacion de las transiciones.
12  *    − Claridad en la entrada salida.
13  */
14
15 import java.util.*;
16
17 import es.us.tad.Salida;
18
19 /***
20  *
21  * @author  Gonzalo A. Aranda Corral
22  * @version 0.9.0
23  * @since   13/07/2008
24  *
25  */
26 public class Automata {
27
28         private String entrada;
29         private ArrayList salida;
30
31         /***
32          * Constructor de la clase
33          * @param Entrada Mensaje de entrada
34          * @param Salida  Lista de tokens de salida
35          */
36         public Automata(String Entrada,ArrayList Salida) {
37                 this.entrada = Entrada;
38                 this.salida  = Salida;
39         }
40
41         /***
42          * Arrancar el automata reconocedor
43          * @return True si es correcto, False en caso contrario.
44          */
45         public boolean analizar() {
46
47                 int estado = 0;
48                 char Caracter;
49                 int indice = 0;
50
51                 Grupos miGrupo = new Grupos();
52                 ArrayList auxiliar = new ArrayList();
53                 boolean devolver = true;
54
55                 this.salida.clear();
56
57                 while(indice < entrada.length()) {
58
59                         Caracter = entrada.charAt(indice++);
60
61                         switch( estado ) {
62                                 case 0:
63                                         /* Acciones semanticas */
64                                         auxiliar.clear();
65
66                                         /* Transiciones desde el estado */
67                                         if (miGrupo.es_del_sinsignificado(Caracter)) {
68                                                 estado = 0;
69
70                                         } else if (miGrupo.es_N(Caracter)) {
71                                                 estado = 1;
72                                         } else if (miGrupo.es_L(Caracter)) {
73                                                 estado = 10;
74
75
76
77                                         } else if (miGrupo.es_fininstruccion(Caracter)) {
78                                                 GeneraToken(";");
79
80                                         } else if (miGrupo.es_coma(Caracter)) {
81                                                 GeneraToken(",");
82
```

```
83                              } else if (miGrupo.es_punto(Caracter)) {
84                                      GeneraToken(".");
85
86                              } else if (miGrupo.es_Comilla(Caracter)) {
87                                      estado = 20;
88
89                              } else if (miGrupo.es_operadorigual(Caracter)) {
90                                      estado = 21;
91
92                              } else if (miGrupo.es_operadormas(Caracter)) {
93                                      estado = 22;
94
95                              } else if (miGrupo.es_operadormenos(Caracter)) {
96                                      estado = 23;
97
98                              } else if (miGrupo.es_operadormult(Caracter)) {
99                                      GeneraToken("*");
100
101                             } else if (miGrupo.es_operadordiv(Caracter)) {
102                                     estado = 100;
103
104                             } else if (miGrupo.es_operadorexpo(Caracter)) {
105                                     GeneraToken("^");
106
107                             } else if (miGrupo.es_operadormayor(Caracter)) {
108                                     estado = 24;
109
110                             } else if (miGrupo.es_operadormenor(Caracter)) {
111                                     estado = 25;
112
113                             } else if (miGrupo.es_corcheteabierto(Caracter)) {
114                                     GeneraToken("[");
115
116                             } else if (miGrupo.es_corchetecerrado(Caracter)) {
117                                     GeneraToken("]");
118
119                             } else if (miGrupo.es_llaveabierta(Caracter)) {
120                                     GeneraToken("{");
121
122                             } else if (miGrupo.es_llavecerrada(Caracter)) {
123                                     GeneraToken("}");
124
125                             } else if (miGrupo.es_parentesisabierto(Caracter)) {
126                                     GeneraToken("(");
127
128                             } else if (miGrupo.es_parentesiscerrado(Caracter)) {
129                                     GeneraToken(")");
130
131                             } else if (miGrupo.es_corte(Caracter)) {
132                                     GeneraToken("!");
133
134                             } else if (miGrupo.es_dospuntos(Caracter)) {
135                                     GeneraToken(":");
136
137                             } else {
138                                     ErrorLexico();
139                                     estado = 0;
140                             }
141                             break;
142
143                     case 1:
144                             /* Transiciones desde el estado */
145                             if (miGrupo.es_N(Caracter)) {
146                                     estado = 1;
147                             } else if (miGrupo.es_punto(Caracter)) {
148                                     estado = 2;
149
150                             } else if (miGrupo.es_del(Caracter)) {
151
152                                     GeneraToken(aCadena(auxiliar));
153                                     indice--;
154                                     estado = 0;
155                             } else {
156                                     ErrorLexico();
157                                     auxiliar.clear();
158                                     estado = 0;
159                             }
160                             break;
161
162                     case 2:
163                             /* Acciones semanticas */
164
```

```
165                                     /* Transiciones desde el estado */
166                                     if (miGrupo.es_N(Caracter)) {
167                                             estado = 2;
168
169                                     } else if (miGrupo.es_del(Caracter)) {
170                                             GeneraToken(aCadena(auxiliar));
171                                             indice--;
172                                             estado = 0;
173
174                                     } else {
175                                             ErrorLexico();
176                                             auxiliar.clear();
177                                             estado = 0;
178                                     }
179                                     break;
180
181                             case 10:
182                                     /* Acciones semanticas */
183
184                                     /* Transiciones desde el estado */
185                                     if (miGrupo.es_N(Caracter)) {
186                                             estado = 11;
187                                     } else if (miGrupo.es_L(Caracter)) {
188                                             estado = 10;
189
190                                     } else if (miGrupo.es_punto(Caracter)) {
191                                             GeneraToken(aCadena(auxiliar));
192                                             indice--;
193                                             estado = 0;
194
195                                     } else if (miGrupo.es_parentesisabierto(Caracter)) {
196                                             GeneraToken(aCadena(auxiliar));
197                                             indice--;
198                                             estado = 0;
199
200                                     } else if (miGrupo.es_corcheteabierto(Caracter)) {
201                                             GeneraToken(aCadena(auxiliar));
202                                             indice--;
203                                             estado = 0;
204
205                                     } else if (miGrupo.es_del(Caracter)) {
206                                             GeneraToken(aCadena(auxiliar));
207                                             indice--;
208                                             estado = 0;
209                                     } else {
210                                             ErrorLexico();
211                                             auxiliar.clear();
212                                             estado = 0;
213                                     }
214                                     break;
215
216                             case 11:
217                                     /* Acciones semanticas */
218
219                                     /* Transiciones desde el estado */
220                                     if (miGrupo.es_N(Caracter)) {
221                                             estado = 11;
222                                     } else if (miGrupo.es_L(Caracter)) {
223                                             estado = 11;
224                                     } else if (miGrupo.es_punto(Caracter)) {
225                                             GeneraToken(aCadena(auxiliar));
226                                             indice--;
227                                             estado = 0;
228
229                                     } else if (miGrupo.es_parentesisabierto(Caracter)) {
230                                             GeneraToken(aCadena(auxiliar));
231                                             indice--;
232                                             estado = 0;
233
234                                     } else if (miGrupo.es_corcheteabierto(Caracter)) {
235                                             GeneraToken(aCadena(auxiliar));
236                                             indice--;
237                                             estado = 0;
238
239                                     } else if (miGrupo.es_del(Caracter)) {
240                                             GeneraToken(aCadena(auxiliar));
241                                             indice--;
242                                             estado = 0;
243                                     } else {
244                                             ErrorLexico();
245                                             auxiliar.clear();
246                                             estado = 0;
```

```
247                         }
248                         break;
249
250                 case 20:
251                         /* Acciones semanticas */
252
253                         /* Transiciones desde el estado */
254                         if (!(miGrupo.es_Comilla(Caracter))) {
255                                 // OJO! si NO es COMILLA
256                                 estado = 20;
257                         } else {
258                                 //auxiliar.Elimina();
259                                 GeneraToken(aCadena(auxiliar));
260                                 estado = 0;
261                         }
262                         break;
263
264                 case 21:
265                         /* Acciones semanticas */
266
267                         /* Transiciones desde el estado */
268                         if (miGrupo.es_operadorigual(Caracter)) {
269                                 GeneraToken("==");
270                                 estado = 0;
271                         } else {
272                                 GeneraToken("=");
273                                 indice--;
274                                 estado = 0;
275                         }
276                         break;
277
278                 case 22:
279                         /* Acciones semanticas */
280
281                         /* Transiciones desde el estado */
282                         if (miGrupo.es_operadormas(Caracter)) {
283                                 GeneraToken("++");
284                                 estado = 0;
285
286                         } else {
287                                 GeneraToken("+");
288                                 indice--;
289                                 estado = 0;
290                         }
291                         break;
292
293                 case 23:
294                         /* Acciones semanticas */
295
296                         /* Transiciones desde el estado */
297                         if (miGrupo.es_operadormenos(Caracter)) {
298                                 GeneraToken("--");
299                                 estado = 0;
300
301                         } else {
302                                 GeneraToken("-");
303                                 indice--;
304                                 estado = 0;
305                         }
306                         break;
307
308                 case 24:
309                         /* Acciones semanticas */
310
311                         /* Transiciones desde el estado */
312                         if (miGrupo.es_operadorigual(Caracter)) {
313                                 GeneraToken(">=");
314                                 estado = 0;
315
316                         } else {
317                                 GeneraToken(">");
318                                 indice--;
319                                 estado = 0;
320                         }
321                         break;
322
323                 case 25:
324                         /* Acciones semanticas */
325
326                         /* Transiciones desde el estado */
327                         if (miGrupo.es_operadorigual(Caracter)) {
328                                 GeneraToken("<=");
```

```
329                                                     estado = 0;
330
331                                             } else {
332                                                     GeneraToken("<");
333                                                     indice--;
334                                                     estado = 0;
335                                             }
336                                             break;
337
338                                     case 100:
339                                             if (miGrupo.es_operadormult(Caracter)) {
340                                                     // Comentarios de bloque
341                                                     estado = 101;
342                                             } else if (miGrupo.es_operadordiv(Caracter)) {
343                                                     // Comentarios de linea
344                                                     estado = 110;
345                                             } else {
346                                                     GeneraToken("/");
347                                                     indice--;
348                                                     estado = 0;
349                                             }
350                                             break;
351
352                                     case 101:
353                                             if (miGrupo.es_operadormult(Caracter)) {
354                                                     estado = 102;
355                                             }
356                                             break;
357
358                                     case 102:
359                                             if (miGrupo.es_operadordiv(Caracter)) {
360                                                     estado = 0;
361                                             } else if (miGrupo.es_operadormult(Caracter)) {
362                                                     estado = 102;
363                                             } else {
364                                                     estado = 101;
365                                             }
366                                             break;
367
368                                     case 110:
369                                             if (miGrupo.es_fin_linea(Caracter)) {
370                                                     // Comentarios de bloque
371                                                     estado = 0;
372                                             }
373                                             break;
374                             }
375                             auxiliar.add(Caracter+"");
376
377                     } // Este es el final del analisis del buffer;
378
379 //                  salida.IrInicio();
380                     return devolver;
381             }
382
383             /***
384              * AÃ±ade el contenido(como token) al buffer de salida
385              * @param contenido
386              */
387             private void GeneraToken(String contenido) {
388                     this.salida.add(contenido);
389             }
390
391             /***
392              * Tratamiento de errores.
393              */
394             private void ErrorLexico(){
395                     Salida.Imprimirln("Error!");
396             }
397
398             /***
399              * Conversion de un ArrayList de caracteres en una cadena
400              * @param cadena ArrayList de caracteres
401              * @return       String con la cadena correspondiente
402              */
403             public String aCadena(ArrayList cadena) {
404                     String devolver = "";
405                     Iterator i = cadena.iterator();
406                     while (i.hasNext()) {
407                             devolver += (String)i.next();
408                     }
409                     return devolver;
410
```

```
411             }
412
413 }
```

```
1  package es.us.ccia.dummy.restricciones.analizador;
2  /**
3   * Version 0.9:
4   *    20080713
5   *    − Clase de analisis sintactico de la salida y
6   *      actualizacion de la teoria con el resultado.
7   *
8   *================================================
9   * TODO:
10  *    − Optimizar el analisis y la devolucion.
11  *    − quizas no se deba actualizar la teoria desde aqui
12  *      para garantizar la modularidad.
13  *
14  */
15
16 import es.us.ccia.dummy.*;
17 import es.us.ccia.dummy.contexto.Contexto;
18 import es.us.ccia.dummy.contexto.elementos.FCAAttribute;
19 import es.us.ccia.dummy.contexto.elementos.FCAObject;
20 import es.us.ccia.dummy.teoria.*;
21
22 /***
23  *
24  * @author  Gonzalo A. Aranda Corral
25  * @version 0.9.0
26  * @since   13/07/2008
27  *
28  */
29 public class Sintactico {
30
31         ALexico aLex;
32         String Actual;
33
34         /***
35          * Constructor de la clase
36          * @param m Mensaje a analizar
37          */
38         public Sintactico(String m) {
39
40                 // QUITAR LAS COMILLAS
41                 int i;
42                 while ((i = m.indexOf('\''))>−1){
43                         m = m.substring(0, i) + m.substring(i+1);
44                 }
45
46                 ALexico ALex = new ALexico(m);
47                 ALex.analizar();
48                 ALex.Reiniciar();
49
50                 this.aLex = ALex;
51         }
52
53         /***
54          * Llamada a la gramatica para comenzar el analisis
55          * @return True si la cadena TIENE FALLOS. False en caso contrario.
56          */
57         public boolean analizar() {
58                 boolean devolver = true;
59                 this.aLex.Reiniciar();
60                 Actual = aLex.SiguienteToken();
61
62                 devolver = Gramatica();
63                 if (devolver) {
64                         //Salida.Imprimirln("analisis sintactico lista OK");
65                 } else {
66                         System.out.println("******* ERROR SINTACTICO **********");
67                 }
68
69                 return !devolver;
70         }
71
72
73         /***
74          * Gramatica −−> Lista
75          */
76         private boolean Gramatica() {
77                 return Lista();
78         }
79
80         /***
81          * Lista −−> . ( Parametro , Lista )
82          *          | []
```

```
83              */
84          private boolean Lista() {
85                  if (Actual.equalsIgnoreCase(".")) {
86                          // Lista Completa
87                          return  consumir(".") &&
88                          consumir("(") &&
89                          Parametro() &&
90                          consumir(",") &&
91                          Lista() &&
92                          consumir(")");
93
94                  } else {
95                          // Lista Vacia
96                          return  consumir("[") && consumir("]");
97                  }
98          }
99
100
101         /***
102          * Parametro --> Nombre ( X1 , Y1 , X2 , Y2 ) => Insertar Region
103          *             | Nombre ( X1 , Y1 )           => Insertar Punto
104          */
105         private boolean Parametro() {
106                 Teoria miteoria = DummyPaella.getTeoria();
107
108                 boolean devolver = true;
109                 String Nombre;
110                 int X1,Y1,X2,Y2;
111
112                 Nombre = Actual;
113                 consumir();
114                 devolver &= consumir("(");
115                 X1 = Integer.parseInt(Actual) * 20;
116                 consumir();
117                 devolver &= consumir(",");
118                 Y1 = Integer.parseInt(Actual) * 20;
119                 consumir();
120                 if (Actual.equalsIgnoreCase(",")) {
121                         consumir(",");
122                         X2 = Integer.parseInt(Actual) * 20;
123                         consumir();
124                         devolver &= consumir(",");
125                         Y2 = Integer.parseInt(Actual) * 20;
126                         consumir();
127                         devolver &= consumir(")");
128                         /**************************
129                         Hay que irse al contexto.. y lo que se saque que sea
130                         un Argumento.
131
132                         ***************************/
133 //                      FCAAttribute a = Contexto.getAttribute(Nombre);
134
135                         String ii = Contexto.getElement(
136                                         Integer.parseInt(Nombre.substring(3))).getNombre();
137
138                         miteoria.insertarRegion(ii, X1,Y1,X2,Y2);
139
140                 } else {
141                         devolver &= consumir(")");
142 //                      FCAObject a = Contexto.getObject(Nombre);
143                         String ii = Contexto.getElement(
144                                         Integer.parseInt(Nombre.substring(3))).getNombre();
145                         miteoria.insertarPunto(ii, X1,Y1);
146
147                 }
148                 return devolver;
149         }
150
151         /***
152          * Comprueba que lo que se recibe del mensaje es lo que corresponde recibir
153          * @param  Palabra Cadena que es la que se debe de recibir.
154          * @return True si Palabra coincide con el contenido del mensaje a analizar.
155          */
156         private boolean consumir(String Palabra) {
157                 boolean devolver = true;
158
159                 if (Actual.equalsIgnoreCase(Palabra) ) {
160                         Actual = aLex.SiguienteToken();
161
162                 } else {
163                         devolver = false;
164                 }
```

```
165
166                     return devolver;
167             }
168
169             /****
170              * Consume un token
171              * @return True siempre.
172              */
173             private boolean consumir() {
174                     Actual = aLex.SiguienteToken();
175
176                     return true;
177             }
178
179     }
```

```
1  package es.us.ccia.dummy.restricciones.analizador;
2  /**
3   * Version 0.9:
4   *    20080713
5   *    − Version inicial
6   *
7   *================================================
8   * TODO:
9   *    − Se deberia de cambiar la forma de devolucion de
10  *      los metodos. Pero en principio esta bien.
11  */
12
13 import java.util.*;
14
15 /***
16  *
17  * @author  Gonzalo A. Aranda Corral
18  * @version 0.9.0
19  * @since   13/07/2008
20  *
21  */
22 public class ALexico {
23
24         String mensaje;
25         private int indice;
26         ArrayList tokens;
27
28         /***
29          * Constructor de la clase
30          * @param m Mensaje para convertir en tokens
31          */
32         public ALexico(String m) {
33                 this.mensaje = m;
34                 indice = 0;
35                 tokens = new ArrayList();
36         }
37
38         /***
39          * Inicializa el analizador
40          */
41         private void inicializar() {
42                 indice = 0;
43                 tokens.clear();
44         }
45
46         /***
47          * Reinicia el indice, para poder volver a recorrerlo
48          */
49         public void Reiniciar(){
50                 indice = 0;
51         }
52
53         /***
54          * Ejecuta el analisis lexico
55          * @return True si encuentra fallo. False en caso contrario.
56          */
57         public boolean analizar() {
58                 boolean devolver;
59
60                 this.inicializar();
61
62                 Automata miAutomata = new Automata(this.mensaje,tokens);
63                 devolver = miAutomata.analizar();
64
65                 return !devolver;
66         }
67
68         /***
69          * Devuelve el token por donde va y pasa al siguiente.
70          * @return Token actual
71          */
72         public String SiguienteToken() {
73                 if (indice<tokens.size()) {
74                         return this.tokens.get(indice++).toString();
75                 } else {
76                         return "";
77                 }
78
79         }
80
81         /***
82          * Comprueba si quedan tokens por analizar.
```

```
83              * @return True si quedan tokens, False en caso contrario.
84              */
85             public boolean HayToken() {
86                     return (indice < this.tokens.size());
87             }
88
89
90
91
92     }
```

```
1  package es.us.ccia.dummy.restricciones;
2  /**
3   * Version 0.9:
4   *    20080807
5   *    − AdaptaciÃ³n a RCC5
6   *
7   *    20080713
8   *    − Clase que llama al sistema CHR−PROLOG
9   *
10  *================================================
11  * TODO:
12  *    − Hacer la version incremental del calculo.
13  *    − Soportar completamente la respuesta de la lista vacia
14  *
15  */
16
17 import java.io.FileWriter;
18 import java.util.Iterator;
19
20 import jpl.Query;
21 import jpl.Term;
22 import jpl.Variable;
23
24 import es.us.ccia.dummy.contexto.Contexto;
25 import es.us.ccia.dummy.contexto.elementos.FCAAttribute;
26 import es.us.ccia.dummy.contexto.elementos.FCAElement;
27 import es.us.ccia.dummy.contexto.elementos.FCAObject;
28 import es.us.ccia.dummy.razonador.Razonador;
29 import es.us.ccia.dummy.restricciones.analizador.Sintactico;
30 import es.us.ccia.dummy.teoria.relaciones.Relacion;
31 import es.us.ccia.dummy.*;
32 import es.us.ccia.dummy.teoria.*;
33 import es.us.tad.Salida;
34
35 /***
36  *
37  * @author  Gonzalo A. Aranda Corral
38  * @version 0.9.0
39  * @since   13/07/2008
40  *
41  */
42 public class ResolverCHR {
43
44         /***
45          * Constructor de la clase
46          *
47          */
48         public ResolverCHR(){
49
50         }
51
52
53
54         /***
55          * Esquema general de resolucion del problema
56          * mediante restricciones.
57          */
58         public void resolv() {
59
60                 // 1. escribir fichero de la pregunta
61                 String nombre = this.crearFichero();
62
63                 // 2. Llamar a prolog para resolverlo
64                 String salida = this.preguntaProlog(nombre);
65
66                 // 3. Analizar la salida prolog
67                 this.analizarSalida(salida);
68
69                 // 4. Actualizar la teoria
70                 // La actualizacion se realiza dentro del analizador
71                 // sintactico.
72
73         }
74
75
76         /***
77          * Conexion con el sistema CHR−PROLOG y realizacion de la pregunta.
78          * @param fichero Fichero donde se encuentran todas las restricciones
79          *                de nuestro problema
80          * @return devuelve la respuesta de Prolog, en un objeto String pero con
81          *         formato de lista Prolog.
82          */
```

```
83          private String preguntaProlog(String fichero){
84                  String devolver = "";
85                  Query query = null;
86
87                  if (DummyPaella.Razonamiento == Razonador.RCC5) {
88                          query = new Query("consult([’rcc5.pl’,’" + fichero + "’])");
89                  } else if (DummyPaella.Razonamiento == Razonador.RCC8) {
90                          query = new Query("consult([’rcc8.pl’,’" + fichero + "’])");
91                  }
92
93                  if ( !query.hasSolution() ){
94                          Salida.Imprimirln( "El sistema CHR no puede cargar los ficheros" );
95
96                  } else {
97                          Variable X = new jpl.Variable("X");
98                          Query q1 = new Query("solucion", new Term[] {X});
99                          java.util.Hashtable[] solution;
100
101                         if (q1.hasSolution()) {
102                                 solution = q1.allSolutions();
103                                 devolver = solution[0].get("X").toString();
104                         } else
105                                 Salida.Imprimirln( "El sistema CHR no ha encontrado solucion" );
106                 }
107                 return devolver;
108
109         }
110
111
112         /***
113          * Construccion del fichero
114          * @return Devuelve una cadena con el nombre del fichero donde se
115          *         han grabado las restricciones.
116          *
117          */
118         private String crearFichero() {
119
120                 Teoria teoria = DummyPaella.getTeoria();
121
122                 String devolver = "pregunta.pl";
123                 StringBuffer buf = new StringBuffer();
124                 Iterator ele;
125                 boolean inicial;
126                 //=========================================================
127             // *  CABECERA DEL FICHERO
128             //=========================================================
129                 buf.append("% Generado automaticamente\n");
130                 buf.append("% % % % % % % % % % % % % % % % % % % % % % % % % %\n");
131                 buf.append("% % %      Preguntas del programa\n");
132                 buf.append("% % % % % % % % % % % % % % % % % % % % % % % % % %\n");
133                 buf.append(":− use_module(library(bounds)).\n");
134                 buf.append("\n");
135                 buf.append("\n");
136                 buf.append("solucion(Soluc) :−\n");
137
138             //=========================================================
139             // *   FORMATO DE LA SOLUCION
140             //=========================================================
141                 buf.append("\tSoluc = [");
142
143                 // CLASES
144                 ele = teoria.getListaClases().iterator();
145                 inicial = true;
146                 while (ele.hasNext()) {
147                         String nombre = ((FCAAttribute)ele.next()).getNombreVar();
148                         if (inicial) {
149                                 inicial = false;
150                         } else {
151                                 buf.append(",");
152                         }
153                         buf.append("’" + nombre + "’(X1" + nombre + ",Y1"+nombre + ",X2"+nombre
    + ",Y2"+ nombre + ")");
154                 }
155
156                 // Individuos
157                 ele = teoria.getListaIndividuos().iterator();
158                 if (teoria.numClases() > 0 && teoria.numIndividuos()>0)
159                         buf.append(",");
160                 inicial = true;
161                 while (ele.hasNext()) {
162                         String nombre = ((FCAObject)ele.next()).getNombreVar();
163                         if (inicial) {
```

```
164                                     inicial = false;
165                             } else {
166                                     buf.append(",");
167                             }
168                             buf.append("'" + nombre + "'(X1" + nombre + ",Y1"+nombre + ")");
169                     }
170                     buf.append("],\n");
171
172
173             //=========================================================
174             // *   LISTA DE VARIABLES
175             //=========================================================
176                     buf.append("\tVariables = [");
177                     ele = teoria.getListaClases().iterator();
178                     inicial = true;
179                     while (ele.hasNext()) {
180                             String nombre = ((FCAAttribute)ele.next()).getNombreVar();
181                             if (inicial) {
182                                     inicial = false;
183                             } else {
184                                     buf.append(",");
185                             }
186                             buf.append("X1" + nombre + ",Y1"+nombre + ",X2"+nombre + ",Y2"+ nombre)
    ;
187                     }
188
189                     ele = teoria.getListaIndividuos().iterator();
190                     if (teoria.numClases() > 0 && teoria.numIndividuos()>0)
191                             buf.append(",");
192                     inicial = true;
193                     while (ele.hasNext()) {
194                             String nombre = ((FCAObject)ele.next()).getNombreVar();
195                             if (inicial) {
196                                     inicial = false;
197                             } else {
198                                     buf.append(",");
199                             }
200                             buf.append("X1" + nombre + ",Y1"+nombre);
201                     }
202
203                     buf.append("],\n\n");
204
205
206
207             //=========================================================
208             // *   RANGO DE LAS VARIABLES
209             //=========================================================
210                     ele = teoria.getListaClases().iterator();
211                     inicial = true;
212                     while (ele.hasNext()) {
213                             String nombre = ((FCAAttribute)ele.next()).getNombreVar();
214                             buf.append("\t[X1" + nombre + ",Y1"+nombre + ",X2"+nombre + ",Y2"+ nombre
    + "] in 1..50,\n");
215                     }
216
217                     ele = teoria.getListaIndividuos().iterator();
218                     inicial = true;
219                     while (ele.hasNext()) {
220                             String nombre = ((FCAObject)ele.next()).getNombreVar();
221                             buf.append("\t[X1" + nombre + ",Y1"+nombre + "] in 1..50,\n");
222                     }
223                     buf.append("\n");
224
225
226             //=========================================================
227             // *   ANCHO DE LOS RECTANGULOS
228             //=========================================================
229                     ele = teoria.getListaClases().iterator();
230                     inicial = true;
231                     while (ele.hasNext()) {
232                             String nombre = ((FCAAttribute)ele.next()).getNombreVar();
233                             buf.append("\tX2" + nombre + " #> X1"+nombre + " + 3,\n" +
234                                             "\tY2"+nombre + " #> Y1"+ nombre + " + 3,\n");
235                     }
236                     buf.append("\n");
237
238             //=========================================================
239             // *   TODOS LOS PUNTOS SON DIFERENTES
240             //=========================================================
241
242                     // Podemos decir que las coordenadas Xs son diferentes.
243
```

```
244                     if (teoria.getListaIndividuos().size() > 1) {
245                             buf.append("\tall_different([");
246
247                             ele = teoria.getListaIndividuos().iterator();
248                             FCAObject o = (FCAObject)(ele.next());
249                             buf.append("X1" + o.getNombreVar());
250
251                             while (ele.hasNext()) {
252                                     FCAObject ob = (FCAObject)(ele.next());
253                                     buf.append(", X1" + ob.getNombreVar());
254                             }
255                             buf.append("]),\n\n");
256                     }
257
258     //              ele = teoria.getListaIndividuos().iterator();
259     //              if (ele.hasNext()) {
260     //                      String primero = ((FCAObject)ele.next()).getNombreVar();
261     //                      String segundo = "";
262     //                      while (ele.hasNext()) {
263     //                              segundo = (String)ele.next();
264     //                              buf.append("(Y1" + primero + "#> Y1" + segundo + " ; " +
265     //                                              "X1" + primero + "#> X1" + segundo + " ),");
266     //                      primero = segundo;
267     //                      }
268     //                      buf.append("\n");
269     //              }
270
271                 //=========================================================
272                 // *   LISTADO DE RELACIONES
273                 //=========================================================
274                     Iterator ite = teoria.getListaRelacionesRCC().iterator();
275                     inicial = true;
276                     while (ite.hasNext()) {
277                             Relacion temp1 = (Relacion)ite.next();
278                             String R = temp1.getNombreRelacion().toLowerCase();
279                             String As = temp1.getA();
280                             String Bs = temp1.getB();
281
282                             if (R.equalsIgnoreCase("in_p") || R.equalsIgnoreCase("out_p")) {
283
284                                     if (temp1.getA().getClass().getSimpleName().equalsIgnoreCase("
    FCAAttribute")) {
285                                             String A = Contexto.getAttribute(As).getNombreVar();
286                                             String B = Contexto.getObject(Bs).getNombreVar();
287                                             buf.append("\t" + R + "((" +
288                                                     "X1" +A + ",Y1" + A + ",X2" + A + ",Y2" + A + ")
    ,(" +
289                                                     "X1" +B + ",Y1" + B + ")" + "),\n");
290                                     } else {
291                                             String A = Contexto.getObject(As).getNombreVar();
292                                             String B = Contexto.getAttribute(Bs).getNombreVar();
293                                             buf.append("\t" + R + "((" +
294                                                             "X1" +B + ",Y1" + B + ",X2" + B + ",Y2"
     + B + "),(" +
295                                                             "X1" +A + ",Y1" + A + ")" + "),\n");
296                                     }
297
298                             } else if (R.equalsIgnoreCase("neq_p")) {
299
300                             } else if (R.equalsIgnoreCase("nodef")) {
301                                     System.out.println(temp1.getTipo());
302
303                             } else {
304
305                                     System.out.println(R);
306
307                                     String A = Contexto.getAttribute(As).getNombreVar();
308                                     String B = Contexto.getAttribute(Bs).getNombreVar();
309                                     buf.append("\t" + R + "((" +
310                                             "X1" +A + ",Y1" + A + ",X2" + A + ",Y2" + A + "),(" +
311                                             "X1" +B + ",Y1" + B + ",X2" + B + ",Y2" + B + ")" + "),\n
    ");
312                             }
313                     }
314                     buf.append("\n");
315
316                     buf.append("\tlabel(Variables),!.");
317
318                 //=========================================================
319                 // *   ESCRIBIR EL FICHERO
320                 //=========================================================
321                     try {
```

```
322                                     FileWriter t_FileWriter = new FileWriter(devolver);
323                                     t_FileWriter.write(buf.toString(), 0, buf.toString().length(
    ));
324                                     t_FileWriter.close();
325                     } catch (Exception e) {
326                             Salida.Imprimirln("Error de fichero: " + e.toString());
327                     }
328
329                     return devolver;
330             }
331
332             /***
333              * Invoca al analizador sintactico
334              * @param salida Cadena que devuelve el sistema de restricciones.
335              */
336             private void analizarSalida(String salida){
337                     Sintactico sint = new Sintactico(salida);
338                     sint.analizar();
339             }
340
341     }
```

```
1  package es.us.ccia.dummy.grafico;
2
3  import java.io.File;
4  import java.util.ArrayList;
5  import java.util.Iterator;
6  //import java.util.LinkedList;
7  import java.util.StringTokenizer;
8
9  import javax.swing.JFileChooser;
10 import javax.swing.event.ListSelectionEvent;
11
12 //import jpl.Query;
13
14 import es.us.ccia.dummy.DummyPaella;
15 import es.us.ccia.dummy.contexto.Contexto;
16 import es.us.ccia.dummy.contexto.elementos.FCAAttribute;
17 import es.us.ccia.dummy.contexto.elementos.FCAObject;
18 import es.us.ccia.dummy.contexto.entsal.Entrada;
19 import es.us.ccia.dummy.contexto.entsal.Grabar;
20 import es.us.ccia.dummy.contexto.entsal.Importar;
21 import es.us.ccia.dummy.dibujo.*;
22 //import es.us.ccia.dummy.historial.cambios.ListaCambios;
23 import es.us.ccia.dummy.razonGrafico.RazonG_RCC5;
24 import es.us.ccia.dummy.razonGrafico.RazonG_RCC8;
25 import es.us.ccia.dummy.razonador.Razonador;
26 import es.us.ccia.dummy.teoria.Teoria;
27 import es.us.ccia.dummy.teoria.regiones.ListaElementos;
28 import es.us.tad.*;
29
30 @SuppressWarnings({"serial","unchecked"})
31 public class EntornoG extends GUI {
32
33         public EntornoG() {
34                 super();
35         }
36
37         @Override
38         public void menuSalir() {
39
40                 // Abria que comprobar que no se ha modificado nada...
41                 // por si se quiere grabar..
42
43
44                 System.exit(0);
45         }
46
47         public void menuAbrir() {
48
49                 JFileChooser fileChooser = new JFileChooser();
50                 fileChooser.setCurrentDirectory(new File("."));
51
52                 int retVal = fileChooser.showOpenDialog(this);
53                 if (retVal == JFileChooser.APPROVE_OPTION) {
54                         String fileName = fileChooser.getSelectedFile().getAbsolutePath();
55                         try {
56                                 Ficheros f = new Ficheros(fileName);
57                                 String contenido = f.LeerFichero();
58                                 Entrada.analizarContenido(contenido);  // Esto lo carga en el
   contexto
59                                 Razonador.inicializar(Razonador.RCC5);
60
61                                 /**********
62                                  *
63                                  *********/
64                                 Salida.Imprimirln(Contexto.imprimir());
65                                 Salida.Imprimirln("----------------------");
66                                 Salida.Imprimirln(Razonador.escribirRel());
67                                 Salida.Imprimirln("----------------------");
68
69 //                                      DummyPaella.getTeoria().reiniciarContexto(c);
70                                 DummyPaella.getGUI().reiniciarContexto(
71                                                 Contexto.getAttributes(),Contexto.getObjects()
   );
72 //                                      DummyPaella.getGUI().getAreaDibujo().repaint();
73
74
75
76                         } catch (Exception ioe) {
77                                 Salida.Imprimirln("ERROR AL LEER EL FICHERO DE CONTEXTO");
78                                 ioe.printStackTrace();
79                         }
80                 }
```

```
81
82          }
83
84
85          public void menuDeshacer() {
86
87                  Teoria teoria = DummyPaella.getTeoria();
88
89                  teoria.undoListaRegiones();
90                  teoria.undoCambiosRelaciones();
91
92
93                  //Incluso este.. creo que va fuera
94                  repintarDibujo();
95  //                      getVentanaDibujo().repaint();
96
97                  pintarRelacionesRCC(null);
98                  pintarVentanaHistorial(null);
99
100         }
101
102         public void menuImportarDel() {
103
104                 Salida.Imprimirln("OPCION DE IMPORTAR");
105                 Salida.Imprimirln("−−Empieza a importar");
106
107                 Contexto.inicializar("importado");
108                 Salida.Imprimirln("−−contexto inicializado");
109
110                 Importar imp = new Importar("garanda","libertad32");
111 //                      Importar imp = new Importar("jaalonso","Wu_Wei_08");
112                 imp.importar();
113
114                 Salida.Imprimirln("−−FIN de la importacion");
115
116 //                      Razonador.inicializar(Razonador.RCC5);
117 //                      Salida.Imprimirln("−−razonador inicializado");
118
119
120                 /**********
121                  *
122                  *********/
123 //                      Salida.Imprimirln("----------------------");
124 //                      Salida.Imprimirln(Contexto.imprimir());
125 //                      Salida.Imprimirln("----------------------");
126 //                      Salida.Imprimirln(Razonador.escribirRel());
127 //                      Salida.Imprimirln("----------------------");
128
129 //                      DummyPaella.getTeoria().reiniciarContexto(c);
130                 DummyPaella.getGUI().reiniciarContexto(null,null);
131
132                 Salida.Imprimirln("−−grafico actualizado..");
133
134
135                         //                      DummyPaella.getGUI().getAreaDibujo().repaint()
    ;
136
137
138         }
139
140
141         public void menuGuardarComo() {
142
143                 Salida.Imprimirln("a guardar...");
144                 JFileChooser fileChooser = new JFileChooser();
145                 fileChooser.setCurrentDirectory(new File("."));
146
147                 int retVal = fileChooser.showSaveDialog(this);
148                 if (retVal == JFileChooser.APPROVE_OPTION) {
149                         String fileName = fileChooser.getSelectedFile().getAbsolutePath();
150                         Grabar.formatoCXT(fileName);
151                 }
152
153         }
154
155         @Override
156         public void menuExportarDel() {
157                 // TODO Auto−generated method stub
158
159         }
160
161         @Override
```

```
162         public void menuNuevo() {
163                 // TODO Auto-generated method stub
164                 System.out.println("opcion nuevaaa");
165         }
166 
167 
168         public void botonEjecutarCambios(ListaElementos _nuevaconfig) {
169 
170                 System.out.println("=============================================");
171                 System.out.println(DummyPaella.getTeoria().getListaRegiones());
172                 System.out.println(_nuevaconfig);
173                 System.out.println("=============================================");
174                 System.out.println(DummyPaella.getTeoria().getListaRelacionesRCC());
175 
176                 if (DummyPaella.Razonamiento == Razonador.RCC8) {
177                         System.out.println(RazonG_RCC8.Dibujo_a_RCC(_nuevaconfig));
178                 } else {
179                         System.out.println(RazonG_RCC5.Dibujo_a_RCC(_nuevaconfig));
180                 }
181                 System.out.println("=============================================");
182 
183 
184 
185 
186 
187 //              LinkedList<ListaCambios> miLista = DummyPaella.getTeoria().getLCambios();
188 //
189 //              Iterator i = miLista.iterator();
190 //              while(i.hasNext()) {
191 //                      ListaCambios cc = (ListaCambios)i.next();
192 //                      cc.ejecutar();
193 //                      Salida.Imprimirln(cc.toString());
194 //              }
195 
196         }
197 
198         @Override
199         public void cambioListaClases(ArrayList _lista) {
200 
201                 /**
202                  * QUE PASA CUANDO CAMBIO UNA SELECCION DE CLASE!!
203                  */
204 
205                 Teoria t = DummyPaella.getTeoria();
206                 t.limpiarClases();
207 
208                 ArrayList<FCAAttribute> listac = new ArrayList<FCAAttribute>();
209                 Iterator i = _lista.iterator();
210                 while (i.hasNext()) {
211                         FCAAttribute att  = Contexto.getAttribute((String)i.next());
212                         if (att == null) {
213                                 System.out.println("CLASE no encontrada");
214                         } else {
215                                 listac.add(att);
216                         }
217                 }
218                 t.insertarClases(listac);
219 
220                 t.Argumentos_a_RCC();
221 
222 
223                 t.calcularRegiones();
224 
225 //              System.out.println(t.getListaRelacionesRCC());
226 
227                 pintarRelacionesRCC(t.getListaRelacionesRCC().toList());
228 
229 //              System.out.println("====================================");
230 //              System.out.println(t.getListaRegiones().size());
231 //              System.out.println("====================================");
232 
233                 pintarVentanaDibujo(t.getListaRegiones());
234 
235 //              Salida.Imprimirln(t.getListaRelacionesRCC().toString());
236 //              Salida.Imprimirln(Contexto.imprimir());
237 
238         }
239 
240         @Override
241         public void cambioListaIndividuos(ArrayList _lista) {
242 
243                 Teoria t = DummyPaella.getTeoria();
```

```
244                     t.limpiarIndiv();
245
246                     ArrayList<FCAObject> listai = new ArrayList<FCAObject>();
247                     Iterator i = _lista.iterator();
248                     while (i.hasNext()) {
249                             FCAObject obj = Contexto.getObject((String)i.next());
250                             if (obj == null) {
251                                     Salida.Imprimirln("OBJETO no encontrado");
252                             } else {
253                                     listai.add(obj);
254                             }
255                     }
256                     t.insertarIndividuos(listai);
257
258 //                  System.out.println(t.toString());
259
260
261
262                     t.Argumentos_a_RCC();
263                     t.calcularRegiones();
264
265                     pintarRelacionesRCC(t.getListaRelacionesRCC().toList());
266
267                     pintarVentanaDibujo(t.getListaRegiones());
268
269
270
271
272 //
273 //                  /************************************************/
274 //                  /*******  CAmbio de LISTA DE INDIVIDUOS   *********/
275 //                  /************************************************/
276 //                  if (!DummyPaella.getGUI().getVentanaDibujo().estaModificada()) {
277 //
278 //                          actualizarIndividuos();
279 //
280 //                  } else {
281 //
282 //                          int res = JOptionPane.showConfirmDialog( DummyPaella.getGUI().getVentanaDibujo(),
283 //                                                  "If you change your selection,you will loose your movements,\n are you SURE?",
284 //                                                  "Confirmation − Paella.",JOptionPane.YES_NO_OPTION );
285 //
286 //                          if( res == JOptionPane.YES_OPTION ) {
287 //                                  actualizarIndividuos();
288 //                                  DummyPaella.getGUI().getVentanaDibujo().setModificado(false);
289 //                          } else {
290 //                                  //respuesta = "Si";
291 //                          }
292 //
293 //                          Salida.Imprimirln("Esto no se puede elegir.. el dibujo esta modificado.");
294 //                  }
295         }
296
297
298         public void cambioListaRelaciones() {
299
300                 /************************************************/
301                 /*******  CAmbio de LISTA DE RELACIONNES   *********/
302                 /************************************************/
303
304                 ListaElementos miteoria = DummyPaella.getTeoria().getListaRegiones();
305
306 //              if (!(arg0.getValueIsAdjusting())) {
307
308                         miteoria.desmarcarElementos();
309                         ArrayList lista = getListaRelaciones();
310                         Iterator i = lista.iterator();
311
312                         while (i.hasNext()) {
313                                 StringTokenizer r = new StringTokenizer(i.next().toString(),"(),");
314                                 r.nextToken();
315                                 miteoria.marcarElemento(r.nextToken());
316                                 miteoria.marcarElemento(r.nextToken());
317                         }
318                         repintarDibujo();
319 //              }
320
321         }
```

```
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337 //      private void actualizarClases(ArrayList _lista) {
338 //
339 //              // TEORIA
340 //              Teoria miteoria = DummyPaella.getTeoria();
341 //
342 //              // INICIALIZACION
343 //              miteoria.limpiarClases();
344 //              miteoria.insertarClases(_lista);
345 //
346 //              // LLAMADA AL RAZONADOR
347 //              miteoria.Argumentos_a_RCC();
348 //
349 //              // MINIMIZAR RELACIONES
350 ////            miteoria.simplificarRelacionesRCC8();
351 //
352 //              // CHR − PROLOG
353 //              miteoria.calcularRegiones();
354 //
355 //              // DESDE EL DIBUJO
356 //              miteoria.Dibujo_a_RCC8();
357 ////            miteoria.simplificarRelacionesRCC8();
358 //
359 //              // ACTUALIZAR GRAFICOS
360 //              getVentanaRelaciones().actualizar(null);
361 //              getVentanaHistorial().limpiar();
362 //              getAreaDibujo().repaint();
363 //
364 //      }
365 //
366 //      private void actualizarIndividuos(ArrayList _lista) {
367 //              // TEORIA
368 //              Teoria miteoria = DummyPaella.getTeoria();
369 //
370 //              // INICIALIZACION
371 //              miteoria.limpiarIndiv();
372 //              miteoria.insertarIndividuos(_lista);
373 //
374 //              // LLAMADA AL RAZONADOR
375 //              miteoria.Argumentos_a_RCC();
376 //
377 //              // CHR − PROLOG
378 //              miteoria.calcularRegiones();
379 //
380 //              // DESDE EL DIBUJO
381 //              miteoria.Dibujo_a_RCC8();
382 ////            miteoria.simplificarRelacionesRCC8();
383 //
384 //              // ACTUALIZAR GRAFICOS
385 //              getAreaDibujo().repaint();
386 //              getVentanaRelaciones().actualizar(null);
387 //              getVentanaHistorial().limpiar();
388 //
389 //      }
390
391         @Override
392         public void cambioVentanaDibujo(ListaElementos _lista) {
393                 // TODO Auto−generated method stub
394 //              System.out.println(_lista.toString());
395
396
397
398 //
399 //              teoria.Dibujo_a_RCC8();
400 //
401 //              ListaCambios lc = teoria.getCambiosRelaciones(activo.getArgumento());
402 //              lc.ejecutar();
403 //
```

```
404 ////            RazonadorRCC8.reCalcula();
405 //
406 //              teoria.Argumentos_a_RCC();
407 //              teoria.calcularRegiones();
408 //              teoria.Dibujo_a_RCC8();
409 //
410 //              Salida.Imprimirln(Contexto.imprimir());
411 //              Salida.Imprimirln("----------------------");
412 //              Salida.Imprimirln(Razonador.escribirRel());
413 //              Salida.Imprimirln("----------------------");
414 //
415 //
416 //
417 //              g.getVentanaRelaciones().actualizar();
418 //              g.getVentanaHistorial().actualizar();
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440         }
441
442         public void valueChanged(ListSelectionEvent e) {
443                 // TODO Auto-generated method stub
444
445         }
446
447
448
449 }
```

```
1  package es.us.ccia.dummy;
2
3  import java.awt.*; //para uar fuente arial 10.
4
5  import es.us.ccia.dummy.contexto.Contexto;
6  import es.us.ccia.dummy.dibujo.GUI;
7  import es.us.ccia.dummy.grafico.EntornoG;
8  import es.us.ccia.dummy.razonador.*;
9  import es.us.ccia.dummy.teoria.*;
10 import es.us.tad.Salida;
11
12
13 public class DummyPaella
14 {
15
16
17         private static EntornoG gUI;
18         private static Teoria teoria;
19         public static int Razonamiento;
20
21         public DummyPaella() {
22                 super();
23                 Salida.inicializar(true);
24                 teoria   = new Teoria();
25                 Contexto.inicializar("pordefecto");
26                 Razonamiento = Razonador.RCC5;
27                 Razonador.inicializar(Razonamiento);
28                 gUI      = new EntornoG();
29                 gUI.init();
30         }
31
32         public void inicializar() {
33                 Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
34                 gUI.setSize(screenSize.width,screenSize.height);
35                 gUI.setBackground( Color.lightGray );
36                 gUI.setVisible(true);
37
38
39
40         }
41
42         public static Teoria getTeoria() {
43                 return teoria;
44         }
45
46         public static GUI getGUI() {
47                 if (gUI==null)
48                         Salida.Imprimirln("GUI NULO");
49                 return gUI;
50         }
51
52
53         static public void main( String args[] ) {
54                 DummyPaella d = new DummyPaella();
55                 d.inicializar();
56         }
57
58 }
59
```

```
1  package es.us.ccia.dummy.razonador;
2
3  import es.us.ccia.dummy.contexto.elementos.FCAAttribute;
4  import es.us.ccia.dummy.contexto.elementos.FCAElement;
5  import es.us.ccia.dummy.contexto.elementos.FCAObject;
6
7  public interface A_Razon {
8
9  //      public abstract boolean inicializar();
10
11         public abstract int getRelacionRCC(FCAElement c1, FCAElement c2);
12
13         public abstract boolean equivalentes(FCAAttribute c1, FCAAttribute c2);
14         public abstract boolean subsume     (FCAAttribute c1, FCAAttribute c2);
15         public abstract boolean disjuntas   (FCAAttribute c1, FCAAttribute c2);
16         public abstract boolean perteneceAClase(FCAObject p,  FCAAttribute  c);
17
18         public abstract String escribirRelaciones();
19
20 }
```

```
1  package es.us.ccia.dummy.razonador;
2
3  import java.util.*;
4
5  import es.us.ccia.dummy.contexto.*;
6  import es.us.ccia.dummy.contexto.elementos.FCAAttribute;
7  import es.us.ccia.dummy.contexto.elementos.FCAElement;
8  import es.us.ccia.dummy.contexto.elementos.FCAObject;
9  import es.us.ccia.dummy.teoria.*;
10 import es.us.ccia.rcc.*;
11 import es.us.tad.Salida;
12
13 /**
14  * Clase que implementa el razonador sobre el contexto.
15  * Puede darse el caso que en otra aplicacion el razonador
16  * sea externo, por lo que la intencion es cambiar unicamente
17  * esta clase.
18  * @author garanda
19  *
20  */
21 @SuppressWarnings("unchecked")
22 public class RazonadorxRCC5 implements A_Razon {
23
24
25         /**
26          * Constructor del razonador RRC5.
27          * No requiere nada, ya que en RCC5 se puede hacer todo en base sÃ³lo a los elementos
28          * y no al resto de valores.
29          *
30          */
31         protected RazonadorxRCC5() {    }
32
33
34         public int getRelacionRCC(FCAElement _a, FCAElement _b) {
35
36                 if ((_a instanceof FCAAttribute) &&(_b instanceof FCAAttribute) ) {
37                         return getRelacionClases((FCAAttribute)_a,(FCAAttribute)_b);
38                 } else if ((_a instanceof FCAAttribute) && (_b instanceof FCAObject) ) {
39                         return getRelacionClaseInd((FCAAttribute)_a,(FCAObject)_b);
40                 } else if ((_a instanceof FCAObject) && (_b instanceof FCAAttribute) ) {
41                         return getRelacionClaseInd((FCAAttribute)_b,(FCAObject)_a);
42                 } else
43                         return RCC.DIFER5;
44
45         }
46
47
48
49
50
51
52
53
54
55
56
57
58
59         public int getRelacionClases(FCAAttribute _a, FCAAttribute _b) {
60
61                 if (equivalentes(_a, _b)) {
62                         return RCC.EQ5;
63
64                 } else if (disjuntas(_a, _b)) {
65                         return RCC.DR;
66
67                 } else if (subsume(_a, _b)) {
68                         return RCC.PPi;
69
70                 } else if (subsume(_b, _a)) {
71                         return RCC.PP;
72
73                 } else {
74                         return RCC.PO5;
75                 }
76
77         }
78
79
80         public int getRelacionClaseInd(FCAAttribute _a, FCAObject _b) {
81
82                 if (perteneceAClase(_b,_a))
```

```
83                              return RCC.IN_P5;
84                      else
85                              return RCC.OUT_P5;
86
87              }
88
89
90
91              public boolean disjuntas(FCAAttribute c1, FCAAttribute c2) {
92                      // Dos clases son disjuntas cuando la interseccion es vacia.
93                      LinkedList l1 = c1.getRelaciones();
94                      LinkedList l2 = c2.getRelaciones();
95                      l1.retainAll(l2);
96
97                      return l1.size() == 0;
98              }
99
100             public boolean subsume(FCAAttribute c1, FCAAttribute c2) {
101                     // Una clase contiene a otra, cuando tiene todos sus elementos
102                     LinkedList l1 = (LinkedList) c1.getRelaciones();
103                     LinkedList l2 = (LinkedList) c2.getRelaciones();
104                     l1.retainAll(l2);
105
106                     return l1.containsAll(l2);
107             }
108
109             public boolean equivalentes(FCAAttribute c1, FCAAttribute c2) {
110                     // Dos conjuntos, a y b, son iguales cuando a subsume b y b subsume a a
111                     return subsume(c1,c2) && subsume(c2,c1);
112             }
113
114
115             public boolean perteneceAClase(FCAObject p, FCAAttribute c) {
116                     // Un punto pertenece a una clase, si estÃ¡ dentro de la lista de enlaces.
117                     return c.getRelaciones().contains(p);
118
119             }
120
121             /**
122              * Devuelve una cadena para el PrettyPrinting de las relaciones.
123              * @return
124              */
125             public String escribirRelaciones() {
126                     StringBuffer devolver = new StringBuffer();
127
128 //                  for (int i=0;i<NEtiquetas;i++) {
129 //                          devolver.append("\t" + Contexto.getArgumentoC(i).toString());
130 //                  }
131 //                  devolver.append('\n');
132 //
133 //                  for (int i=0;i<NEtiquetas;i++) {
134 //                          devolver.append(Contexto.getArgumentoC(i).toString());
135 //
136 //                          for (int j=0;j<NEtiquetas;j++) {
137 //                                  if (i==j) {
138 //                                          devolver.append("\t−");
139 //                                  } else {
140 //                                          devolver.append("\t" +
141 //                                                  RCC8.toString(getRelacionRCC8(i,j)));
142 //                                  }
143 //                          }
144 //                          devolver.append('\n');
145 //
146 //                  }
147 //
148
149
150
151                     return devolver.toString();
152 //                          return "No se pueden escribir las relaciones.";
153             }
154
155 }
```

```
1  package es.us.ccia.dummy.razonador;
2
3
4  import java.util.*;
5
6  import es.us.ccia.dummy.contexto.elementos.FCAAttribute;
7  import es.us.ccia.dummy.contexto.elementos.FCAElement;
8  import es.us.ccia.dummy.contexto.elementos.FCAObject;
9  import es.us.ccia.rcc.RCC;
10
11
12 /**
13  * Clase que implementa el razonador sobre el contexto.
14  * Puede darse el caso que en otra aplicacion el razonador
15  * sea externo, por lo que la intencion es cambiar unicamente
16  * esta clase.
17  * @author garanda
18  *
19  */
20 @SuppressWarnings("unchecked")
21 public class RazonadorRCC8 implements A_Razon {
22
23         /**
24          * Estructura principal de datos
25          */
26         private int[][] Valores;
27         private int[][] Relaciones;
28
29         /**
30          * Limites auxiliares
31          */
32         private int NEtiquetas, NObjetos;
33
34
35
36 //-----------------------------------------------------------------
37
38         /**
39          * Constructor del razonador con un conjunto inicial de valores.
40          * Es privado para que nadie pueda instanciarlo directamente.
41          * @param _valores Matriz de valores del contexto.
42          */
43         public RazonadorRCC8() {
44
45 //              Valores    = _valores;
46 //              Relaciones = null;
47 //
48 //              if (Valores==null || Valores.length == 0) {
49 //                      NEtiquetas = 0;
50 //                      NObjetos   = 0;
51 //              } else {
52 //                      NEtiquetas = Valores.length;
53 //                      NObjetos   = Valores[0].length;
54 //              }
55 //              Relaciones = new int[NEtiquetas][NEtiquetas];
56 //
57 //
58 //
59 //              System.out.println("Dimensiones de valores " + NEtiquetas + " x " + NObjetos);
60 //
61 //
62 //
63 //
64 //
65
66         }
67
68 //      public static int getRelacionRCC8(int _et, int _ob) {
69 //              return razon.Relaciones[_et][_ob];
70 //      }
71 //
72 //
73 //      public static int pertenecePuntoClase(int _punto, int _clase) {
74 //              if (razon.Valores[_clase][_punto] == 1)
75 //                      return RCC8.IN_P;
76 //              else
77 //                      return RCC8.OUT_P;
78 //      }
79
80
81
82         private int encontrarRelacionRCC5(int _a, int _b) {
```

```
83
84                  int devolver  = RCC.EQ;
85                  boolean comun = false;
86
87                  int obj = 0;
88
89
90                  while (obj<NObjetos && devolver != RCC.PO) {
91
92                          int A  = Valores[_a][obj];
93                          int B  = Valores[_b][obj];
94
95                          switch (devolver) {
96
97                          case RCC.EQ:
98
99                                  if(A==1 && B == 1) {
100                                         comun = true;
101
102                                 } else if (A==0 && B == 1) {
103                                         devolver = RCC.NTPP;
104
105                                 } else if (A==1 && B == 0) {
106                                         devolver = RCC.NTPPi;
107                                 }
108                                 break;
109
110                         case RCC.NTPP:
111
112                                 if(A==1) {
113                                         if( B == 1) {
114                                                 comun = true;
115
116                                         } else {
117                                                 if (comun) {
118                                                         devolver = RCC.PO;
119                                                 } else {
120                                                         devolver = RCC.DC;
121                                                 }
122                                         }
123                                 }
124                                 break;
125
126                         case RCC.NTPPi:
127
128                                 if(A==0 && B == 1) {
129                                         if (comun) {
130                                                 devolver = RCC.PO;
131                                         } else {
132                                                 devolver = RCC.DC;
133                                         }
134                                 } else if (A==1 && B==1){
135                                         comun = true;
136                                 }
137                                 break;
138
139                         case RCC.DC:
140
141                                 if(A==1 && B == 1) {
142                                         devolver = RCC.PO;
143                                 }
144                                 break;
145
146                         }
147                         obj++;
148                 }
149
150                 return(devolver);
151         }
152
153
154
155
156 //      private void procesaContexto() {
157 //
158 //              for(int i=0;i<NEtiquetas;i++) {
159 //                      for(int j=i+1;j<NEtiquetas;j++) {
160 //                              int r = encontrarRelacionRCC5(i, j);
161 //                              Relaciones[i][j] = r;
162 //                              Relaciones[j][i] = RCC8.reflexiva(r);
163 //                      }
164 //              }
```

```
165 //
166 //              convertiraRCC8();
167 //
168 //      }
169 //
170 //      public static void reCalcula() {
171 //              razon.procesaContexto();
172 //      }
173 //
174 //      public static void reCalcula(ArgumentoC _a) {
175 //              razon.procesaContexto(_a);
176 //      }
177 //
178 //      private void procesaContexto(FCAAttribute _a) {
179 //
180 //              for(int i=0;i<NEtiquetas;i++) {
181 ////                      int r = encontrarRelacionRCC5(i, _a.getNumero());
182 ////                      Relaciones[i][_a.getNumero()] = r;
183 ////                      Relaciones[_a.getNumero()][i] = RCC8.reflexiva(r);

184 //              }
185 //
186 //              convertiraRCC8();
187 //
188 //      }
189
190
191
192
193 ////////////////////////////////////////////////////////////////////////
194 ////////////////////////////////////////////////////////////////////////
195 ////////////////////////////////////////////////////////////////////////
196 ////////////////////////////////////////////////////////////////////////
197 ////////////////////////////////////////////////////////////////////////
198 ////////////////////////////////////////////////////////////////////////
199
200
201
202
203
204 //      private void convertiraRCC8(){
205 //
206 //              convertir_PO_EC();
207 //              convertir_NTPP_TPP();
208 //
209 //      }
210 //
211 //
212
213
214
215
216
217
218
219         private void convertir_PO_EC(){
220
221                 for(int i=0;i<NEtiquetas;i++) {
222                         for(int j=i+1;j<NEtiquetas;j++) {
223                                 if (Relaciones[i][j]==RCC.PO){
224                                         ArrayList la = getSubClases(i);
225                                         ArrayList lb = getSubClases(j);
226                                         lb.trimToSize();
227                                         la.retainAll(lb);
228                                         if (la.isEmpty()) {
229                                                 Relaciones[i][j] = RCC.EC;
230                                                 Relaciones[j][i] = RCC.EC;
231                                         }
232
233                                 }
234
235                         }
236                 }
237
238         }
239
240         private void convertir_NTPP_TPP(){
241
242                 for(int i=0;i<NEtiquetas;i++) {
243                         for(int j=0;j<NEtiquetas;j++) {
244                                 if (Relaciones[i][j]==RCC.NTPP){
245                                         ArrayList la = get_EC(i);
```

```
246                                         ArrayList lb = get_EC(j);
247                                         la.retainAll(lb);
248                                         if (!la.isEmpty()) {
249                                                 Relaciones[i][j] = RCC.TPP;
250                                                 Relaciones[j][i] = RCC.TPPi;
251                                         }
252
253                                 }
254
255                         }
256                 }
257
258
259
260
261
262
263
264
265
266 //          Ademas hay que ampliar rcc5 a rcc8
267
268         }
269
270
271
272 ////////////////////////////////////////////////////////////////////////
273
274         private ArrayList getSubClases(int _a) {
275                 ArrayList devolver = new ArrayList();
276 //
277 //              for (int i=0;i<NEtiquetas;i++) {
278 //
279 //                      if (Relaciones[i][_a] == RCC8.NTPP) {
280 //                              ArgumentoC ci = Contexto.getArgumentoC(i);
281 //                              devolver.add(ci);
282 //                      }
283 //              }
284 //
285                 return devolver;
286         }
287
288 //      public static ArrayList getEC(ArgumentoC _a) {
289 //              return razon.get_EC(_a.getNumero());
290 //      }
291 //
292
293         private  ArrayList get_EC(int _a) {
294                 ArrayList devolver = new ArrayList();
295
296 //              for (int i=0;i<NEtiquetas;i++) {
297 //
298 //                      if (Relaciones[i][_a] == RCC8.EC) {
299 //                              ArgumentoC ci = Contexto.getArgumentoC(i);
300 //                              devolver.add(ci);
301 //                      }
302 //              }
303
304                 return devolver;
305         }
306
307
308         public ArrayList get_PO(FCAAttribute _a) {
309                 ArrayList devolver = new ArrayList();
310
311 //              for (int i=0;i<razon.NEtiquetas;i++) {
312 //
313 //                      if (Relaciones[i][_a.getNumero()] == RCC8.PO) {
314 //                              ArgumentoC ci = Contexto.getArgumentoC(i);
315 //                              devolver.add(ci);
316 //                      }
317 //              }
318
319                 return devolver;
320         }
321
322
323
324
325
326
327
```

```
328
329
330
331
332
333
334
335
336 ////////////////////////////////////////////////////////////////////////
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356 ////////////////////////////////////////////////////////////////////////
357 ////////////////////////////////////////////////////////////////////////
358 ////////////////////////////////////////////////////////////////////////
359 ////////////////////////////////////////////////////////////////////////
360 ////////////////////////////////////////////////////////////////////////
361 ////////////////////////////////////////////////////////////////////////
362
363 //      public static ListaArgum get_ObjetosClase(ArgumentoC nombre) {
364 //              return razon.getObjetosClase(nombre);
365 //      }
366
367 //      private LinkedList getObjetosClase(FCAAttribute nombre) {
368 //              return nombre.getRelaciones();
369
370 //              if (nombre != null) {
371 //                      for(int j=0; j<NObjetos; j++) {
372 //                              if (Valores[nombre.getNumero()][j]==1) {
373 //                                      lista.add(Contexto.getArgumentoI(j));

374 //                              }
375 //                      }
376 //              }
377 //
378 //              return lista;
379 //      }
380
381
382         /**
383          * Realiza la UNION(de conjuntos) de los individuos que pertenecen a
384          * una lista de clases.
385          * @param _lista Lista de clases para unir
386          * @return la lista de todos los individuos que pertenecen, al menos
387          * una vez, a alguna de las clases
388          */
389         public static Collection unionObjPertClases(Collection _lista) {
390                 Collection resultado = new LinkedList();
391
392
393 //              Iterator<ArgumentoC> it = _lista.iterator();
394 //              while (it.hasNext()) {
395 //                      ArrayList<ArgumentoI> temp = razon.getObjetosClase(it.next()).toArrayL
    ist();
396 //                      Iterator<ArgumentoI> it2 = temp.iterator();
397 //                      while (it2.hasNext()) {
398 //                              ArgumentoI t2 = it2.next();
399 //                              if (!resultado.contains(t2)) {
400 //                                      resultado.add(t2);
401 //                              }
402 //                      }
403 //              }
404
405                 return resultado;
406         }
407
```

```
408
409
410             /***************
411              * FALTARIA LA INTERSECCIONNN
412              *
413              */
414
415
416             /**
417              *
418              * @return
419              */
420     //      public static String escribirRel() {
421     //              return razon.escribirRelaciones();
422     //      }
423
424             /**
425              *
426              * @return
427              */
428             public String escribirRelaciones() {
429                     StringBuffer devolver = new StringBuffer();
430
431     //              for (int i=0;i<NEtiquetas;i++) {
432     //                      devolver.append("\t" + Contexto.getArgumentoC(i).toString());
433     //              }
434     //              devolver.append('\n');
435     //
436     //              for (int i=0;i<NEtiquetas;i++) {
437     //                      devolver.append(Contexto.getArgumentoC(i).toString());
438     //
439     //                      for (int j=0;j<NEtiquetas;j++) {
440     //                              if (i==j) {
441     //                                      devolver.append("\t-");
442     //                              } else {
443     //                                      devolver.append("\t" +
444     //                                                      RCC8.toString(getRelacionRCC8(i,j)));
445     //                              }
446     //                      }
447     //                      devolver.append('\n');
448     //
449     //              }
450
451
452
453
454                     return devolver.toString();
455     //              return "No se pueden escribir las relaciones.";
456             }
457
458
459     //      public static void set_ObjetoClase(ArgumentoI _ind, ArgumentoC _etiq) {
460     //              razon.Valores[_etiq.getNumero()][_ind.getNumero()] = 1;
461     //      }
462     //
463     //      public static void remove_ObjetoClase(ArgumentoI _ind, ArgumentoC _etiq) {
464     //              razon.Valores[_etiq.getNumero()][_ind.getNumero()] = 0;
465     //      }
466     //
467     //
468
469
470
471
472
473             public boolean disjuntas(FCAAttribute c1, FCAAttribute c2) {
474                     // TODO Auto-generated method stub
475                     return false;
476             }
477
478
479
480
481
482
483
484             public boolean equivalentes(FCAAttribute c1, FCAAttribute c2) {
485                     // TODO Auto-generated method stub
486                     return false;
487             }
488
489
```

```
490
491
492
493
494
495         public int getRelacionRCC(FCAElement c1, FCAElement c2) {
496                 // TODO Auto-generated method stub
497                 return 0;
498         }
499
500
501
502
503
504         public boolean puntoDentro(FCAObject c1, FCAAttribute c2) {
505                 // TODO Auto-generated method stub
506                 return false;
507         }
508
509
510
511
512
513
514
515         public boolean subsume(FCAAttribute c1, FCAAttribute c2) {
516                 // TODO Auto-generated method stub
517                 return false;
518         }
519
520
521
522
523
524
525
526         public boolean perteneceAClase(FCAObject p, FCAAttribute c) {
527                 // TODO Auto-generated method stub
528                 return false;
529         }
530
531
532
533
534
535
536
537
538
539 }
540
541
542
543 ///*************************************************
544 // *************************************************
545 // *
546 // *   FUNCIONES PRIMITIVAS
547 // *
548 // *************************************************
549 // *************************************************/
550 //      public static boolean equivalentes(OWLNamedClass c1,OWLNamedClass c2) {
551 //              boolean devolver = false;
552 //              try {
553 //                      Collection coleccion = Paella.getRazonador().getEquivalentClasses(c1,
    null);
554 //                      if (coleccion.contains(c2)) {
555 //                              devolver = true;
556 //                      }
557 //              } catch (Exception ee) {
558 //                              devolver = false;
559 //                      }
560 //
561 //              return devolver;
562 //      }
563 //
564 //      public static boolean subsume(OWLNamedClass c1,OWLNamedClass c2) {
565 //              // c1 subsume a c2
566 //              boolean devolver = false;
567 //              try {
568 //                      devolver = Paella.getRazonador().isSubsumedBy(c2, c1, null);
569 //                      } catch (Exception ee) {
570 //                              devolver = false;
```

```
571 //                      }
572 //              return devolver;
573 //      }
574 //
575 //      public static boolean disjuntas(OWLNamedClass c1,OWLNamedClass c2) {
576 //              boolean devolver = false;
577 //              try {
578 //              devolver = Paella.getRazonador().isDisjointTo(c1, c2, null);
579 //              } catch (Exception ee) {
580 //                      devolver = false;
581 //              }
582 //              return devolver;
583 //      }
584 //
585 //      public static boolean puntoDentro(OWLNamedClass c1, OWLIndividual p1) {
586 //              boolean devolver = false;
587 //              try {
588 //                      Collection cc = Paella.getRazonador().getIndividualsBelongingToClass(c
    1, null);
589 //                      devolver = cc.contains(p1);
590 //              } catch (Exception ee) {
591 //                      devolver = false;
592 //              }
593 //              return devolver;
594 //      }
```

```
1  package es.us.ccia.dummy.razonador;
2
3  import es.us.ccia.dummy.contexto.elementos.FCAAttribute;
4  import es.us.ccia.dummy.contexto.elementos.FCAElement;
5  import es.us.ccia.dummy.contexto.elementos.FCAObject;
6
7  public class Razonador {
8
9          public static final int Ninguno = 0;
10         public static final int RCC5    = 1;
11         public static final int RCC8    = 2;
12
13
14
15         static A_Razon razon;
16         static int tipoRazon;
17
18
19         public static boolean inicializar(int _tipoRazon){
20                 tipoRazon = _tipoRazon;
21
22                 switch (tipoRazon) {
23                 case RCC5:
24                         razon = new RazonadorxRCC5();
25         //              razon.procesaContexto();
26                         break;
27
28                 case RCC8:
29                         razon = new RazonadorRCC8();
30         //              razon.procesaContexto();
31                         break;
32                 }
33                 return true;
34         }
35
36
37         public static int getRelacionRCC(FCAElement c1, FCAElement c2) {
38                 return razon.getRelacionRCC(c1, c2);
39         }
40
41
42
43         public static boolean disjuntas(FCAAttribute c1, FCAAttribute c2) {
44                 return razon.disjuntas(c1, c2);
45         }
46
47
48
49         public static boolean equivalentes(FCAAttribute c1, FCAAttribute c2) {
50                 return razon.equivalentes(c1, c2);
51         }
52
53
54         public static boolean subsume(FCAAttribute c1, FCAAttribute c2) {
55                 return razon.subsume(c1, c2);
56         }
57
58
59         public static boolean perteneceAClase(FCAObject p, FCAAttribute c) {
60                 return razon.perteneceAClase(p,c);
61         }
62
63
64         /**
65           *
66           * @return
67           */
68         public static String escribirRel() {
69                 return razon.escribirRelaciones();
70         }
71
72
73
74
75
76
77
78 }
```

```
1  package es.us.ccia.dummy.historial;
2
3  import es.us.ccia.dummy.contexto.elementos.FCAAttribute;
4  import es.us.ccia.dummy.contexto.elementos.FCAElement;
5  import es.us.ccia.dummy.historial.cambioRCC5.Cambio_Null;
6  import es.us.ccia.dummy.historial.cambioRCC5.dr.*;
7  import es.us.ccia.dummy.historial.cambioRCC5.eq5.*;
8  import es.us.ccia.dummy.historial.cambioRCC5.outin.IN5_OUT5;
9  import es.us.ccia.dummy.historial.cambioRCC5.outin.OUT5_IN5;
10 import es.us.ccia.dummy.historial.cambioRCC5.po5.*;
11 import es.us.ccia.dummy.historial.cambioRCC5.pp.*;
12 import es.us.ccia.dummy.historial.cambioRCC5.ppi.*;
13 import es.us.ccia.rcc.RCC;
14
15
16 public class H_Cambios5 {
17
18
19         public static Cambio getCambio(FCAElement _A, FCAElement _B,
20                         int _anterior, int _nuevo, FCAElement _act) {
21
22                 Cambio devolver = null;
23
24                 switch (_anterior) {
25
26                 case RCC.DR:
27                         switch (_nuevo) {
28
29                         case RCC.PO5:
30                                 devolver = new DR_PO5((FCAAttribute)_A,(FCAAttribute)_B,(FCAAt
    tribute)_act);
31                                 break;
32
33                         case RCC.PP:
34                                 devolver = new DR_PP((FCAAttribute)_A,(FCAAttribute)_B,(FCAAtt
    ribute)_act);
35                                 break;
36
37                         case RCC.PPi:
38                                 devolver = new DR_PPi((FCAAttribute)_A,(FCAAttribute)_B,(FCAAt
    tribute)_act);
39                                 break;
40
41                         case RCC.EQ5:
42                                 devolver = new DR_EQ5((FCAAttribute)_A,(FCAAttribute)_B,(FCAAt
    tribute)_act);
43                                 break;
44
45
46                         default:
47                                 devolver = new Cambio_Null(_A,_B,_act);
48
49                         }
50
51                         break;
52
53
54
55                 case RCC.EQ5:
56                         switch (_nuevo) {
57                         case RCC.DR:
58                                 devolver = new EQ5_DR((FCAAttribute)_A,(FCAAttribute)_B,(FCAAt
    tribute)_act);
59                                 break;
60
61                         case RCC.PO5:
62                                 devolver = new EQ5_PO5((FCAAttribute)_A,(FCAAttribute)_B,(FCAA
    ttribute)_act);
63                                 break;
64
65                         case RCC.PP:
66                                 devolver = new EQ5_PP((FCAAttribute)_A,(FCAAttribute)_B,(FCAAt
    tribute)_act);
67                                 break;
68
69                         case RCC.PPi:
70                                 devolver = new EQ5_PPi((FCAAttribute)_A,(FCAAttribute)_B,(FCAA
    ttribute)_act);
71                                 break;
72
73                         default:
74                                 devolver = new Cambio_Null(_A,_B,_act);
```

```
75
76                          }
77
78                          break;
79
80
81                  case RCC.PO5:
82                          switch (_nuevo) {
83                          case RCC.DR:
84                                  devolver = new PO5_DR((FCAAttribute)_A,(FCAAttribute)_B,(FCAAt
    tribute)_act);
85                                  break;
86
87                          case RCC.EQ5:
88                                  devolver = new PO5_EQ5((FCAAttribute)_A,(FCAAttribute)_B,(FCAA
    ttribute)_act);
89                                  break;
90
91                          case RCC.PP:
92                                  devolver = new PO5_PP((FCAAttribute)_A,(FCAAttribute)_B,(FCAAt
    tribute)_act);
93                                  break;
94
95                          case RCC.PPi:
96                                  devolver = new PO5_PPi((FCAAttribute)_A,(FCAAttribute)_B,(FCAA
    ttribute)_act);
97                                  break;
98
99                          default:
100                                 devolver = new Cambio_Null(_A,_B,_act);
101
102                         }
103
104                         break;
105
106
107                 case RCC.PP:
108                         switch (_nuevo) {
109                         case RCC.DR:
110                                 devolver = new PP_DR((FCAAttribute)_A,(FCAAttribute)_B,(FCAAtt
    ribute)_act);
111                                 break;
112
113                         case RCC.EQ5:
114                                 devolver = new PP_EQ5((FCAAttribute)_A,(FCAAttribute)_B,(FCAAt
    tribute)_act);
115                                 break;
116
117                         case RCC.PO5:
118                                 devolver = new PP_PO5((FCAAttribute)_A,(FCAAttribute)_B,(FCAAt
    tribute)_act);
119                                 break;
120
121                         case RCC.PPi:
122                                 devolver = new PP_PPi((FCAAttribute)_A,(FCAAttribute)_B,(FCAAt
    tribute)_act);
123                                 break;
124
125                         default:
126                                 devolver = new Cambio_Null(_A,_B,_act);
127
128                         }
129
130                         break;
131
132                 case RCC.PPi:
133                         switch (_nuevo) {
134                         case RCC.DR:
135                                 devolver = new PPi_DR((FCAAttribute)_A,(FCAAttribute)_B,(FCAAt
    tribute)_act);
136                                 break;
137
138                         case RCC.EQ5:
139                                 devolver = new PPi_EQ5((FCAAttribute)_A,(FCAAttribute)_B,(FCAA
    ttribute)_act);
140                                 break;
141
142                         case RCC.PP:
143                                 devolver = new PPi_PP((FCAAttribute)_A,(FCAAttribute)_B,(FCAAt
    tribute)_act);
144                                 break;
145
```

```
146                             case RCC.PO5:
147                                     devolver = new PPi_PO5((FCAAttribute)_A,(FCAAttribute)_B,(FCAA
    ttribute)_act);
148                                     break;
149
150                             default:
151                                     devolver = new Cambio_Null(_A,_B,_act);
152
153                             }
154
155                             break;
156
157
158
159                     case RCC.IN_P5:
160                             switch (_nuevo) {
161                             case RCC.OUT_P5:
162                                     devolver = new IN5_OUT5((FCAAttribute)_A,(FCAAttribute)_B,(FCA
    Attribute)_act);
163                                     break;
164
165                             default:
166                                     devolver = new Cambio_Null(_A,_B,_act);
167
168                             }
169
170                             break;
171
172                     case RCC.OUT_P5:
173                             switch (_nuevo) {
174                             case RCC.IN_P5:
175                                     devolver = new OUT5_IN5((FCAAttribute)_A,(FCAAttribute)_B,(FCA
    Attribute)_act);
176                                     break;
177
178                             default:
179                                     devolver = new Cambio_Null(_A,_B,_act);
180
181                             }
182
183                             break;
184
185
186
187 //                  case RCC8.OUT_P:
188 //                          switch (_nuevo) {
189 //                          case RCC8.IN_P:
190 //                                  devolver = new NP_P(_A,_B);
191 //                                  break;
192 //
193 //                          case RCC8.FRONT_P:
194 //                                  devolver = new NP_FRONT(_A,_B);
195 //                                  break;
196 //                          }
197 //                          break;
198 //
199 //
200 //                  case RCC8.IN_P:
201 //                          switch (_nuevo) {
202 //                          case RCC8.OUT_P:
203 //                                  devolver = new P_NP(_A,_B);
204 //                                  break;
205 //
206 //                          case RCC8.FRONT_P:
207 //                                  devolver = new P_FRONT(_A,_B);
208 //                                  break;
209 //                          }
210 //
211 //                          break;
212
213
214
215
216
217                     }
218
219                     if (devolver == null) {
220                             devolver = new Cambio_Null(_A,_B,_act);
221                     }
222
223
224
```

```
225                     return devolver;
226             }
227
228
229
230     }
```

```
1  package es.us.ccia.dummy.historial.cambioRCC5.pp;
2  /**
3   * Version 0.9:
4   *    20080920
5   *    − Implementacion del cambio de  (DC −> EC)
6   *
7   *================================================
8   * TODO:
9   *    −
10  *
11  */
12
13 import java.util.*;
14
15 import es.us.ccia.dummy.contexto.elementos.FCAAttribute;
16 import es.us.ccia.dummy.historial.*;
17 import es.us.ccia.dummy.razonador.*;
18 import es.us.ccia.dummy.teoria.*;
19 import es.us.ccia.rcc.RCC;
20 import es.us.tad.Salida;
21
22
23 public class PP_DR extends Cambio {
24
25         public PP_DR(FCAAttribute _a, FCAAttribute _b,FCAAttribute _act) {
26                 super(_a, _b, RCC.EC, RCC.DC, _act);
27         }
28
29
30         public void ejecutar() {
31
32
33                 //XXX:Cambio RCC8
34
35
36                 //              /**
37 //                * Esto se basa en eliminar a ACTIVO
38 //                * los elementos de la interseccion
39 //                *
40 //                */
41 //
42 //                Argumento otro;
43 //                if (getActivo() == getClaseA()) {
44 //                        otro = getClaseB();
45 //                } else {
46 //                        otro = getClaseA();
47 //                }
48 //
49 //
50 //                ArrayList a = RazonadorRCC5.get_ObjetosClase((ArgumentoC)otro).toArrayList();
51 //
52 //
53 //                Salida.Imprimirln("La clase " + otro.getNombre() + " tiene " + a.size() + " el
   ementos");
54 //
55 //
56 //
57 //
58 //                Iterator it = a.iterator();
59 //                while (it.hasNext()) {
60 //                        ArgumentoI indiv = (ArgumentoI)it.next();
61 //                        RazonadorRCC5.remove_ObjetoClase(indiv, (ArgumentoC)getActivo());
62 //                }
63
64
65
66         }
67
68 }
```

```
1  package es.us.ccia.dummy.historial.cambioRCC5.pp;
2  /**
3   * Version 0.9:
4   *    20080920
5   *    − Implementacion del cambio de  (DC −> EC)
6   *
7   *================================================
8   * TODO:
9   *    −
10  *
11  */
12
13 import es.us.ccia.dummy.contexto.elementos.FCAAttribute;
14 import es.us.ccia.dummy.historial.Cambio;
15 import es.us.ccia.dummy.teoria.*;
16 import es.us.ccia.rcc.RCC;
17
18
19 public class PP_PPi extends Cambio {
20
21         public PP_PPi(FCAAttribute _a, FCAAttribute _b,FCAAttribute _act) {
22                 super(_a, _b, RCC.EC, RCC.TPPi, _act);
23         }
24
25
26         public void ejecutar() {
27                 System.out.println(
28                                 "Ejecucion de " + getTipoviejo() +
29                                 " a " + getTiponuevo() +
30                                 " de (" +
31                                 getClaseA().getNombre() + "," +
32                                 getClaseB().getNombre() + ")");
33
34         }
35
36 }
```

```
1  package es.us.ccia.dummy.historial.cambioRCC5.pp;
2  /**
3   * Version 0.9:
4   *    20080920
5   *    − Implementacion del cambio de  (DC −> EC)
6   *
7   *================================================
8   * TODO:
9   *    −
10  *
11  */
12
13 import java.util.ArrayList;
14
15 import javax.swing.JFrame;
16
17 import es.us.ccia.dummy.DummyPaella;
18 import es.us.ccia.dummy.contexto.Contexto;
19 import es.us.ccia.dummy.contexto.elementos.FCAAttribute;
20 import es.us.ccia.dummy.dibujo.GUI;
21 import es.us.ccia.dummy.dibujo.ventanas.dialogo.*;
22 import es.us.ccia.dummy.historial.Cambio;
23 import es.us.ccia.dummy.teoria.*;
24 import es.us.ccia.rcc.RCC;
25
26 @SuppressWarnings("unchecked")
27 public class PP_PO5 extends Cambio {
28
29         public PP_PO5(FCAAttribute _a, FCAAttribute _b,FCAAttribute _act) {
30                 super(_a, _b, RCC.EC, RCC.PO, _act);
31         }
32
33
34         public void ejecutar() {
35                 System.out.println(
36                                 "Ejecucion de " + getTipoviejo() +
37                                 " a " + getTiponuevo() +
38                                 " de (" +
39                                 getClaseA().getNombre() + "," +
40                                 getClaseB().getNombre() + ")");
41
42                 //
43                 /*******
44                  *
45                  * Cambio de DC −> PO
46                  *
47                  * Se trata el elegir elementos que pertenezcan a la interseccion
48                  *
49                  * Los elementos a elegir son de la primera.. o de las 2??
50                  *
51                  *
52                  */
53
54 //              GUI g = DummyPaella.getGUI();
55 //
56 //              ArrayList a = new ArrayList();
57 //              a.add((ArgumentoC)getClaseA());
58 //              a.add((ArgumentoC)getClaseB());
59 //
60 //              ArrayList<ArgumentoI> names = Razonador.unionObjPertClases(a);
61 //
62 //
63 //              System.out.println("Los ind de la interseccion son: " + names.size());
64 //
65 //
66 //              for(int i=0;i<names.size();i++) {
67 //
68 //                      System.out.println((ArgumentoI)names.get(i));
69 //
70 //              }
71 //
72 //
73 //
74 //
75 //
76 //
77 //
78 //
79 //              ElegirAlgo.initialize(g, names, "Elegir Individuos.",
80 //                              "Elige los elementos que sean de las dos etiquetas");
81 //
82 //              ArrayList opcion = ElegirAlgo.showDialog(null, null);
```

```
83  //
84  //              if (opcion==null || opcion.size()==0) {
85  //                      System.out.println("nulo");
86  //              } else {
87  //
88  ////                    // Ahora a procesar los elementos de la ventana.
89  ////                    for (int i=0;i<opcion.length;i++) {
90  ////                            System.out.println(opcion[i] + "−" + opcion[i].getNumero());
91  ////                            Razonador.setInd2Etiq(opcion[i], (ArgumentoC)getClaseA());
92  ////                            Razonador.setInd2Etiq(opcion[i], (ArgumentoC)getClaseB());
93  ////
94  ////
95  ////
96  ////                    }
97  //
98  //                      System.out.println(Contexto.imprimir());
99  //              }
100
101
102
103
104
105         }
106
107 }
```

```
1  package es.us.ccia.dummy.historial.cambioRCC5.pp;
2  /**
3   * Version 0.9:
4   *    20080920
5   *    − Implementacion del cambio de  (DC −> EC)
6   *
7   *================================================
8   * TODO:
9   *    −
10  *
11  */
12
13 import es.us.ccia.dummy.contexto.elementos.FCAAttribute;
14 import es.us.ccia.dummy.historial.Cambio;
15 import es.us.ccia.dummy.teoria.*;
16 import es.us.ccia.rcc.RCC;
17
18
19 public class PP_EQ5 extends Cambio {
20
21         public PP_EQ5(FCAAttribute _a, FCAAttribute _b,FCAAttribute _act) {
22                 super(_a, _b, RCC.EC, RCC.EQ, _act);
23         }
24
25
26         public void ejecutar() {
27                 System.out.println(
28                                 "Ejecucion de " + getTipoviejo() +
29                                 " a " + getTiponuevo() +
30                                 " de (" +
31                                 getClaseA().getNombre() + "," +
32                                 getClaseB().getNombre() + ")");
33
34         }
35
36 }
```

```
1  package es.us.ccia.dummy.historial.cambioRCC5.dr;
2  /**
3   * Version 0.9:
4   *    20080920
5   *    − Implementacion del cambio de  (DC −> EC)
6   *
7   *================================================
8   * TODO:
9   *    −
10  *
11  */
12
13 import java.util.Collection;
14 import java.util.Iterator;
15 import java.util.LinkedList;
16
17
18 import es.us.ccia.dummy.DummyPaella;
19 import es.us.ccia.dummy.contexto.Contexto;
20 import es.us.ccia.dummy.contexto.elementos.FCAAttribute;
21 import es.us.ccia.dummy.contexto.elementos.FCAElement;
22 import es.us.ccia.dummy.contexto.elementos.FCAObject;
23 import es.us.ccia.dummy.dibujo.GUI;
24 import es.us.ccia.dummy.dibujo.ventanas.dialogo.*;
25 import es.us.ccia.dummy.historial.Cambio;
26 import es.us.ccia.dummy.teoria.*;
27 import es.us.ccia.rcc.RCC;
28
29 @SuppressWarnings("unchecked")
30 public class DR_PO5 extends Cambio {
31         /*******
32          *
33          * Cambio de DC −> PO
34          *
35          * 1. Se crea una etiqueta que va a estar en la interseccion
36          *
37          * 2. Se eligen los elementos que pertenecen a la interseccion
38          *
39          */
40
41         public DR_PO5(FCAAttribute _a, FCAAttribute _b,FCAAttribute _act) {
42                 super(_a, _b, RCC.DC, RCC.PO,_act);
43         }
44
45
46         public void ejecutar() {
47
48                 GUI g = DummyPaella.getGUI();
49
50                 ////////////////////////////////////
51                 //   CREAR CLASE
52                 ////////////////////////////////////
53
54                 Collection names = new LinkedList();
55                 names.addAll(Contexto.get_ObjetosClase((FCAAttribute)getClaseA()));
56                 names.addAll(Contexto.get_ObjetosClase((FCAAttribute)getClaseB()));
57
58                 NuevaClase.initialize(g, names.toArray(), "Create Tag");
59
60                 //  FCAAttribute opcion = NuevaClase.showDialog(g, null);
61                 FCAAttribute opcion = null; //NuevaClase.showDialog(g, null);
62
63
64                 ////////////////////////////////////////////////////
65                 //Ahora hay que aÃ±adir los objetos de 'opcion' a A y B
66                 ////////////////////////////////////////////////////
67
68                 Collection la = Contexto.get_ObjetosClase(opcion);
69                 Iterator it = la.iterator();
70                 while (it.hasNext()) {
71                         FCAObject ind = (FCAObject)it.next();
72                         Contexto.addRelation((FCAAttribute)getClaseA(), ind);
73                         Contexto.addRelation((FCAAttribute)getClaseB(), ind);
74                 }
75 //              g.getVentanaEtiquetas().actualizar();
76         }
77
78 }
```

```
1  package es.us.ccia.dummy.historial.cambioRCC5.dr;
2  /**
3   * Version 0.9:
4   *    20080920
5   *    − Implementacion del cambio de  (DC −> EC)
6   *
7   *================================================
8   * TODO:
9   *    −
10  *
11  */
12
13 import es.us.ccia.dummy.contexto.elementos.*;
14 import es.us.ccia.dummy.historial.Cambio;
15 import es.us.ccia.dummy.teoria.*;
16 import es.us.ccia.rcc.RCC;
17
18
19 public class DR_PP extends Cambio {
20
21         public DR_PP(FCAAttribute _a, FCAAttribute _b,FCAAttribute _act) {
22                 super(_a, _b, RCC.DC, RCC.NTPP,_act);
23         }
24
25
26         public void ejecutar() {
27
28
29                 //XXX: ESTE CAMBIO NO SE REALIZA POR SER RCC8
30                 //XXX: Deberiamos de poner una ventana que lo avisara.
31
32
33
34
35
36
37
38
39
40
41 //              /*****
42 //               * Primero hay que incluir todo lo de la clase A en B
43 //               */
44 //
45 //              Collection a  = Contexto.get_ObjetosClase(getClaseA());
46 //
47 //              Iterator it3 = a.iterator();
48 //              while (it3.hasNext()) {
49 //                      FCAObject indiv = (FCAObject)it3.next();
50 //                      Contexto.addRelation(indiv, getClaseB());
51 //              }
52 //
53 //
54 //              ArrayList eca = RazonadorRCC5.getEC((ArgumentoC)getClaseA());
55 //              ArrayList ecb = RazonadorRCC5.getEC((ArgumentoC)getClaseB());
56 //
57 //              ecb.retainAll(eca);  // EN ecb queda la interseccion.
58 //              /**
59 //               *
60 //               * Si ecb NO es vacia... entonces hay que eliminar los elementos
61 //               * de la clase interior, que pertenezcan a cada uno de ecb
62 //               *
63 //               */
64 //
65 //              Iterator it = ecb.iterator();
66 //              while (it.hasNext()) {
67 //                      ArgumentoC clase = (ArgumentoC)it.next();
68 //                      ArrayList inds = RazonadorRCC5.get_ObjetosClase(clase).toArrayList();
69 //                      Iterator it2 = inds.iterator();
70 //                      while (it2.hasNext()) {
71 //                              ArgumentoI indiv = (ArgumentoI)it2.next();
72 //                              RazonadorRCC5.remove_ObjetoClase(indiv, clase);
73 //                      }
74 //              }
75
76
77
78
79
80
81         }
82
```

```
83  }
```

```
1  package es.us.ccia.dummy.historial.cambioRCC5.dr;
2  /**
3   * Version 0.9:
4   *    20080920
5   *    − Implementacion del cambio de  (DC −> EC)
6   *
7   *================================================
8   * TODO:
9   *    −
10  *
11  */
12
13 import java.util.Collection;
14 import java.util.Iterator;
15
16 import es.us.ccia.dummy.contexto.Contexto;
17 import es.us.ccia.dummy.contexto.elementos.FCAAttribute;
18 import es.us.ccia.dummy.contexto.elementos.FCAObject;
19 import es.us.ccia.dummy.historial.Cambio;
20 import es.us.ccia.dummy.teoria.*;
21 import es.us.ccia.rcc.RCC;
22
23
24 @SuppressWarnings("unchecked")
25 public class DR_EQ5 extends Cambio {
26
27         public DR_EQ5(FCAAttribute _a, FCAAttribute _b,FCAAttribute _act) {
28                 super(_a, _b, RCC.DC, RCC.EQ,_act);
29         }
30
31
32         public void ejecutar() {
33
34                 Collection la = Contexto.get_ObjetosClase((FCAAttribute)getClaseA());
35                 Iterator it = la.iterator();
36                 while (it.hasNext()) {
37                         FCAObject ind = (FCAObject)it.next();
38                         Contexto.addRelation((FCAAttribute)getClaseB(), ind);
39                 }
40                 Collection lb = Contexto.get_ObjetosClase((FCAAttribute)getClaseB());
41                 Iterator it2 = lb.iterator();
42                 while (it2.hasNext()) {
43                         FCAObject ind = (FCAObject)it2.next();
44                         Contexto.addRelation((FCAAttribute)getClaseA(),ind);
45                 }
46
47         }
48
49 }
```

```
1  package es.us.ccia.dummy.historial.cambioRCC5.dr;
2  /**
3   * Version 0.9:
4   *    20080920
5   *    − Implementacion del cambio de  (DC −> EC)
6   *
7   *================================================
8   * TODO:
9   *    −
10  *
11  */
12
13 import es.us.ccia.dummy.contexto.elementos.FCAAttribute;
14 import es.us.ccia.dummy.historial.Cambio;
15 import es.us.ccia.dummy.teoria.*;
16 import es.us.ccia.rcc.RCC;
17
18
19 public class DR_PPi extends Cambio {
20
21         public DR_PPi(FCAAttribute _a, FCAAttribute _b,FCAAttribute _act) {
22                 super(_a, _b, RCC.DC, RCC.NTPPi,_act);
23         }
24
25
26         public void ejecutar() {
27                 DR_PP cambio = new DR_PP(
28                                 (FCAAttribute)getClaseB(),
29                                 (FCAAttribute)getClaseA(),
30                                 (FCAAttribute)getActivo());
31                 cambio.ejecutar();
32         }
33
34 }
```

```
1  package es.us.ccia.dummy.historial.cambioRCC5.outin;
2  /**
3   * Version 0.9:
4   *    20080920
5   *    − Implementacion del cambio de  (DC −> EC)
6   *
7   *================================================
8   * TODO:
9   *    −
10  *
11  */
12
13 import es.us.ccia.dummy.contexto.elementos.FCAAttribute;
14 import es.us.ccia.dummy.historial.Cambio;
15 import es.us.ccia.dummy.teoria.*;
16 import es.us.ccia.rcc.RCC;
17
18
19 public class IN5_OUT5 extends Cambio {
20
21         public IN5_OUT5(FCAAttribute _a, FCAAttribute _b,FCAAttribute _act) {
22                 super(_a, _b, RCC.EC, RCC.EQ, _act);
23         }
24
25
26         public void ejecutar() {
27                 System.out.println(
28                                 "Ejecucion de " + getTipoviejo() +
29                                 " a " + getTiponuevo() +
30                                 " de (" +
31                                 getClaseA().getNombre() + "," +
32                                 getClaseB().getNombre() + ")");
33
34         }
35
36 }
```

```
1  package es.us.ccia.dummy.historial.cambioRCC5.outin;
2  /**
3   * Version 0.9:
4   *    20080920
5   *    − Implementacion del cambio de  (DC −> EC)
6   *
7   *================================================
8   * TODO:
9   *    −
10  *
11  */
12
13 import java.util.*;
14
15 import es.us.ccia.dummy.contexto.elementos.FCAAttribute;
16 import es.us.ccia.dummy.historial.*;
17 import es.us.ccia.dummy.razonador.*;
18 import es.us.ccia.dummy.teoria.*;
19 import es.us.ccia.rcc.RCC;
20 import es.us.tad.Salida;
21
22
23 public class OUT5_IN5 extends Cambio {
24
25         public OUT5_IN5(FCAAttribute _a, FCAAttribute _b,FCAAttribute _act) {
26                 super(_a, _b, RCC.EC, RCC.DC, _act);
27         }
28
29
30         public void ejecutar() {
31
32
33                 //XXX:Cambio RCC8
34
35
36                 //              /**
37 //                * Esto se basa en eliminar a ACTIVO
38 //                * los elementos de la interseccion
39 //                *
40 //                */
41 //
42 //                Argumento otro;
43 //                if (getActivo() == getClaseA()) {
44 //                        otro = getClaseB();
45 //                } else {
46 //                        otro = getClaseA();
47 //                }
48 //
49 //
50 //                ArrayList a = RazonadorRCC5.get_ObjetosClase((ArgumentoC)otro).toArrayList();
51 //
52 //
53 //                Salida.Imprimirln("La clase " + otro.getNombre() + " tiene " + a.size() + " el
   ementos");
54 //
55 //
56 //
57 //
58 //                Iterator it = a.iterator();
59 //                while (it.hasNext()) {
60 //                        ArgumentoI indiv = (ArgumentoI)it.next();
61 //                        RazonadorRCC5.remove_ObjetoClase(indiv, (ArgumentoC)getActivo());
62 //                }
63
64
65
66         }
67
68 }
```

```
1  package es.us.ccia.dummy.historial.cambioRCC5.po5;
2  /**
3   * Version 0.9:
4   *    20080920
5   *    − Implementacion del cambio de  (DC −> EC)
6   *
7   *================================================
8   * TODO:
9   *    −
10  *
11  */
12
13 import es.us.ccia.dummy.contexto.elementos.FCAAttribute;
14 import es.us.ccia.dummy.historial.Cambio;
15 import es.us.ccia.dummy.teoria.*;
16 import es.us.ccia.rcc.RCC;
17
18
19 public class PO5_PP extends Cambio {
20
21         public PO5_PP(FCAAttribute _a, FCAAttribute _b,FCAAttribute _act) {
22                 super(_a, _b, RCC.EC, RCC.TPP, _act);
23         }
24
25
26         public void ejecutar() {
27                 System.out.println(
28                                 "Ejecucion de " + getTipoviejo() +
29                                 " a " + getTiponuevo() +
30                                 " de (" +
31                                 getClaseA().getNombre() + "," +
32                                 getClaseB().getNombre() + ")");
33
34         }
35
36 }
```

```
1  package es.us.ccia.dummy.historial.cambioRCC5.po5;
2  /**
3   * Version 0.9:
4   *    20080920
5   *    − Implementacion del cambio de  (DC −> EC)
6   *
7   *================================================
8   * TODO:
9   *    −
10  *
11  */
12
13 import java.util.*;
14
15 import es.us.ccia.dummy.contexto.elementos.FCAAttribute;
16 import es.us.ccia.dummy.historial.*;
17 import es.us.ccia.dummy.razonador.*;
18 import es.us.ccia.dummy.teoria.*;
19 import es.us.ccia.rcc.RCC;
20 import es.us.tad.Salida;
21
22
23 public class PO5_DR extends Cambio {
24
25         public PO5_DR(FCAAttribute _a, FCAAttribute _b,FCAAttribute _act) {
26                 super(_a, _b, RCC.EC, RCC.DC, _act);
27         }
28
29
30         public void ejecutar() {
31
32
33                 //XXX:Cambio RCC8
34
35
36                 //              /**
37 //                * Esto se basa en eliminar a ACTIVO
38 //                * los elementos de la interseccion
39 //                *
40 //                */
41 //
42 //                Argumento otro;
43 //                if (getActivo() == getClaseA()) {
44 //                        otro = getClaseB();
45 //                } else {
46 //                        otro = getClaseA();
47 //                }
48 //
49 //
50 //                ArrayList a = RazonadorRCC5.get_ObjetosClase((ArgumentoC)otro).toArrayList();
51 //
52 //
53 //                Salida.Imprimirln("La clase " + otro.getNombre() + " tiene " + a.size() + " el
   ementos");
54 //
55 //
56 //
57 //
58 //                Iterator it = a.iterator();
59 //                while (it.hasNext()) {
60 //                        ArgumentoI indiv = (ArgumentoI)it.next();
61 //                        RazonadorRCC5.remove_ObjetoClase(indiv, (ArgumentoC)getActivo());
62 //                }
63
64
65
66         }
67
68 }
```

```
1  package es.us.ccia.dummy.historial.cambioRCC5.po5;
2  /**
3   * Version 0.9:
4   *    20080920
5   *    − Implementacion del cambio de  (DC −> EC)
6   *
7   *================================================
8   * TODO:
9   *    −
10  *
11  */
12
13 import es.us.ccia.dummy.contexto.elementos.FCAAttribute;
14 import es.us.ccia.dummy.historial.Cambio;
15 import es.us.ccia.dummy.teoria.*;
16 import es.us.ccia.rcc.RCC;
17
18
19 public class PO5_PPi extends Cambio {
20
21         public PO5_PPi(FCAAttribute _a, FCAAttribute _b,FCAAttribute _act) {
22                 super(_a, _b, RCC.EC, RCC.TPPi, _act);
23         }
24
25
26         public void ejecutar() {
27                 System.out.println(
28                                 "Ejecucion de " + getTipoviejo() +
29                                 " a " + getTiponuevo() +
30                                 " de (" +
31                                 getClaseA().getNombre() + "," +
32                                 getClaseB().getNombre() + ")");
33
34         }
35
36 }
```

```
1  package es.us.ccia.dummy.historial.cambioRCC5.po5;
2  /**
3   * Version 0.9:
4   *    20080920
5   *    − Implementacion del cambio de  (DC −> EC)
6   *
7   *================================================
8   * TODO:
9   *    −
10  *
11  */
12
13 import es.us.ccia.dummy.contexto.elementos.FCAAttribute;
14 import es.us.ccia.dummy.historial.Cambio;
15 import es.us.ccia.dummy.teoria.*;
16 import es.us.ccia.rcc.RCC;
17
18
19 public class PO5_EQ5 extends Cambio {
20
21         public PO5_EQ5(FCAAttribute _a, FCAAttribute _b,FCAAttribute _act) {
22                 super(_a, _b, RCC.EC, RCC.EQ, _act);
23         }
24
25
26         public void ejecutar() {
27                 System.out.println(
28                                 "Ejecucion de " + getTipoviejo() +
29                                 " a " + getTiponuevo() +
30                                 " de (" +
31                                 getClaseA().getNombre() + "," +
32                                 getClaseB().getNombre() + ")");
33
34         }
35
36 }
```

```
1  package es.us.ccia.dummy.historial.cambioRCC5;
2  /**
3   * Version 0.9:
4   *    20080920
5   *    − Implementacion del cambio de  (DC −> EC)
6   *
7   *================================================
8   * TODO:
9   *    −
10  *
11  */
12
13 import es.us.ccia.dummy.contexto.elementos.FCAElement;
14 import es.us.ccia.dummy.historial.Cambio;
15 import es.us.ccia.rcc.*;
16 import es.us.tad.Salida;
17
18
19 public class Cambio_Null extends Cambio {
20
21         public Cambio_Null(FCAElement _a, FCAElement _b,FCAElement _act) {
22                 super(_a, _b, RCC.NODEF, RCC.NODEF,_act);
23         }
24
25
26         public void ejecutar() {
27                 Salida.Imprimirln(
28                                 "Ejecucion de " + getTipoviejo() +
29                                 " a " + getTiponuevo() +
30                                 " de (" +
31                                 getClaseA().getNombre() + "," +
32                                 getClaseB().getNombre() + ")");
33         }
34
35         public String toString() {
36                 return "ALERT: " +
37                                 "(" +
38                             super.getClaseA() + "," + super.getClaseB() +
39                         ") " +
40                         super.getTipoviejo() + " −> " + super.getTiponuevo();
41
42
43
44         }
45
46 }
```

```java
package es.us.ccia.dummy.historial.cambioRCC5.eq5;
/**
 * Version 0.9:
 *    20080920
 *    − Implementacion del cambio de  (DC −> EC)
 *
 *================================================
 * TODO:
 *    −
 *
 */

import es.us.ccia.dummy.contexto.elementos.FCAAttribute;
import es.us.ccia.dummy.historial.Cambio;
import es.us.ccia.dummy.teoria.*;
import es.us.ccia.rcc.RCC;


public class EQ5_PPi extends Cambio {

        public EQ5_PPi(FCAAttribute _a, FCAAttribute _b,FCAAttribute _act) {
                super(_a, _b, RCC.EC, RCC.TPPi, _act);
        }


        public void ejecutar() {
                System.out.println(
                                "Ejecucion de " + getTipoviejo() +
                                " a " + getTiponuevo() +
                                " de (" +
                                getClaseA().getNombre() + "," +
                                getClaseB().getNombre() + ")");

        }

}
```

```
1  package es.us.ccia.dummy.historial.cambioRCC5.eq5;
2  /**
3   * Version 0.9:
4   *    20080920
5   *    − Implementacion del cambio de  (DC −> EC)
6   *
7   *================================================
8   * TODO:
9   *    −
10  *
11  */
12
13 import es.us.ccia.dummy.contexto.elementos.FCAAttribute;
14 import es.us.ccia.dummy.historial.Cambio;
15 import es.us.ccia.dummy.teoria.*;
16 import es.us.ccia.rcc.RCC;
17
18
19 public class EQ5_PP extends Cambio {
20
21         public EQ5_PP(FCAAttribute _a, FCAAttribute _b,FCAAttribute _act) {
22                 super(_a, _b, RCC.EC, RCC.TPP, _act);
23         }
24
25
26         public void ejecutar() {
27                 System.out.println(
28                                 "Ejecucion de " + getTipoviejo() +
29                                 " a " + getTiponuevo() +
30                                 " de (" +
31                                 getClaseA().getNombre() + "," +
32                                 getClaseB().getNombre() + ")");
33
34         }
35
36 }
```

```
1  package es.us.ccia.dummy.historial.cambioRCC5.eq5;
2  /**
3   * Version 0.9:
4   *    20080920
5   *    − Implementacion del cambio de  (DC −> EC)
6   *
7   *================================================
8   * TODO:
9   *    −
10  *
11  */
12
13 import java.util.*;
14
15 import es.us.ccia.dummy.contexto.elementos.FCAAttribute;
16 import es.us.ccia.dummy.historial.*;
17 import es.us.ccia.dummy.razonador.*;
18 import es.us.ccia.dummy.teoria.*;
19 import es.us.ccia.rcc.RCC;
20 import es.us.tad.Salida;
21
22
23 public class EQ5_DR extends Cambio {
24
25         public EQ5_DR(FCAAttribute _a, FCAAttribute _b,FCAAttribute _act) {
26                 super(_a, _b, RCC.EC, RCC.DC, _act);
27         }
28
29
30         public void ejecutar() {
31
32
33                 //XXX:Cambio RCC8
34
35
36                 //              /**
37 //                * Esto se basa en eliminar a ACTIVO
38 //                * los elementos de la interseccion
39 //                *
40 //                */
41 //
42 //                Argumento otro;
43 //                if (getActivo() == getClaseA()) {
44 //                        otro = getClaseB();
45 //                } else {
46 //                        otro = getClaseA();
47 //                }
48 //
49 //
50 //                ArrayList a = RazonadorRCC5.get_ObjetosClase((ArgumentoC)otro).toArrayList();
51 //
52 //
53 //                Salida.Imprimirln("La clase " + otro.getNombre() + " tiene " + a.size() + " el
   ementos");
54 //
55 //
56 //
57 //
58 //                Iterator it = a.iterator();
59 //                while (it.hasNext()) {
60 //                        ArgumentoI indiv = (ArgumentoI)it.next();
61 //                        RazonadorRCC5.remove_ObjetoClase(indiv, (ArgumentoC)getActivo());
62 //                }
63
64
65
66         }
67
68 }
```

```
1  package es.us.ccia.dummy.historial.cambioRCC5.eq5;
2  /**
3   * Version 0.9:
4   *    20080920
5   *    − Implementacion del cambio de  (DC −> EC)
6   *
7   *================================================
8   * TODO:
9   *    −
10  *
11  */
12
13 import java.util.ArrayList;
14
15 import javax.swing.JFrame;
16
17 import es.us.ccia.dummy.DummyPaella;
18 import es.us.ccia.dummy.contexto.Contexto;
19 import es.us.ccia.dummy.contexto.elementos.FCAAttribute;
20 import es.us.ccia.dummy.dibujo.GUI;
21 import es.us.ccia.dummy.dibujo.ventanas.dialogo.*;
22 import es.us.ccia.dummy.historial.Cambio;
23 import es.us.ccia.dummy.teoria.*;
24 import es.us.ccia.rcc.RCC;
25
26 @SuppressWarnings("unchecked")
27 public class EQ5_PO5 extends Cambio {
28
29         public EQ5_PO5(FCAAttribute _a, FCAAttribute _b,FCAAttribute _act) {
30                 super(_a, _b, RCC.EC, RCC.PO, _act);
31         }
32
33
34         public void ejecutar() {
35                 System.out.println(
36                                 "Ejecucion de " + getTipoviejo() +
37                                 " a " + getTiponuevo() +
38                                 " de (" +
39                                 getClaseA().getNombre() + "," +
40                                 getClaseB().getNombre() + ")");
41
42                 //
43                 /*******
44                  *
45                  * Cambio de DC −> PO
46                  *
47                  * Se trata el elegir elementos que pertenezcan a la interseccion
48                  *
49                  * Los elementos a elegir son de la primera.. o de las 2??
50                  *
51                  *
52                  */
53
54 //              GUI g = DummyPaella.getGUI();
55 //
56 //              ArrayList a = new ArrayList();
57 //              a.add((ArgumentoC)getClaseA());
58 //              a.add((ArgumentoC)getClaseB());
59 //
60 //              ArrayList<ArgumentoI> names = Razonador.unionObjPertClases(a);
61 //
62 //
63 //              System.out.println("Los ind de la interseccion son: " + names.size());
64 //
65 //
66 //              for(int i=0;i<names.size();i++) {
67 //
68 //                      System.out.println((ArgumentoI)names.get(i));
69 //
70 //              }
71 //
72 //
73 //
74 //
75 //
76 //
77 //
78 //
79 //              ElegirAlgo.initialize(g, names, "Elegir Individuos.",
80 //                              "Elige los elementos que sean de las dos etiquetas");
81 //
82 //              ArrayList opcion = ElegirAlgo.showDialog(null, null);
```

```
83  //
84  //              if (opcion==null || opcion.size()==0) {
85  //                      System.out.println("nulo");
86  //              } else {
87  //
88  ////                    // Ahora a procesar los elementos de la ventana.
89  ////                    for (int i=0;i<opcion.length;i++) {
90  ////                            System.out.println(opcion[i] + "−" + opcion[i].getNumero());
91  ////                            Razonador.setInd2Etiq(opcion[i], (ArgumentoC)getClaseA());
92  ////                            Razonador.setInd2Etiq(opcion[i], (ArgumentoC)getClaseB());
93  ////
94  ////
95  ////
96  ////                    }
97  //
98  //                      System.out.println(Contexto.imprimir());
99  //              }
100
101
102
103
104
105         }
106
107 }
```

```
1  package es.us.ccia.dummy.historial.cambioRCC5.ppi;
2  /**
3   * Version 0.9:
4   *    20080920
5   *    − Implementacion del cambio de  (DC −> EC)
6   *
7   *================================================
8   * TODO:
9   *    −
10  *
11  */
12
13 import java.util.ArrayList;
14
15 import javax.swing.JFrame;
16
17 import es.us.ccia.dummy.DummyPaella;
18 import es.us.ccia.dummy.contexto.Contexto;
19 import es.us.ccia.dummy.contexto.elementos.FCAAttribute;
20 import es.us.ccia.dummy.dibujo.GUI;
21 import es.us.ccia.dummy.dibujo.ventanas.dialogo.*;
22 import es.us.ccia.dummy.historial.Cambio;
23 import es.us.ccia.dummy.teoria.*;
24 import es.us.ccia.rcc.RCC;
25
26 @SuppressWarnings("unchecked")
27 public class PPi_PO5 extends Cambio {
28
29         public PPi_PO5(FCAAttribute _a, FCAAttribute _b,FCAAttribute _act) {
30                 super(_a, _b, RCC.EC, RCC.PO, _act);
31         }
32
33
34         public void ejecutar() {
35                 System.out.println(
36                                 "Ejecucion de " + getTipoviejo() +
37                                 " a " + getTiponuevo() +
38                                 " de (" +
39                                 getClaseA().getNombre() + "," +
40                                 getClaseB().getNombre() + ")");
41
42                 //
43                 /*******
44                  *
45                  * Cambio de DC −> PO
46                  *
47                  * Se trata el elegir elementos que pertenezcan a la interseccion
48                  *
49                  * Los elementos a elegir son de la primera.. o de las 2??
50                  *
51                  *
52                  */
53
54 //              GUI g = DummyPaella.getGUI();
55 //
56 //              ArrayList a = new ArrayList();
57 //              a.add((ArgumentoC)getClaseA());
58 //              a.add((ArgumentoC)getClaseB());
59 //
60 //              ArrayList<ArgumentoI> names = Razonador.unionObjPertClases(a);
61 //
62 //
63 //              System.out.println("Los ind de la interseccion son: " + names.size());
64 //
65 //
66 //              for(int i=0;i<names.size();i++) {
67 //
68 //                      System.out.println((ArgumentoI)names.get(i));
69 //
70 //              }
71 //
72 //
73 //
74 //
75 //
76 //
77 //
78 //
79 //              ElegirAlgo.initialize(g, names, "Elegir Individuos.",
80 //                              "Elige los elementos que sean de las dos etiquetas");
81 //
82 //              ArrayList opcion = ElegirAlgo.showDialog(null, null);
```

```
83  //
84  //              if (opcion==null || opcion.size()==0) {
85  //                      System.out.println("nulo");
86  //              } else {
87  //
88  ////                    // Ahora a procesar los elementos de la ventana.
89  ////                    for (int i=0;i<opcion.length;i++) {
90  ////                            System.out.println(opcion[i] + "−" + opcion[i].getNumero());
91  ////                            Razonador.setInd2Etiq(opcion[i], (ArgumentoC)getClaseA());
92  ////                            Razonador.setInd2Etiq(opcion[i], (ArgumentoC)getClaseB());
93  ////
94  ////
95  ////
96  ////                    }
97  //
98  //                      System.out.println(Contexto.imprimir());
99  //              }
100
101
102
103
104
105         }
106
107 }
```

```
1  package es.us.ccia.dummy.historial.cambioRCC5.ppi;
2  /**
3   * Version 0.9:
4   *    20080920
5   *    − Implementacion del cambio de  (DC −> EC)
6   *
7   *================================================
8   * TODO:
9   *    −
10  *
11  */
12
13 import java.util.*;
14
15 import es.us.ccia.dummy.contexto.elementos.FCAAttribute;
16 import es.us.ccia.dummy.historial.*;
17 import es.us.ccia.dummy.razonador.*;
18 import es.us.ccia.dummy.teoria.*;
19 import es.us.ccia.rcc.RCC;
20 import es.us.tad.Salida;
21
22
23 public class PPi_DR extends Cambio {
24
25         public PPi_DR(FCAAttribute _a, FCAAttribute _b,FCAAttribute _act) {
26                 super(_a, _b, RCC.EC, RCC.DC, _act);
27         }
28
29
30         public void ejecutar() {
31
32
33                 //XXX:Cambio RCC8
34
35
36                 //              /**
37 //                * Esto se basa en eliminar a ACTIVO
38 //                * los elementos de la interseccion
39 //                *
40 //                */
41 //
42 //                Argumento otro;
43 //                if (getActivo() == getClaseA()) {
44 //                        otro = getClaseB();
45 //                } else {
46 //                        otro = getClaseA();
47 //                }
48 //
49 //
50 //                ArrayList a = RazonadorRCC5.get_ObjetosClase((ArgumentoC)otro).toArrayList();
51 //
52 //
53 //                Salida.Imprimirln("La clase " + otro.getNombre() + " tiene " + a.size() + " el
   ementos");
54 //
55 //
56 //
57 //
58 //                Iterator it = a.iterator();
59 //                while (it.hasNext()) {
60 //                        ArgumentoI indiv = (ArgumentoI)it.next();
61 //                        RazonadorRCC5.remove_ObjetoClase(indiv, (ArgumentoC)getActivo());
62 //                }
63
64
65
66         }
67
68 }
```

```
1  package es.us.ccia.dummy.historial.cambioRCC5.ppi;
2  /**
3   * Version 0.9:
4   *    20080920
5   *    − Implementacion del cambio de  (DC −> EC)
6   *
7   *================================================
8   * TODO:
9   *    −
10  *
11  */
12
13 import es.us.ccia.dummy.contexto.elementos.FCAAttribute;
14 import es.us.ccia.dummy.historial.Cambio;
15 import es.us.ccia.dummy.teoria.*;
16 import es.us.ccia.rcc.RCC;
17
18
19 public class PPi_PP extends Cambio {
20
21         public PPi_PP(FCAAttribute _a, FCAAttribute _b,FCAAttribute _act) {
22                 super(_a, _b, RCC.EC, RCC.TPP, _act);
23         }
24
25
26         public void ejecutar() {
27                 System.out.println(
28                                 "Ejecucion de " + getTipoviejo() +
29                                 " a " + getTiponuevo() +
30                                 " de (" +
31                                 getClaseA().getNombre() + "," +
32                                 getClaseB().getNombre() + ")");
33
34         }
35
36 }
```

```
1  package es.us.ccia.dummy.historial.cambioRCC5.ppi;
2  /**
3   * Version 0.9:
4   *    20080920
5   *    − Implementacion del cambio de  (DC −> EC)
6   *
7   *================================================
8   * TODO:
9   *    −
10  *
11  */
12
13 import es.us.ccia.dummy.contexto.elementos.FCAAttribute;
14 import es.us.ccia.dummy.historial.Cambio;
15 import es.us.ccia.dummy.teoria.*;
16 import es.us.ccia.rcc.RCC;
17
18
19 public class PPi_EQ5 extends Cambio {
20
21         public PPi_EQ5(FCAAttribute _a, FCAAttribute _b,FCAAttribute _act) {
22                 super(_a, _b, RCC.EC, RCC.EQ, _act);
23         }
24
25
26         public void ejecutar() {
27                 System.out.println(
28                                 "Ejecucion de " + getTipoviejo() +
29                                 " a " + getTiponuevo() +
30                                 " de (" +
31                                 getClaseA().getNombre() + "," +
32                                 getClaseB().getNombre() + ")");
33
34         }
35
36 }
```

```
1  package es.us.ccia.dummy.historial;
2
3  import es.us.ccia.dummy.contexto.elementos.FCAElement;
4  import es.us.ccia.dummy.historial.cambioRCC5.Cambio_Null;
5
6
7  public class H_Cambios8 {
8
9
10
11
12
13         public static Cambio getCambio(FCAElement _A, FCAElement _B,
14                         int _anterior, int _nuevo, FCAElement _act) {
15
16                 Cambio devolver = null;
17
18 //              switch (_anterior) {
19 //
20 //              case RCC.DC:
21 //                      switch (_nuevo) {
22 //                      case RCC.EC:
23 //                              devolver = new DC_EC((FCAAttribute)_A,(FCAAttribute)_B,(FCAAtt
   ribute)_act);
24 //                              break;
25 //
26 //                      case RCC.PO:
27 //                              devolver = new DR_PO5((FCAAttribute)_A,(FCAAttribute)_B,(FCAAt
   tribute)_act);
28 //                              break;
29 //
30 //                      case RCC.NTPP:
31 //                              devolver = new DR_PP((FCAAttribute)_A,(FCAAttribute)_B,(FCAAtt
   ribute)_act);
32 //                              break;
33 //
34 //                      case RCC.NTPPi:
35 //                              devolver = new DR_PPi((FCAAttribute)_A,(FCAAttribute)_B,(FCAAt
   tribute)_act);
36 //                              break;
37 //
38 //                      case RCC.TPP:
39 //                              devolver = new DC_TPP((FCAAttribute)_A,(FCAAttribute)_B,(FCAAt
   tribute)_act);
40 //                              break;
41 //
42 //                      case RCC.TPPi:
43 //                              devolver = new DC_TPPi((FCAAttribute)_A,(FCAAttribute)_B,(FCAA
   ttribute)_act);
44 //                              break;
45 //
46 //                      case RCC.EQ:
47 //                              devolver = new DR_EQ5((FCAAttribute)_A,(FCAAttribute)_B,(FCAAt
   tribute)_act);
48 //                              break;
49 //
50 //
51 //                      default:
52 //                              devolver = new Cambio_Null(_A,_B,_act);
53 //
54 //                      }
55 //
56 //                      break;
57 //
58 //
59 //
60 //              case RCC.EC:
61 //                      switch (_nuevo) {
62 //                      case RCC.DC:
63 //                              devolver = new EC_DC((FCAAttribute)_A,(FCAAttribute)_B,(FCAAtt
   ribute)_act);
64 //                              break;
65 //
66 //                      case RCC.PO:
67 //                              devolver = new EC_PO((FCAAttribute)_A,(FCAAttribute)_B,(FCAAtt
   ribute)_act);
68 //                              break;
69 //
70 //                      case RCC.NTPP:
71 //                              devolver = new EC_NTPP((FCAAttribute)_A,(FCAAttribute)_B,(FCAA
   ttribute)_act);
72 //                              break;
```

```
73  //
74  //                          case RCC.NTPPi:
75  //                                  devolver = new EC_NTPPi((FCAAttribute)_A,(FCAAttribute)_B,(FCA
    Attribute)_act);
76  //                                  break;
77  //
78  //                          case RCC.TPP:
79  //                                  devolver = new EC_TPP((FCAAttribute)_A,(FCAAttribute)_B,(FCAAt
    tribute)_act);
80  //                                  break;
81  //
82  //                          case RCC.TPPi:
83  //                                  devolver = new EC_TPPi((FCAAttribute)_A,(FCAAttribute)_B,(FCAA
    ttribute)_act);
84  //                                  break;
85  //
86  //                          case RCC.EQ:
87  //                                  devolver = new EC_EQ((FCAAttribute)_A,(FCAAttribute)_B,(FCAAtt
    ribute)_act);
88  //                                  break;
89  //
90  //
91  //                          default:
92  //                                  devolver = new Cambio_Null(_A,_B,_act);
93  //
94  //                          }
95  //
96  //                          break;
97  //
98  //
99  //
100 //
101 ////             case RCC8.OUT_P:
102 ////                     switch (_nuevo) {
103 ////                     case RCC8.IN_P:
104 ////                             devolver = new NP_P(_A,_B);
105 ////                             break;
106 ////
107 ////                     case RCC8.FRONT_P:
108 ////                             devolver = new NP_FRONT(_A,_B);
109 ////                             break;
110 ////                     }
111 ////                     break;
112 ////
113 ////
114 ////             case RCC8.IN_P:
115 ////                     switch (_nuevo) {
116 ////                     case RCC8.OUT_P:
117 ////                             devolver = new P_NP(_A,_B);
118 ////                             break;
119 ////
120 ////                     case RCC8.FRONT_P:
121 ////                             devolver = new P_FRONT(_A,_B);
122 ////                             break;
123 ////                     }
124 ////
125 ////                     break;
126 //
127 //
128 //
129 //
130 //
131 //              }
132
133                 if (devolver == null) {
134                         devolver = new Cambio_Null(_A,_B,_act);
135                 }
136
137
138
139                 return devolver;
140         }
141
142
143
144 }
```

```
1  package es.us.ccia.dummy.historial;
2  /**
3   * Version 0.9:
4   *    20080920
5   *    − Creacion de una clase abstracta para llamar
6   *      a cualquiera de los cambios
7   *
8   *================================================
9   * TODO:
10  *    −
11  *
12  */
13
14
15 import es.us.ccia.dummy.contexto.elementos.FCAElement;
16 import es.us.ccia.rcc.RCC;
17
18
19 public abstract class Cambio implements I_Cambio {
20
21         private FCAElement ClaseA;
22         private FCAElement ClaseB;
23         private int    tipoviejo;
24         private int    tiponuevo;
25         private FCAElement Activo;
26
27
28
29         public Cambio(FCAElement _a, FCAElement _b,
30                       int _tipoV, int _tipoN,
31                       FCAElement _act) {
32                 this.ClaseA = _a;
33                 this.ClaseB = _b;
34                 this.tipoviejo = _tipoV;
35                 this.tiponuevo = _tipoN;
36                 this.Activo = _act;
37         }
38
39
40
41         /* (non−Javadoc)
42          * @see es.us.ccia.paella.historial.I_Cambios#getClaseA()
43          */
44         public FCAElement getClaseA() {
45                 return ClaseA;
46         }
47
48
49         /* (non−Javadoc)
50          * @see es.us.ccia.paella.historial.I_Cambios#getClaseB()
51          */
52         public FCAElement getClaseB() {
53                 return ClaseB;
54         }
55
56
57
58         /* (non−Javadoc)
59          * @see es.us.ccia.paella.historial.I_Cambios#getTipoviejo()
60          */
61         public int getTipoviejo() {
62                 return tipoviejo;
63         }
64
65
66
67         /* (non−Javadoc)
68          * @see es.us.ccia.paella.historial.I_Cambios#getTiponuevo()
69          */
70         public int getTiponuevo() {
71                 return tiponuevo;
72         }
73
74         /* (non−Javadoc)
75          * @see es.us.ccia.paella.historial.I_Cambios#getTiponuevo()
76          */
77         public FCAElement getActivo() {
78                 return this.Activo;
79         }
80
81
82         @Override
```

```
83          public String toString() {
84                  String devolver  = new String(
85                                  "(" + getClaseA().getNombre() + "," + getClaseB().getNombre()
    + ") " +
86                                  RCC.toString(tipoviejo) +  "−−>" +
87                                  RCC.toString(tiponuevo));
88                  devolver += " *" + Activo.getNombre() + "*";
89                  return devolver;
90          }
91
92
93          /* (non−Javadoc)
94           * @see es.us.ccia.paella.historial.I_Cambios#ejecutar()
95           */
96          public abstract void ejecutar();
97
98
99
100
101
102
103
104
105
106 }
```

```
1  package es.us.ccia.dummy.historial;
2
3  import java.util.*;
4
5  import es.us.ccia.dummy.contexto.elementos.FCAElement;
6  import es.us.ccia.rcc.RCC;
7  import es.us.tad.Salida;
8
9
10
11 @SuppressWarnings("serial")
12 public class ListaCambios extends ArrayList<Cambio> {
13
14
15
16
17
18
19         public void ejecutar() {
20                 Salida.Imprimirln("Aqui se ejecutan los cambios...");
21
22                 Iterator<Cambio> i=iterator();
23
24                 while (i.hasNext()) {
25
26                         Cambio c = i.next();
27                         c.ejecutar();
28
29
30                 }
31
32         }
33
34 //      IMPORTANTIIIiiiiSSIMMAAA!!!!
35
36
37         public ListaCambios getCambiosRelaciones(FCAElement _activo) {
38
39                 ListaCambios lc = new ListaCambios();
40
41 //              for (int i=0;i< LClases.size();i++) {
42 //                      for (int j=i+1;j< LClases.size();j++) {
43 //
44 //                              FCAElement a = (FCAElement)LClases.get(i);
45 //                              FCAElement b = (FCAElement)LClases.get(j);
46 //
47 //                              int ranter = this.getRelacion(a,b);
48 //
49 //                              //<<<-------------- ESTO ESTA MAL !!
50 //                              int rnueva = this.getRelacion(a,b);
51 //
52 //                              if (ranter != rnueva) {
53 //                                      Cambio cc = H_Cambios8.getCambio(a, b, ranter, rnueva,
   _activo);
54 //                                      lc.add(cc);
55 //                              }
56 //
57 //                      }
58 //              }
59 //              LCambios.add(lc);
60
61                 return lc;
62         }
63
64         public int getRelacion(FCAElement a, FCAElement b){
65                 int devolver = RCC.NODEF;
66 //
67 //              Iterator it = PilaRelacRCC.peek().iterator();
68 //              while (it.hasNext()) {
69 //                      Relacion r = (Relacion)it.next();
70 //                      if (r.getA().equalsIgnoreCase(a.getNombre()) &&
71 //                                      (r.getB().equalsIgnoreCase(b.getNombre()))) {
72 //                              devolver = r.getTipo();
73 //                      } else if (r.getA().equalsIgnoreCase(b.getNombre()) &&
74 //                                      (r.getB().equalsIgnoreCase(a.getNombre()))) {
75 //                              devolver = RCC.reflexiva(r.getTipo());
76 //                      }
77 //              }
78                 return devolver;
79         }
80
81
```

```
82
83  }
```

```
1  package es.us.ccia.dummy.historial;
2
3  import es.us.ccia.dummy.contexto.elementos.FCAElement;
4
5  public interface I_Cambio {
6
7          public FCAElement getClaseA();
8
9          public FCAElement getClaseB();
10
11         public int getTipoviejo();
12
13         public int getTiponuevo();
14
15         /**
16          * @see java.lang.Object#toString()
17          */
18         public String toString();
19
20         public void ejecutar();
21
22 }
```

```
1  package es.us.tad;
2
3  /**
4   * @author Gonzalo A. Aranda Corral
5   *
6   */
7  public class Salida {
8
9          private static Salida salida;
10
11         public static void inicializar() {
12                 salida = new Salida(true);
13         }
14
15         public static void inicializar(boolean activo) {
16                 salida = new Salida(activo);
17         }
18
19         public static void activar(boolean activo) {
20                 salida.activacion(activo);
21         }
22
23
24 ////////////////////////////////////////////////
25
26         boolean mostrar = true;
27
28         private void activacion(boolean mostra) {
29                 mostrar = mostra;
30         }
31
32         private Salida() {
33         }
34
35         private Salida(boolean mostra) {
36                 this.mostrar = mostra;
37                 System.out.print("Interfaz con el usuario....");
38                 System.out.println("ok");
39         }
40
41
42         public static boolean Imprimir(String cadena) {
43                 if (salida.mostrar) System.out.print(cadena);
44                 return true;
45         }
46
47         public static boolean Imprimirln(String cadena) {
48                 if (salida.mostrar) System.out.println(cadena);
49                 return true;
50         }
51
52 }
```

```
1  /*
2   * Created on 17-ene-2005
3   *
4   * TODO To change the template for this generated file go to
5   * Window - Preferences - Java - Code Style - Code Templates
6   */
7  package es.us.tad;
8
9  import java.io.*;
10
11 /**
12  * @author garanda
13  *
14  */
15 public class Ficheros {
16
17         private String nombre;
18
19         public Ficheros(String nombreFichero) {
20                 this.nombre = nombreFichero;
21         }
22
23         public boolean existe(){
24                 boolean devolver = false;
25                 try {
26                 File f = new File(this.nombre);
27                 devolver = f.exists();
28                 } catch(Exception ee) {
29                         Salida.Imprimirln("Error: Algun problema con el fichero;");
30                 }
31                 return devolver;
32         }
33
34         public boolean sepuedeleer(){
35                 boolean devolver = false;
36                 try {
37                 File f = new File(this.nombre);
38                 devolver = f.canRead();
39                 } catch(Exception ee) {
40                         Salida.Imprimirln("Error: Algun problema con el fichero;");
41                 }
42                 return devolver;
43         }
44
45         public String LeerFichero() {
46
47                 StringBuffer devolver= new StringBuffer();;
48
49                 //              boolean fin = false;
50                 int c;
51                 try {
52                 File f = new File( this.nombre);
53                 if( f.exists() ) {
54                         FileInputStream miFicheroSt;
55                     miFicheroSt = new FileInputStream(f);
56                     while (!((c =  miFicheroSt.read())<0)) {
57                         devolver.append((char)c);
58                     }
59                 } else
60                         Salida.Imprimirln( "Error: El fichero no existe." );
61         } catch(Exception EE)  {
62                 devolver = null;
63                 Salida.Imprimirln( "Error: Ha habido algun problema en el fichero");
64         }
65                 return devolver.toString();
66         }
67
68     public void InfoFichero() {
69         try {
70                 File f = new File(this.nombre);
71                 Salida.Imprimirln( "Nombre: "+f.getName() );
72                 Salida.Imprimirln( "Camino: "+f.getPath() );
73                 if( f.exists() ) {
74                         Salida.Imprimir( "Fichero existente " );
75                         Salida.Imprimir( (f.canRead() ? " y se puede Leer" : "" ) );
76                         Salida.Imprimir( (f.canWrite() ? " y se puese Escribir" : "" ) );
77                         Salida.Imprimirln( "." );
78                 } else
79                         Salida.Imprimirln( "El fichero no existe." );
80         } catch(Exception EE)  {
81                 Salida.Imprimirln("Ha habido algun problema en el fichero");
82         }
```

```
83          }
84
85
86      public void Grabar(String cont) {
87
88          File textFile = new File(this.nombre);
89          FileWriter textOut;
90          try {
91                  textOut = new FileWriter(textFile);
92                  textOut.write(cont);
93                  textOut.close();
94          } catch (IOException e) {
95                  e.printStackTrace();
96          }
97
98      }
99
100     public String ElegirFichero() {
101         String devolver = null;
102
103
104
105
106
107         return devolver;
108     }
109
110 }
```