```
1  /**
2   *
3   */
4  package es.us.ccia.dummy.teoria.regiones;
5
6  /**
7   * @author USUARIO
8   *
9   */
10 public interface I_Elemento {
11
12         public static final int REGION = 1;
13         public static final int PUNTO  = 2;
14
15 }
```

```
1  /**
2   *
3   */
4  package es.us.ccia.dummy.teoria.regiones;
5
6  import java.awt.Color;
7  import java.awt.Graphics;
8
9  /**
10  * @author USUARIO
11  *
12  */
13 public class Region extends Elemento {
14
15
16         int X2;
17         int Y2;
18
19         /**
20          * @param nombre
21          * @param x1
22          * @param y1
23          * @param x2
24          * @param y2
25          */
26         public Region(Object nombre, int x1, int y1, int x2, int y2) {
27                 super(REGION,nombre, x1, y1);
28                 this.X2 = x2;
29                 this.Y2 = y2;
30         }
31
32         public int getX2(){
33                 return this.X2;
34         }
35
36         public int getY2(){
37                 return this.Y2;
38         }
39
40         @Override
41         public boolean pertenece(int X, int Y) {
42                 //return  (X1 < X) && (X < X2) &&(Y1 < Y) && (Y < Y2);
43                 return
44                    (
45                                 ((X1−7 < X) && (X < X1+7) &&(Y1 < Y) && (Y < Y2)) ||
46                                 ((X2−7 < X) && (X < X2+7) &&(Y1 < Y) && (Y < Y2)) ||
47                                 ((Y1−7 < Y) && (Y < Y1+7) &&(X1 < X) && (X < X2)) ||
48                                 ((Y2−7 < Y) && (Y < Y2+7) &&(X1 < X) && (X < X2))
49                    );
50
51
52
53
54         }
55
56
57         public int getAncho() {
58                 return X2 − X1;
59         }
60
61         public int getAlto() {
62                 return Y2 − Y1;
63         }
64
65         @Override
66         public void posicionar(int X, int Y) {
67                 int Ancho = X2−X1;
68                 int Alto  = Y2−Y1;
69                 X1 = X;
70                 Y1 = Y;
71                 X2 = X1 + Ancho;
72                 Y2 = Y1 + Alto;
73         }
74
75         public void actualizar(int Ancho, int Alto) {
76                 this.X2 = X1 + Ancho;
77                 this.Y2 = Y1 + Alto;
78         }
79
80         @Override
81         public void pintar(Graphics g) {
82                 if (super.estaSeleccionado()) {
```

```
83                          g.setColor(Color.BLUE);
84                          g.drawRect(X1+1, Y1+1, X2−X1−2, Y2−Y1−2);
85                          g.drawRect(X1,   Y1,   X2−X1,   Y2−Y1);
86                          g.drawRect(X1−1, Y1−1, X2−X1+2, Y2−Y1+2);
87                  } else {
88                          g.drawRect(X1, Y1, X2−X1, Y2−Y1);
89                          g.setColor(Color.BLACK);
90                  }
91                  g.drawString(Nombre.toString(), X1, Y1−2);
92          }
93
94          @Override
95          public boolean esFrontera(int X, int Y) {
96                  boolean devolver = false;
97                  if ((((Y > Y1−7) && (Y < Y1+7) ) || ((Y > Y2−7) && (Y < Y2+7)))
98                                  && (X>=X1) && (X<=X2)) {
99                          devolver = true;
100                 } else if ((    ((X > X1−7) && (X < X1+7)) || ((X > X2−7) && (X < X2+7) ))
101                                 && (Y>=Y1) && (Y<=Y2)) {
102                         devolver = true;
103                 }
104                 return devolver;
105         }
106
107         public boolean contiene(int X, int Y) {
108                 return  (X1 < X) && (X < X2) &&(Y1 < Y) && (Y < Y2);
109         }
110
111         public boolean contiene(int X, int Y, int XX, int YY) {
112                 return (X1 < X) && (XX < X2) && (Y1 < Y) && (YY < Y2);
113         }
114
115         public boolean contiene(Punto p) {
116                 return
117                      (X1 < p.getX1()) && (p.getX1() < X2) &&
118                      (Y1 < p.getY1()) && (p.getY1() < Y2);
119         }
120
121         public boolean contiene(Region r) {
122                 return
123                      (X1 < r.getX1()) && (r.getX2() < X2) &&
124                      (Y1 < r.getY1()) && (r.getY2() < Y2);
125         }
126
127         /* (non−Javadoc)
128          * @see java.lang.Object#toString()
129          */
130         @Override
131         public String toString() {
132                 return "Region:" + super.getNombre() +
133                           "(" + super.X1 + "," + super.Y1 + "," + this.X2 + "," + this.Y2 + ")";
134         }
135
136
137
138
139
140 }
141
```

```
/**************************************
 * Nombre: ListaElementos.java
 * Autor:  Gonzalo A. Aranda Corral
 *
 *
 * Listado de correcciones:
 *
 *   - AÃ±adir recuperar una region por su nombre.
 *
 *
 **************************************/

package es.us.ccia.dummy.teoria.regiones;


import java.util.ArrayList;
import java.util.Collection;
import java.util.Iterator;


//@SuppressWarnings({"unchecked","serial"})
public class ListaElementos implements Iterator,I_Elemento, Cloneable {




        ArrayList L;


        public ListaElementos(){
                this.L = new ArrayList();
        }



        public Region getRegion(String nombre) {
                Region devolver = null;
                boolean encontrado = false;

                Iterator i = L.iterator();
                while(i.hasNext() && !encontrado){
                        Elemento e = (Elemento)i.next();
                        if (e.getTipo()==REGION &&
                                        e.getNombre().equalsIgnoreCase(nombre)) {
                                devolver = (Region)e;
                                encontrado = true;
                        }
                }

                return devolver;
        }


        public Punto getIndividuo(String nombre) {
                Punto devolver = null;
                boolean encontrado = false;

                Iterator i = L.iterator();
                while(i.hasNext() && !encontrado){
                        Elemento e = (Elemento)i.next();
                        if (e.getTipo()==PUNTO &&
                                        e.getNombre().equalsIgnoreCase(nombre)) {
                                devolver = (Punto)e;
                                encontrado = true;
                        }
                }

                return devolver;
        }


        @Override
        public ListaElementos clone() {

                ListaElementos devolver = new ListaElementos();

                Iterator i = L.iterator();
                while (i.hasNext()) {
                        Elemento e = (Elemento)i.next();
                        devolver.add(e);
                }

```

```
83
84                  return devolver;
85          }
86
87
88
89
90
91          @Override
92          public String toString() {
93                  String devolver = "";
94                  Iterator i = this.L.iterator();
95                  while (i.hasNext()) {
96                          Elemento aux= (Elemento)i.next();
97                          devolver += aux.toString() + "−−";
98                  }
99                  return devolver;
100         }
101
102
103
104         public Iterator iterator() {
105                 return this.L.iterator();
106         }
107
108         public boolean hasNext() {
109                 // TODO Auto−generated method stub
110                 return false;
111         }
112
113
114         public Object next() {
115                 // TODO Auto−generated method stub
116                 return null;
117         }
118
119
120         public void remove() {
121                 // TODO Auto−generated method stub
122
123         }
124
125         public int size() {
126                 return this.L.size();
127         }
128
129         public void add(Elemento r) {
130                 this.L.add(r);
131         }
132
133         public void clear() {
134                 this.L.clear();
135         }
136
137         public void remove(Elemento r) {
138                 this.L.remove(r);
139         }
140
141         public void remove(Collection c) {
142                 this.L.remove(c);
143         }
144
145         public Elemento get(int i) {
146                 return (Elemento)this.L.get(i);
147         }
148
149
150
151
152
153
154         public Collection obtenerMarcadas() {
155                 Collection devolver = new ArrayList();
156
157                 Iterator i = this.L.iterator();
158                 while (i.hasNext()) {
159                         Elemento r = (Elemento)i.next();
160                         if (r.estaSeleccionado())
161                                 devolver.add(r);
162                 }
163
164
```

```
165                     return devolver;
166             }
167
168             public ListaElementos getListaTipo(int _tipo) {
169                     ListaElementos devolver = new ListaElementos();
170
171                     Iterator i1 = this.L.iterator();
172                     while (i1.hasNext()) {
173                             Elemento r1 = (Elemento)i1.next();
174                             if (r1.getTipo() == _tipo) {
175                                     devolver.add(r1);
176                             }
177                     }
178
179                     return devolver;
180             }
181
182             public void marcarElemento(String c1) {
183                     boolean fin = false;
184
185                     Iterator i1 = this.L.iterator();
186                     while ((i1.hasNext()) && !fin) {
187                             Elemento r1 = (Elemento)i1.next();
188
189                             if (r1.getNombre().equalsIgnoreCase(c1)) {
190                                     r1.seleccionar(true);
191                                     fin = true;
192                             }
193                     }
194             }
195
196             public void desmarcarElementos() {
197                     Iterator i1 = this.L.iterator();
198                     while (i1.hasNext()) {
199                             Elemento r1 = (Elemento)i1.next();
200                             r1.seleccionar(false);
201                     }
202             }
203
204
205
206     }
```

```java
/**
 *
 */
package es.us.ccia.dummy.teoria.regiones;

import java.awt.Graphics;

/**
 * @author USUARIO
 *
 */
public class Punto extends Elemento {

	/**
	 * @param n
	 * @param x
	 * @param y
	 */
	public Punto(Object n, int x, int y) {
		super(PUNTO,n, x, y);

	}

	@Override
	public boolean pertenece(int X, int Y) {
		return  (X1−3 < X) && (X < X1+3) &&(Y1−3 < Y) && (Y < Y1+3);
	}

	@Override
	public void posicionar(int X, int Y) {
		this.X1 = X;
		this.Y1 = Y;
	}

	/* (non−Javadoc)
	 * @see es.us.ccia.paella.teoria.elementos.Elemento#pintar(java.awt.Graphics)
	 */
	@Override
	public void pintar(Graphics g) {
		g.drawOval(X1, Y1, 3, 3);
		g.drawString(Nombre.toString(), X1+4, Y1);
	}

	@Override
	public boolean esFrontera(int X, int Y) {
		return pertenece(X,Y);
	}


	/* (non−Javadoc)
	 * @see java.lang.Object#toString()
	 */
	@Override
	public String toString() {
		return "Punto: " + super.getNombre();
	}


}
```

```
1  /**
2   *
3   */
4  package es.us.ccia.dummy.teoria.regiones;
5
6  import java.awt.Color;
7  import java.awt.Graphics;
8
9
10 /**
11  * @author USUARIO
12  *
13  */
14 public abstract class Elemento implements I_Elemento {
15
16         protected Object Nombre;
17         protected int X1;
18         protected int Y1;
19         protected int tipo;
20         protected Color color;
21         boolean seleccionado;
22
23         public Elemento(int Tipo, Object n,int x, int y) {
24                 this.Nombre = n;
25                 this.X1     = x;
26                 this.Y1     = y;
27                 this.tipo = Tipo;
28                 seleccionado = false;
29         }
30
31         public abstract boolean pertenece (int X, int Y);
32         public abstract boolean esFrontera(int X, int Y);
33         public abstract void    posicionar(int X, int Y);
34         public abstract void    pintar    (Graphics g);
35
36         public int getTipo(){
37                 return this.tipo;
38         }
39
40         public Object getArgumento() {
41                 return this.Nombre;
42         }
43
44         public String getNombre() {
45                 if (Nombre == null) {
46                         System.out.println("el nombre es nulo");
47                 }
48 //              else
49 //                      System.out.println(Nombre.getClass().getName());
50
51
52                 return this.Nombre.toString();
53         }
54         public int getX1(){
55                 return this.X1;
56         }
57
58         public int getY1(){
59                 return this.Y1;
60         }
61
62         public void setColor(Color c){
63                 this.color = c;
64         }
65
66         public boolean estaSeleccionado() {
67                 return seleccionado;
68         }
69
70         public void seleccionar(boolean seleccionado) {
71                 this.seleccionado = seleccionado;
72         }
73
74
75
76
77 }
```

```
1  package es.us.ccia.dummy.teoria.movimientos;
2
3  import es.us.ccia.dummy.teoria.regiones.*;
4
5  public class Movimiento {
6
7          Elemento      Origen;
8          int            TipoMovimiento;
9          ListaElementos Resultado;
10
11         public Movimiento(Elemento _origen, int _tipo, ListaElementos _res) {
12                 Origen         = _origen;
13                 TipoMovimiento = _tipo;
14                 Resultado      = _res;
15         }
16
17         public ListaElementos getListaRegiones() {
18                 return Resultado;
19         }
20
21
22 }
```

```
1  package es.us.ccia.dummy.teoria.movimientos;
2
3  public class I_Mov {
4
5          public static final int NOMOV = 0;
6
7          public static final int REDIM = 1;
8          public static final int DESPL = 2;
9
10
11
12 }
```

```
1  package es.us.ccia.dummy.dibujo.eventos;
2
3  /**
4   * Version 0.9:
5   *    20080713
6   *    − Clase que realiza todos los movimientos sobre
7   *      el dibujo.
8   *
9   *================================================
10  * TODO: Revisar por que se pone uno en azul que no
11  *       corresponde.
12  *     − Hay un fallo, cuando se selecciona el primero
13  *       de la lista de relaciones... luego no funciona
14  *       el movimiento sobre las regiones
15  *
16  */
17
18 import java.awt.event.MouseEvent;
19 import java.awt.event.MouseListener;
20 import java.awt.event.MouseMotionListener;
21 import java.util.HashSet;
22 import java.util.Iterator;
23 import java.util.Set;
24
25 import es.us.ccia.dummy.dibujo.ventanas.VentanaDibujo;
26 import es.us.ccia.dummy.dibujo.GUI;
27 import es.us.ccia.dummy.teoria.regiones.*;
28
29
30 /***
31  *
32  * @author  Gonzalo A. Aranda Corral
33  * @version 0.9.0
34  * @since   13/07/2008
35  *
36  */
37 @SuppressWarnings({"unchecked"})
38 public class Movimientos implements MouseListener, MouseMotionListener, I_Elemento {
39         int BotonActivo;  // Boton pulsado
40         int mposX;
41         int mposY;
42
43         int mancho;
44         int malto;
45         boolean    marcado;
46
47         boolean    modificado;
48
49         Elemento   activo;      // Elemento activo o pinchado
50         Elemento   iluminado;
51
52         ListaElementos historial;
53         ListaElementos ListaE;
54
55         GUI gui = null;
56
57         /**
58          * Constructor de la clase
59          * @param g Enlace con el entorno grafico
60          */
61         public Movimientos(GUI g, ListaElementos _le) {
62                 // INICIALIZACION
63                 activo = null;
64                 iluminado = null;
65
66                 BotonActivo = −1;
67                 mposX = −1;
68                 mposY = −1;
69
70                 marcado = false;
71
72                 modificado = false;
73
74                 ListaE = _le;
75
76                 gui = g;
77
78         }
79
80         public void actualizar(ListaElementos _le) {
81                 ListaE = _le;
82                 System.out.println(ListaE.toString());
```

```
83          }
84
85
86
87
88          /***
89           * Procedimiento donde se activa una region
90           */
91          public void mousePressed(MouseEvent arg0) {
92
93                  if (ListaE != null) {
94
95                          BotonActivo = arg0.getButton();
96                          mposX = arg0.getX();
97                          mposY = arg0.getY();
98
99                  Iterator e = ListaE.iterator();
100                 while (e.hasNext()) {
101                         Elemento rr = (Elemento)e.next();
102                         if (rr.pertenece(arg0.getX(), arg0.getY())) {
103                                 activo = rr;
104                         }
105                 }
106
107                 } else {
108                         System.out.println("La lista es nula");
109                 }
110
111         }
112
113
114         /***
115          * Procedimiento para SOLTAR una region
116          */
117         public void mouseReleased(MouseEvent arg0) {
118
119                 if (activo != null) {
120                         modificado = true;
121
122                         if (BotonActivo == MouseEvent.BUTTON1) {
123                                 int coordx = activo.getX1();
124                                 int coordy = activo.getY1();
125
126                                 // AJUSTE A LA CUADRICULA
127                                 if (coordx % 10 > 5 ) {
128                                         coordx = coordx + 10 − coordx % 10;
129                                 } else {
130                                         coordx = coordx − coordx % 10;
131                                 }
132                                 if (coordy % 10 > 5 ) {
133                                         coordy = coordy + 10 − coordy % 10;
134                                 } else {
135                                         coordy = coordy − coordy % 10;
136                                 }
137                                 activo.posicionar(coordx, coordy);
138
139                         } else if (BotonActivo == MouseEvent.BUTTON3) {
140
141                                 if (activo.getTipo() == REGION) {
142
143                                         Region rr = (Region)activo;
144                                         int ancho = rr.getAncho();
145                                         int alto  = rr.getAlto();
146
147                                         //  AJUSTE A LA CUADRICULA
148                                         int sobra = ancho % 10;
149                                         if (sobra > 5) {
150                                                 ancho = ancho − sobra + 10;
151                                         } else {
152                                                 ancho = ancho − sobra;
153                                         }
154                                         sobra = alto % 10;
155                                         if (sobra > 5) {
156                                                 alto = alto − sobra + 10;
157                                         } else {
158                                                 alto = alto − sobra;
159                                         }
160                                         rr.actualizar(ancho, alto);
161
162                                 }
163                         }
164                 }
```

```
165
166
167                 // ESto es como idea para el undo!!
168
169 //              Movimiento m = new Movimiento(activo.getArgumento(),
170 //              I_Mov.REDIM,((ListaElementos)teoria.getListaRegiones()).clone());
171 //teoria.doMovimiento(m);
172
173
174
175                 arg0.getComponent().repaint();
176                 activo = null;
177                 BotonActivo = −1;
178                 mposX = −1;   // Posicion del raton al arrastrar
179                 mposY = −1;
180
181                 gui.cambioVentanaDibujo(ListaE);
182
183         }
184
185         /***
186          * Procedimiento para ARRASTRAR una region
187          */
188         public void mouseDragged(MouseEvent arg0) {
189
190                 if (activo != null) {
191                         int nuevaX = arg0.getX();
192                         int nuevaY = arg0.getY();
193
194                         if (BotonActivo == MouseEvent.BUTTON1) {
195                                 int cx = activo.getX1() + (nuevaX−mposX);
196                                 int cy = activo.getY1() + (nuevaY−mposY);
197
198                                 if (cx<0) cx = 0;  if (cy<0) cy = 0;
199                                 if (cx>VentanaDibujo.TAMANOX) cx = VentanaDibujo.TAMANOX;
200                                 if (cy>VentanaDibujo.TAMANOY) cy = VentanaDibujo.TAMANOY;
201
202                                 activo.posicionar( cx , cy );
203
204                         } else if (BotonActivo == MouseEvent.BUTTON3) {
205                                 if (activo.getTipo() == REGION) {
206                                         Region rr = (Region)activo;
207                                         int cx = rr.getAncho() + (nuevaX−mposX);
208                                         int cy = rr.getAlto() + (nuevaY−mposY);
209                                         if (cx<30) cx = 30;  if (cy<30) cy = 30;
210                                         rr.actualizar(cx, cy);
211                                 }
212                         }
213                         mposX = nuevaX;
214                         mposY = nuevaY;
215
216                 }
217
218                 // Aqui se puede hacer lo del UNDO
219 //              gui.getAreaDibujo().repaint();
220                 gui.repintarDibujo();
221
222
223         }
224
225
226
227
228         /***
229          * Procedimiento para RESALTAR una region cuando paso por encima
230          */
231         public void mouseMoved(MouseEvent arg0) {
232
233                 // Aqui guardarÃ© las regiones que voy acumulando
234                 Set listaRegiones = new HashSet();
235
236                 if (ListaE != null) {
237                 Iterator e = ListaE.iterator();
238                 while (e.hasNext()) {
239                         Elemento rr = (Elemento)e.next();
240                         if (rr.esFrontera(arg0.getX(), arg0.getY())) {
241                                 rr.seleccionar(true);
242                                 listaRegiones.add(rr.getNombre());
243                         } else {
244                                 rr.seleccionar(false);
245                         }
246                 }
```

```
247                     }
248                     arg0.getComponent().repaint();
249 //
250 //              //=====================================================
251 //              // *   Actualizacion en negrita de las relaciones
252 //                  //     seleccionadas, dentro de la ventana
253 //              //=====================================================
254 //              Iterator e = teoria.getListaRelacionesRCC().iterator();
255 //              while (e.hasNext()) {
256 //                  Relacion rr = (Relacion)e.next();
257 //                  if (listaRegiones.contains(rr.getA()) ||  listaRegiones.contains(rr.getB()) ){
258 //                          rr.seleccionar(true);
259 //                  } else {
260 //                          rr.seleccionar(false);
261 //                  }
262 //              }
263 //              g.getVentanaRelaciones().actualizar();
264 //                  //gui.getVHistorial().actualizar(teoria.getListaRelacionesRCC8());
265 //
266             }
267
268
269             public void mouseClicked(MouseEvent arg0) {
270             }
271             public void mouseEntered(MouseEvent arg0) {
272             }
273             public void mouseExited(MouseEvent arg0) {
274             }
275
276
277             public boolean estaModificado() {
278                     return this.modificado;
279             }
280
281             public void setModificado(boolean m) {
282                     this.modificado = m;
283             }
284
285
286 }
```

```
1  package es.us.ccia.dummy.dibujo.eventos;
2
3  import java.util.ArrayList;
4
5  import javax.swing.JList;
6  import javax.swing.event.ListSelectionEvent;
7  import javax.swing.event.ListSelectionListener;
8
9  import es.us.ccia.dummy.dibujo.GUI;
10 import es.us.ccia.dummy.dibujo.ventanas.ElementoLista;
11
12 @SuppressWarnings("unchecked")
13 public class EventosClases implements ListSelectionListener {
14
15         GUI g;
16
17
18         public EventosClases(GUI _g) {
19                 super();
20                 g = _g;
21         }
22
23
24         public void valueChanged(ListSelectionEvent arg0) {
25                 ArrayList devolver = new ArrayList();
26                 if (!arg0.getValueIsAdjusting()) {
27                         JList pp = (JList)arg0.getSource();
28                         Object[] dd = pp.getSelectedValues();
29                         for(int i=0;i<dd.length;i++) {
30                                 ElementoLista el = (ElementoLista)dd[i];
31                                 devolver.add(el.getNombre());
32                         }
33                         g.cambioListaClases(devolver);
34                 }
35         }
36
37 }
```

```
1  package es.us.ccia.dummy.dibujo.eventos;
2
3  import java.util.ArrayList;
4
5  import javax.swing.JList;
6  import javax.swing.event.ListSelectionEvent;
7  import javax.swing.event.ListSelectionListener;
8
9  import es.us.ccia.dummy.dibujo.GUI;
10 import es.us.ccia.dummy.dibujo.ventanas.ElementoLista;
11
12 @SuppressWarnings("unchecked")
13 public class EventosObjetos implements ListSelectionListener {
14
15         GUI g;
16
17
18         public EventosObjetos(GUI _g) {
19                 super();
20                 g = _g;
21         }
22
23
24         public void valueChanged(ListSelectionEvent arg0) {
25                 ArrayList devolver = new ArrayList();
26                 if (!arg0.getValueIsAdjusting()) {
27                         JList pp = (JList)arg0.getSource();
28                         Object[] dd = pp.getSelectedValues();
29                         for(int i=0;i<dd.length;i++) {
30                                 ElementoLista el = (ElementoLista)dd[i];
31                                 devolver.add(el.getNombre());
32                         }
33                         g.cambioListaIndividuos(devolver);
34                 }
35         }
36
37 }
```

```
1  package es.us.ccia.dummy.dibujo;
2
3  import java.awt.*;
4  import java.awt.event.*;
5  import java.util.ArrayList;
6  import java.util.Collection;
7  import java.util.LinkedList;
8
9  import javax.swing.*;
10 import javax.swing.event.* ;
11
12 import es.us.ccia.dummy.dibujo.ventanas.*;
13 import es.us.ccia.dummy.teoria.regiones.ListaElementos;
14
15
16
17 @SuppressWarnings({"serial","unchecked"})
18 public abstract class GUI extends JFrame
19         implements SwingConstants
20                 //, ChangeListener
21                 //, ListSelectionListener
22
23                 {
24
25         private Container cliente;
26
27         private JSplitPane Izquierdo;
28         private VentanaEtiquetas vetiq;
29         private VentanaObjetos   vindiv;
30
31         private JPanel Derecho;
32         private VentanaBoton      vboton;
33         private VentanaHistorial  vhist;
34         private VentanaRelaciones vrel;
35         private VentanaDibujo vdibujo;
36         private BarraIconos biconos;
37         private BarraMenu barraMenu;
38
39         ActionListener al  = null;
40
41         public GUI() {
42                 super( "* DummyPaella *" );
43
44                 // Se habilitan los eventos
45                 enableEvents( AWTEvent.WINDOW_EVENT_MASK );
46
47         al = new ActionListener(){
48                 public void actionPerformed(ActionEvent e) {
49                         String clase = e.getSource().getClass().getSimpleName();
50                         String comando = e.getActionCommand();
51
52                         System.out.println("he recibido evento");
53                         System.out.println(clase + " − " + comando);
54
55
56                         if (clase.equalsIgnoreCase("JMenuItem")) {
57                                 if (comando.equalsIgnoreCase("Abrir")) {
58                                         menuAbrir();
59                                 } else if (comando.equalsIgnoreCase("Salir")) {
60                                         menuSalir();
61                                 } else if (comando.equalsIgnoreCase("Guardar")) {
62
63                                 } else if (comando.equalsIgnoreCase("Guardar Como")) {
64                                         menuGuardarComo();
65
66                                 } else if (comando.equalsIgnoreCase("Importar Delic")) {
67                                         menuImportarDel();
68
69                                 } else if (comando.equalsIgnoreCase("Exportar Delic")) {
70                                         menuExportarDel();
71
72                                 } else if (comando.equalsIgnoreCase("Nuevo")) {
73                                         menuNuevo();
74
75                                 } else if (comando.equalsIgnoreCase("Acerca de")) {
76                                         menuAcercade();
77
78                                 } else if (comando.equalsIgnoreCase("Deshacer")) {
79                                         menuDeshacer();
80
81                                 } else {
82                                         System.out.println("Opcion del menu no implementada");
```

```
                                }

                        } else if (clase.equalsIgnoreCase("JButton")) {
                                if (comando.equalsIgnoreCase("S A V E !")) {
                                        botonEjecutarCambios(vdibujo.getListaElementos());
                                }
                        } else  {

                                System.out.println(" no implementado ");



                        }



                }

        };

        }


    public void init() {


        // Barra de MENU
        barraMenu  = new BarraMenu(al);
        setJMenuBar( barraMenu );

        // Barra de Iconos
        biconos = new BarraIconos();

        //  Ventana de la Izquierda    //
        vetiq = new VentanaEtiquetas(this);
        vindiv = new VentanaObjetos(this);

        Izquierdo = new JSplitPane(JSplitPane.VERTICAL_SPLIT);
        Izquierdo.add(vetiq);
        Izquierdo.add(vindiv);
        Izquierdo.setPreferredSize(new Dimension(250,800));

        //  Ventanas de la Derecha
        vboton = new VentanaBoton(al);
        vrel   = new VentanaRelaciones(this);
        vhist  = new VentanaHistorial();
        Derecho = new JPanel();
        Derecho.setLayout( new GridLayout(3,1) );
        Derecho.add(vboton);
        Derecho.add(vrel);
        Derecho.add(vhist);


        //      Panel Central
        vdibujo = new VentanaDibujo(this);


        // COMPOSICION PANTALLA
        cliente = this.getContentPane();
        cliente.setLayout( new BorderLayout() );
        cliente.add( biconos,BorderLayout.NORTH);
        cliente.add( Izquierdo,BorderLayout.LINE_START);
        cliente.add( vdibujo, BorderLayout.CENTER );
        cliente.add( Derecho,BorderLayout.LINE_END );

    }


    public void stateChanged( ChangeEvent ce ) {
    }


    public void processEvent( AWTEvent evt ) {
        if( evt.getID() == WindowEvent.WINDOW_CLOSING ) {
                System.exit( 0 );
        } else {
                super.processEvent( evt );
        }
    }

```

```
165     public void reiniciarContexto(Collection etiq,Collection ind){
166         vetiq.reiniciar(etiq);
167         vindiv.reiniciar(ind);
168 
169         //        vhist.actualizar();
170         vrel.actualizar(null);
171         //              vdibujo.repaint();
172     }
173 
174 
175 
176 
177 
178 
179 
180 
181 //
182 //
183 //    public void valueChanged(ListSelectionEvent arg0) {
184 //
185 //      System.out.println("aqui vienen las listas " + arg0.getSource());
186 //
187 //    }
188 
189 
190     public abstract void menuSalir();
191     public abstract void menuNuevo();
192     public abstract void menuAbrir();
193     public abstract void menuDeshacer();
194     public abstract void menuImportarDel();
195     public abstract void menuExportarDel();
196     public abstract void menuGuardarComo();
197 
198     public abstract void botonEjecutarCambios(ListaElementos _nuevaconfig);
199 
200         public abstract void cambioListaClases(ArrayList clases);
201     public abstract void cambioListaIndividuos(ArrayList indivs);
202     public abstract void cambioListaRelaciones();
203 
204     public abstract void cambioVentanaDibujo(ListaElementos _lista);
205 
206 
207 
208 
209 
210 
211 
212     public ArrayList getListaRelaciones(){
213         return this.vrel.getSeleccion();
214     }
215 
216     public void pintarRelacionesRCC(LinkedList lista){
217 
218 
219 
220 
221         this.vrel.actualizar(lista);
222 
223         /***
224          * HAY QUE AÃM−^QADIR:
225          *   − teniendo la lista de relaciones..
226          *         que se calculen las regiones.
227          */
228 
229         ////////////////////////////////////////////
230 
231         /***
232          * Pintar las regiones.
233          */
234 
235 
236 
237 
238 //      this.vdibujo.actualizar(null);
239 
240     }
241 
242     public void pintarVentanaHistorial(LinkedList lista){
243 
244         //  <−−−−−−−− incluir la lista dentro de la llamadas
245 
246         this.vhist.actualizar();
```

```
247         }
248 
249         public void pintarVentanaDibujo(ListaElementos lista) {
250             this.vdibujo.actualizar(lista);
251         }
252 
253         public void menuAcercade() {
254             System.out.println("Acerca de ... Gonzalo A. Aranda Corral");
255         }
256 
257         public void repintarDibujo() {
258             this.vdibujo.getDibujo().repaint();
259         }
260 
261     }
```

```
1  package es.us.ccia.dummy.dibujo;
2
3  import java.awt.event.ActionListener;
4  import javax.swing.*;
5
6  //import es.us.tad.*;
7  //import es.us.ccia.dummy.DummyPaella;
8  //import es.us.ccia.dummy.contexto.Contexto;
9  //import es.us.ccia.dummy.contexto.entsal.Entrada;
10 //import es.us.ccia.dummy.contexto.entsal.Grabar;
11 //import es.us.ccia.dummy.contexto.entsal.Importar;
12 //import es.us.ccia.dummy.contexto.razonador.*;
13 //import es.us.ccia.dummy.teoria.*;
14
15 @SuppressWarnings("serial")
16 public class BarraMenu extends JMenuBar {
17
18         private JMenu menuArchivo;
19         private JMenuItem opcionAbrir;
20         private JMenuItem opcionGuardar;
21         private JMenuItem opcionGComo;
22         private JMenuItem opcionImp;
23         private JMenuItem opcionExp;
24         private JMenuItem opcionSalir;
25         private JMenuItem opcionNuevo;
26
27         private JMenu menuEditar;
28         private JMenuItem Undo;
29
30         private JMenu menuAyuda;
31         private JMenuItem Acercade;
32
33
34         public BarraMenu(ActionListener aL) {
35                 super();
36             menuArchivo = new JMenu( "Archivo" );
37
38             opcionAbrir   = new JMenuItem("Abrir");
39             opcionGuardar = new JMenuItem("Guardar");
40             opcionGComo   = new JMenuItem("Guardar como");
41             opcionImp     = new JMenuItem("Importar Delic");
42             opcionExp     = new JMenuItem("Exportar Delic");
43 //          JMenuItem.
44             opcionSalir   = new JMenuItem("Salir");
45             opcionNuevo   = new JMenuItem("Nuevo");
46
47
48             opcionAbrir.addActionListener( aL );
49             opcionGuardar.addActionListener( aL );
50             opcionGComo.addActionListener( aL );
51             opcionImp.addActionListener( aL );
52             opcionExp.addActionListener( aL );
53             opcionSalir.addActionListener( aL );
54             opcionNuevo.addActionListener( aL );
55
56             menuArchivo.add( opcionNuevo );
57             menuArchivo.add( opcionAbrir );
58             menuArchivo.add( opcionGuardar );
59             menuArchivo.add( opcionGComo );
60             menuArchivo.addSeparator();
61             menuArchivo.add( opcionImp );
62             menuArchivo.add( opcionExp );
63             menuArchivo.addSeparator();
64             menuArchivo.add( opcionSalir );
65             add( menuArchivo);
66
67             /************************/
68
69             menuEditar  = new JMenu( "Editar" );
70             Undo        = new JMenuItem("Deshacer");
71
72             Undo.addActionListener(aL);
73
74             menuEditar.add(Undo);
75
76             add(menuEditar);
77
78
79             /************************/
80             menuAyuda   = new JMenu( "Ayuda" );
81             Acercade    = new JMenuItem("Acerca de");
82
```

```
83                Acercade.addActionListener(aL);
84
85                menuAyuda.add(Acercade);
86                add(menuAyuda);
87
88
89            }
90
91
92
93
94
95
96    }
```

```
1  package es.us.ccia.dummy.dibujo;
2
3  import javax.swing.*;
4  import java.awt.*;
5
6
7  @SuppressWarnings("serial")
8  public class Barralconos extends JComponent {
9
10         /**
11          *
12          */
13         JToolBar toolBar;
14         JButton openButton,saveButton,copyButton,cutButton,pasteButton;
15
16         /**
17          * Constructor de la barra de iconos
18          */
19         public BarraIconos() {
20
21
22                 /*****JToolBar*****/
23                 toolBar = new JToolBar();
24                 //Crea los botones asignando sendas imÃ¡genes y los agrega al toolbar
25                 openButton = new JButton();
26 //              openButton.setIcon(new ImageIcon(getClass().getResource("/abrir.gif")));
27                 openButton.setMargin(new Insets(0, 0, 0, 0));
28                 openButton.setText("Abrir");
29                 toolBar.add(openButton);
30
31                 saveButton = new JButton();
32 //              saveButton.setIcon(new ImageIcon(getClass().getResource("/folder.png")));
33                 saveButton.setMargin(new Insets(0, 0, 0, 0));
34                 toolBar.add(saveButton);
35
36                 //agrega un separador en la toolbar
37                 toolBar.addSeparator();
38
39 //              copyButton = new JButton();
40 //              copyButton.setIcon(new ImageIcon(getClass().getResource("/copy.gif")));
41 //              copyButton.setMargin(new Insets(0, 0, 0, 0));
42 //              toolBar.add(copyButton);
43 //
44 //              cutButton = new javax.swing.JButton();
45 //              cutButton.setIcon(new ImageIcon(getClass().getResource("/cut.gif")));
46 //              cutButton.setMargin(new Insets(0, 0, 0, 0));
47 //              toolBar.add(cutButton);
48 //
49 //              pasteButton = new javax.swing.JButton();
50 //              pasteButton.setIcon(new ImageIcon(getClass().getResource("/paste.gif")));
51 //              pasteButton.setMargin(new Insets(0, 0, 0, 0));
52 //              toolBar.add(pasteButton);
53
54
55                 this.add(toolBar);
56
57
58                 this.setPreferredSize(toolBar.getPreferredSize());
59         }
60
61
62
63
64 }
```

```
1  package es.us.ccia.dummy.dibujo.ventanas.dialogo;
2
3  import java.awt.BorderLayout;
4  import java.awt.Component;
5  import java.awt.Container;
6  import java.awt.Dimension;
7  import java.awt.Frame;
8  import java.awt.event.ActionEvent;
9  import java.awt.event.ActionListener;
10 import java.awt.event.MouseAdapter;
11 import java.awt.event.MouseEvent;
12 import javax.swing.*;
13
14
15 @SuppressWarnings("serial")
16 public class NuevoIndividuo extends JDialog {
17
18         public  static NuevoIndividuo dialog;
19     private static Object value;
20     private JList list;
21         private JTextField NombreC;
22
23     private NuevoIndividuo(Frame frame, Object[] data) {
24         super(frame, "New Individual", true);
25
26
27         JLabel Texto = new JLabel("Individual Name:");
28         NombreC = new JTextField();
29
30
31
32         JPanel Encabezado = new JPanel();
33         Encabezado.setLayout(new BoxLayout(Encabezado,BoxLayout.Y_AXIS));
34         Encabezado.setBorder(BorderFactory.createEmptyBorder(10,10,10,10));
35         Encabezado.add(Texto);
36         Encabezado.add(NombreC);
37
38
39
40         //buttons
41         JButton cancelButton = new JButton("Cancel");
42         cancelButton.addActionListener(new ActionListener() {
43                 public void actionPerformed(ActionEvent e) {
44                         NuevoIndividuo.dialog.setVisible(false);
45                 }
46         });
47
48
49         final JButton setButton = new JButton("Accept");
50 //      setButton.addActionListener(new ActionListener() {
51 //              public void actionPerformed(ActionEvent e) {
52 //
53 //                      boolean salir = false;
54 //                              String nombre = NombreC.getText().trim();
55 //
56 //                              if (nombre.length()>0) {
57 //                                      if (nombre.length()>0) {
58 //
59 //                                              Contexto.createObject(nombre);
60 //
61 //                                              Object[] lista = list.getSelectedValues();
62 //                                              for (int i=0;i<lista.length;i++) {
63 //                                                      FCAElement a = (FCAElement)lista[i];
64 //                                                      Contexto.addRelation(a.getNombre(),nom
   bre);
65 //                                              }
66 //                                              Salida.Imprimirln(Contexto.imprimir());
67 //
68 //
69 //                                      } else {
70 //                                              int n = JOptionPane.showConfirmDialog(
71 //                                                      DummyPaella.getGUI(),
72 //                                                      "You have not choosen any Tags.\nAre y
   ou sure?",
73 //                                                      "Warning",
74 //                                                      JOptionPane.OK_CANCEL_OPTION);
75 //
76 //                                              if (n == JOptionPane.OK_OPTION) {
77 //                                                      salir = true;
78 //                                              }
79 //
80 //                                      }
```

```
81  //                                  } else {
82  //                                                  JOptionPane.showMessageDialog(DummyPaella.getG
    UI(),
83  //                                                 "You must enter the Individual's name",
84  //                                                 "Error",
85  //                                                 JOptionPane.ERROR_MESSAGE);
86  //                                  }
87  //
88  //                          if (salir) {
89  ////                                         NuevaClase.value = valores;
90  ////                                         NuevaClase.value = null;
91  //                                  NuevoIndividuo.dialog.setVisible(false);
92  //                          }
93  //
94  //                  }
95  //      });
96          getRootPane().setDefaultButton(setButton);
97
98          //main part of the dialog
99          list = new JList(data);
100         list.setSelectionMode(ListSelectionModel.MULTIPLE_INTERVAL_SELECTION);
101         list.addMouseListener(new MouseAdapter() {
102                 public void mouseClicked(MouseEvent e) {
103                         if (e.getClickCount() == 2) {
104                                 setButton.doClick();
105                         }
106                 }
107         });
108
109         JScrollPane listScroller = new JScrollPane(list);
110         listScroller.setPreferredSize(new Dimension(250, 80));
111         //XXX: Must do the following, too, or else the scroller thinks
112         //XXX: it's taller than it is:
113         listScroller.setMinimumSize(new Dimension(250, 80));
114         listScroller.setAlignmentX(LEFT_ALIGNMENT);
115
116         //Create a container so that we can add a title around
117         //the scroll pane.  Can't add a title directly to the
118         //scroll pane because its background would be white.
119         //Lay out the label and scroll pane from top to button.
120         JPanel listPane = new JPanel();
121         listPane.setLayout(new BoxLayout(listPane, BoxLayout.Y_AXIS));
122         JLabel label = new JLabel("Individual's Tags");
123         listPane.add(label);
124         listPane.add(Box.createRigidArea(new Dimension(0,5)));
125         listPane.add(listScroller);
126         listPane.setBorder(BorderFactory.createEmptyBorder(10,10,10,10));
127
128         //Lay out the buttons from left to right.
129         JPanel buttonPane = new JPanel();
130         buttonPane.setLayout(new BoxLayout(buttonPane, BoxLayout.X_AXIS));
131         buttonPane.setBorder(BorderFactory.createEmptyBorder(0, 10, 10, 10));
132         buttonPane.add(Box.createHorizontalGlue());
133         buttonPane.add(cancelButton);
134         buttonPane.add(Box.createRigidArea(new Dimension(10, 0)));
135         buttonPane.add(setButton);
136
137         //Put everything together, using the content pane's BorderLayout.
138         Container contentPane = getContentPane();
139         contentPane.add(Encabezado, BorderLayout.NORTH);
140         contentPane.add(listPane, BorderLayout.CENTER);
141         contentPane.add(buttonPane, BorderLayout.SOUTH);
142
143         pack();
144 }
145
146
147     /**
148      * Set up the dialog.  The first argument can be null,
149      * but it really should be a component in the dialog's
150      * controlling frame.
151      */
152     public static void initialize(Component comp,
153                                   Object[] datos) {
154         Frame frame = JOptionPane.getFrameForComponent(comp);
155         dialog = new NuevoIndividuo(frame, datos);
156     }
157
158 //    public static FCAObject showDialog(Component comp, FCAObject[] initialValue) {
159 //        if (dialog != null) {
160 //            dialog.setValue(initialValue);
161 //            dialog.setLocationRelativeTo(comp);
```

```
162 //             dialog.setVisible(true);
163 //         } else {
164 //             System.err.println("ListDialog requires you to call initialize "
165 //                                + "before calling showDialog.");
166 //         }
167 //
168 //         return value;
169 //     }
170 //
171 //     private void setValue(FCAObject[] newValue) {
172 //
173 ////             if (newValue != null) {
174 ////                     for(int i=0;i<newValue.length;i++) {
175 ////                             value.add(newValue[i]);
176 ////                     }
177 ////             }
178 ////         list.setSelectedValue(value, true);
179 //
180 //
181 //
182 //     }
183 //
184
185
186 }
```

```
1  package es.us.ccia.dummy.dibujo.ventanas.dialogo;
2
3  import java.awt.BorderLayout;
4  import java.awt.Component;
5  import java.awt.Container;
6  import java.awt.Dimension;
7  import java.awt.Frame;
8  import java.awt.event.ActionEvent;
9  import java.awt.event.ActionListener;
10 import java.awt.event.MouseAdapter;
11 import java.awt.event.MouseEvent;
12
13 import javax.swing.*;
14
15
16 @SuppressWarnings("serial")
17 public class NuevaClase extends JDialog {
18
19         public  static NuevaClase dialog;
20     private static Object value;
21     private JList list;
22         private JTextField NombreC;
23
24     private NuevaClase(Frame frame, Object[] data, String title) {
25         super(frame, title, true);
26
27
28         JLabel Texto = new JLabel("Class Name:");
29         NombreC = new JTextField();
30
31
32
33         JPanel Encabezado = new JPanel();
34         Encabezado.setLayout(new BoxLayout(Encabezado,BoxLayout.Y_AXIS));
35         Encabezado.setBorder(BorderFactory.createEmptyBorder(10,10,10,10));
36         Encabezado.add(Texto);
37         Encabezado.add(NombreC);
38
39
40
41         //buttons
42         JButton cancelButton = new JButton("Cancel");
43         cancelButton.addActionListener(new ActionListener() {
44                 public void actionPerformed(ActionEvent e) {
45                         NuevaClase.dialog.setVisible(false);
46                 }
47         });
48
49
50         final JButton setButton = new JButton("Accept");
51 //      setButton.addActionListener(new ActionListener() {
52 //              public void actionPerformed(ActionEvent e) {
53 //
54 //                      boolean salir = false;
55 //                              String nombre = NombreC.getText().trim();
56 //                              Contexto.createAttribute(nombre);
57 //
58 //                      if (nombre.length()>0) {
59 //
60 //                              Object[] lista = list.getSelectedValues();
61 //
62 //                              if (lista.length>0) {
63 //
64 //
65 //                                      for(int i=0;i<lista.length;i++) {
66 //                                              FCAElement a = (FCAElement)lista[i];
67 //                                              Contexto.addRelation(nombre, a.getNombre());
68 //                                      }
69 //
70 //                                      System.out.println(Contexto.imprimir());
71 //
72 //
73 //                              } else {
74 //                                      int n = JOptionPane.showConfirmDialog(
75 //                                                      DummyPaella.getGUI(),
76 //                                          "You have not choosen any Individuals.\nAre you su
   re?",
77 //                                          "Warning",
78 //                                          JOptionPane.OK_CANCEL_OPTION);
79 //
80 //                                      if (n == JOptionPane.OK_OPTION) {
81 //                                              salir = true;
```

```
82  //                                      }
83  //
84  //                                  }
85  //
86  //                          } else {
87  //                                  JOptionPane.showMessageDialog(DummyPaella.getGUI(),
88  //                                          "You must enter the tag's name",
89  //                                          "Error",
90  //                                          JOptionPane.ERROR_MESSAGE);
91  //                          }
92  //                          if (salir) {
93  ////                                     NuevaClase.value = valores;
94  ////                                     NuevaClase.value = null;
95  //                                  NuevaClase.dialog.setVisible(false);
96  //                          }
97  //
98  //                  }
99  //      });
100         getRootPane().setDefaultButton(setButton);
101
102         //main part of the dialog
103         list = new JList(data);
104         list.setSelectionMode(ListSelectionModel.MULTIPLE_INTERVAL_SELECTION);
105         list.addMouseListener(new MouseAdapter() {
106                 public void mouseClicked(MouseEvent e) {
107                         if (e.getClickCount() == 2) {
108                                 setButton.doClick();
109                         }
110                 }
111         });
112
113         JScrollPane listScroller = new JScrollPane(list);
114         listScroller.setPreferredSize(new Dimension(250, 80));
115         //XXX: Must do the following, too, or else the scroller thinks
116         //XXX: it's taller than it is:
117         listScroller.setMinimumSize(new Dimension(250, 80));
118         listScroller.setAlignmentX(LEFT_ALIGNMENT);
119
120         //Create a container so that we can add a title around
121         //the scroll pane.  Can't add a title directly to the
122         //scroll pane because its background would be white.
123         //Lay out the label and scroll pane from top to button.
124         JPanel listPane = new JPanel();
125         listPane.setLayout(new BoxLayout(listPane, BoxLayout.Y_AXIS));
126         JLabel label = new JLabel("Etiqueta a poner");
127         listPane.add(label);
128         listPane.add(Box.createRigidArea(new Dimension(0,5)));
129         listPane.add(listScroller);
130         listPane.setBorder(BorderFactory.createEmptyBorder(10,10,10,10));
131
132         //Lay out the buttons from left to right.
133         JPanel buttonPane = new JPanel();
134         buttonPane.setLayout(new BoxLayout(buttonPane, BoxLayout.X_AXIS));
135         buttonPane.setBorder(BorderFactory.createEmptyBorder(0, 10, 10, 10));
136         buttonPane.add(Box.createHorizontalGlue());
137         buttonPane.add(cancelButton);
138         buttonPane.add(Box.createRigidArea(new Dimension(10, 0)));
139         buttonPane.add(setButton);
140
141         //Put everything together, using the content pane's BorderLayout.
142         Container contentPane = getContentPane();
143         contentPane.add(Encabezado, BorderLayout.NORTH);
144         contentPane.add(listPane, BorderLayout.CENTER);
145         contentPane.add(buttonPane, BorderLayout.SOUTH);
146
147         pack();
148 }
149
150
151     /**
152      * Set up the dialog.  The first argument can be null,
153      * but it really should be a component in the dialog's
154      * controlling frame.
155      */
156     public static void initialize(Component comp,
157                                   Object[] possibleValues,
158                                   String title) {
159         Frame frame = JOptionPane.getFrameForComponent(comp);
160         dialog = new NuevaClase(frame, possibleValues, title);
161     }
162
163 //    public static FCAAttribute showDialog(
```

```
164 //                Component comp, FCAObject[] initialValue) {
165 //
166 //      if (dialog != null) {
167 //            dialog.setValue(initialValue);
168 //            dialog.setLocationRelativeTo(comp);
169 //            dialog.setVisible(true);
170 //        } else {
171 //            System.err.println("ListDialog requires you to call initialize "
172 //                               + "before calling showDialog.");
173 //        }
174 //
175 //        return value;
176 //    }
177 //
178 //    private void setValue(FCAObject[] newValue) {
179 //
180 ////              if (newValue != null) {
181 ////                      for(int i=0;i<newValue.length;i++) {
182 ////                              value.add(newValue[i]);
183 ////                      }
184 ////              }
185 ////        list.setSelectedValue(value, true);
186 //
187 //
188 //
189 //    }
190
191
192
193 }
```

```
1  package es.us.ccia.dummy.dibujo.ventanas.dialogo;
2
3  import java.awt.*;
4  import java.awt.event.*;
5  import java.util.*;
6  import javax.swing.*;
7
8
9  @SuppressWarnings({"unchecked","serial"})
10 public class ElegirIndividuos extends JDialog {
11
12         public  static ElegirIndividuos dialog;
13     private static ArrayList value = new ArrayList();
14     private JList list;
15
16     private ElegirIndividuos(Frame frame, Object[] data, String title) {
17         super(frame, title, true);
18
19         //buttons
20         JButton nuevoButton = new JButton("Nuevo");
21 //      nuevoButton.addActionListener(new ActionListener() {
22 //              public void actionPerformed(ActionEvent e) {
23 //
24 //                      Object[] names = Contexto.getAttributes().toArray();
25 //                      NuevoIndividuo.initialize(DummyPaella.getGUI(), names);
26 //                      FCAObject opcion = NuevoIndividuo.showDialog(null, null);
27 //                      ArrayList resultado = new ArrayList();
28 //                      resultado.add(opcion);
29 //                      ElegirIndividuos.value = resultado;
30 //                      ElegirIndividuos.dialog.setVisible(false);
31 //              }
32 //      });
33 //
34
35
36         JButton cancelButton = new JButton("Cancelar");
37         cancelButton.addActionListener(new ActionListener() {
38                 public void actionPerformed(ActionEvent e) {
39                         ElegirIndividuos.dialog.setVisible(false);
40                 }
41         });
42
43         final JButton setButton = new JButton("Aceptar");
44         setButton.addActionListener(new ActionListener() {
45                 public void actionPerformed(ActionEvent e) {
46
47                         Object[] lista = list.getSelectedValues();
48                         int l1 = lista.length;
49                         ArrayList valores = new ArrayList();
50
51                         for(int i=0;i<l1;i++) {
52 //                              valores.add((FCAObject)lista[i]);
53                         }
54
55
56                         ElegirIndividuos.value = valores;
57                         ElegirIndividuos.dialog.setVisible(false);
58                 }
59         });
60         getRootPane().setDefaultButton(setButton);
61
62         //main part of the dialog
63         list = new JList(data);
64         list.setSelectionMode(ListSelectionModel.MULTIPLE_INTERVAL_SELECTION);
65         list.addMouseListener(new MouseAdapter() {
66                 public void mouseClicked(MouseEvent e) {
67                         if (e.getClickCount() == 2) {
68                                 setButton.doClick();
69                         }
70                 }
71         });
72
73         JScrollPane listScroller = new JScrollPane(list);
74         listScroller.setPreferredSize(new Dimension(250, 80));
75         //XXX: Must do the following, too, or else the scroller thinks
76         //XXX: it's taller than it is:
77         listScroller.setMinimumSize(new Dimension(250, 80));
78         listScroller.setAlignmentX(LEFT_ALIGNMENT);
79
80         //Create a container so that we can add a title around
81         //the scroll pane.  Can't add a title directly to the
82         //scroll pane because its background would be white.
```

```
83          //Lay out the label and scroll pane from top to button.
84          JPanel listPane = new JPanel();
85          listPane.setLayout(new BoxLayout(listPane, BoxLayout.Y_AXIS));
86          JLabel label = new JLabel("Choose the objects:");
87          listPane.add(label);
88          listPane.add(Box.createRigidArea(new Dimension(0,5)));
89          listPane.add(listScroller);
90          listPane.setBorder(BorderFactory.createEmptyBorder(10,10,10,10));
91
92          //Lay out the buttons from left to right.
93          JPanel buttonPane = new JPanel();
94          buttonPane.setLayout(new BoxLayout(buttonPane, BoxLayout.X_AXIS));
95          buttonPane.setBorder(BorderFactory.createEmptyBorder(0, 10, 10, 10));
96
97          buttonPane.add(Box.createHorizontalGlue());
98          buttonPane.add(nuevoButton);
99
100         buttonPane.add(Box.createRigidArea(new Dimension(10, 0)));
101         buttonPane.add(cancelButton);
102         buttonPane.add(Box.createRigidArea(new Dimension(10, 0)));
103         buttonPane.add(setButton);
104
105         //Put everything together, using the content pane's BorderLayout.
106         Container contentPane = getContentPane();
107         contentPane.add(listPane, BorderLayout.CENTER);
108         contentPane.add(buttonPane, BorderLayout.SOUTH);
109
110         pack();
111 }
112
113
114
115
116     /**
117      * Set up the dialog.  The first argument can be null,
118      * but it really should be a component in the dialog's
119      * controlling frame.
120      */
121     public static void initialize(Component comp,
122                                   Object[] possibleValues,
123                                   String title) {
124         Frame frame = JOptionPane.getFrameForComponent(comp);
125         dialog = new ElegirIndividuos(frame, possibleValues,title);
126     }
127
128 //      public static ArrayList<FCAObject> showDialog(Component comp, FCAObject[] initialValue
    ) {
129 //        if (dialog != null) {
130 //            dialog.setValue(initialValue);
131 //            dialog.setLocationRelativeTo(comp);
132 //            dialog.setVisible(true);
133 //        } else {
134 //            System.err.println("ListDialog requires you to call initialize "
135 //                               + "before calling showDialog.");
136 //        }
137 //
138 //        return value;
139 //    }
140 //
141 //    private void setValue(FCAObject[] newValue) {
142 //
143 //      if (newValue != null) {
144 //              for(int i=0;i<newValue.length;i++) {
145 //                      value.add(newValue[i]);
146 //              }
147 //      }
148 //        list.setSelectedValue(value, true);
149 //    }
150
151
152
153 }
```

```
1  package es.us.ccia.dummy.dibujo.ventanas;
2
3  /**
4   * Version 0.9:
5   *
6   *    20080715
7   *    - Implementacion inicial
8   *
9   *================================================
10  * TODO:
11  *
12  */
13
14 import java.awt.*;
15 import java.awt.event.ActionListener;
16
17 import javax.swing.JButton;
18 import javax.swing.JPanel;
19
20 import es.us.ccia.dummy.dibujo.*;
21
22 /***
23  *
24  * @author Gonzalo A. Aranda Corral
25  * @version 0.9.0
26  * @since   12/07/2008
27  *
28  */
29 @SuppressWarnings("serial")
30 public class VentanaBoton extends JPanel {
31
32         JPanel panelCls;
33 //    LabeledComponent labelClases;
34       JPanel tapiz;
35       JButton miboton;
36
37 //      VentanaClases vclases;
38 //      VentanaIndividuos vindiv;
39
40
41       public VentanaBoton(ActionListener g) {
42               super();
43 //              vclases = g.getVClases();
44 //              vindiv  = g.getVIndividuos();
45
46               Dimension dim = new Dimension(30,30);
47
48               miboton = new JButton("S A V E !");
49               miboton.setPreferredSize(dim);
50
51               miboton.addActionListener((ActionListener)g);
52
53               tapiz = new JPanel();
54               tapiz.setMinimumSize(dim);
55               tapiz.setPreferredSize(dim);
56               tapiz.setSize(dim);
57
58               //tapiz.setLayout(new BorderLayout());
59               tapiz.setLayout(new FlowLayout());
60       tapiz.add(miboton,BorderLayout.CENTER);
61
62       panelCls = new JPanel();
63       panelCls.setPreferredSize(dim);
64
65       panelCls.setLayout(new BorderLayout());
66               panelCls.add(tapiz,BorderLayout.CENTER);
67
68       setLayout(new BorderLayout());
69       add(panelCls,BorderLayout.CENTER);
70
71       }
72
73 }
```

```
1  package es.us.ccia.dummy.dibujo.ventanas;
2
3  import java.awt.*;
4  import java.util.*;
5  import javax.swing.*;
6
7  import es.us.ccia.dummy.dibujo.GUI;
8  import es.us.ccia.dummy.dibujo.eventos.EventosClases;
9
10 /***
11  *
12  * @author  Gonzalo A. Aranda Corral
13  * @version 0.1.0
14  * @since   28/02/2009
15  * @see
16  *    20090228 Carga de las etiquetas del usuario
17  *
18  *
19  */
20 @SuppressWarnings({"serial","unchecked"})
21 public class VentanaEtiquetas extends JComponent {
22
23         JList lista;
24         DefaultListModel Lmodelo;
25         JScrollPane panelCls;
26
27
28         public VentanaEtiquetas(GUI g) {
29                 super();
30
31                 lista   = new JList();
32                 lista.setCellRenderer(new ClasesRenderer());
33         Lmodelo = new DefaultListModel();
34         lista.setModel(Lmodelo);
35
36         lista.addListSelectionListener(new EventosClases(g));
37
38         reiniciar(null);
39
40         panelCls = new JScrollPane(lista);
41
42         setLayout(new BorderLayout());
43         add(panelCls,BorderLayout.CENTER);
44
45         }
46
47
48         public void reiniciar(Collection lista) {
49
50                 if (lista != null) {
51
52                         Set orden = new TreeSet();
53
54                         Lmodelo.clear();
55
56                 for (Iterator i= lista.iterator();i.hasNext();) {
57                         ElementoLista cl =
58                                 new ElementoLista((i.next()).toString(),true);
59                         orden.add(cl);
60                 }
61
62                 Iterator i = orden.iterator();
63                 while (i.hasNext())
64                         Lmodelo.addElement(i.next());
65
66         //      lista.setSelectedValue(null, false);
67                 }
68         }
69
70
71         public ArrayList getSeleccion() {
72
73                 ArrayList devolver = new ArrayList();
74 //              int tamano = lista.getSelectedIndices().length;
75 //
76 //              for(int indice=0;indice<tamano;indice++) {
77 //
78 //                      String cadena = lista.getSelectedValues()[indice].toString();
79 //                      Salida.Imprimirln(cadena);
80 //
81 //
82 //                      FCAAttribute clase = Contexto.getAttribute(cadena);
```

```
83  //                      devolver.add(clase);
84  //              }
85  //
86                  return devolver;
87          }
88
89          public void addClase(String clase) {
90                  ElementoLista cl = new ElementoLista(clase,true);
91                  Lmodelo.addElement(cl);
92          }
93
94
95
96
97  }
98
99  /*********************************************
100  *
101  *
102  * @author garanda
103  *
104  */
105
106 @SuppressWarnings("serial")
107 class ClasesRenderer extends JLabel implements ListCellRenderer {
108
109 private static final Color HIGHLIGHT_COLOR = new Color(0, 0, 128);
110
111   public ClasesRenderer() {
112     setOpaque(true);
113     setIconTextGap(12);
114   }
115
116   public Component getListCellRendererComponent(JList list, Object value,
117       int index, boolean isSelected, boolean cellHasFocus) {
118           ElementoLista entry = (ElementoLista) value;
119
120 //        setIcon(Icons.getClsIcon());
121           setText(entry.getNombre());
122
123         setBackground(Color.white);
124         setForeground(Color.black);
125
126     if (isSelected) {
127         if (entry.getConsistente()) {
128                         setBackground(HIGHLIGHT_COLOR);
129                         setForeground(Color.white);
130                 this.setFont(new Font("Default",Font.BOLD,12));
131         } else {
132                 setForeground(Color.white);
133                 setBackground(Color.red);
134                 this.setFont(new Font("Default",Font.BOLD,12));
135         }
136
137     } else {
138                 setBackground(Color.white);
139                 setForeground(Color.black);
140         if (entry.getConsistente()) {
141                 this.setFont(new Font("Default",Font.PLAIN,12));
142         } else {
143                 setForeground(Color.red);
144                 this.setFont(new Font("Default",Font.BOLD,12));
145         }
146
147     }
148     return this;
149   }
150 }
```

```java
package es.us.ccia.dummy.dibujo.ventanas;
/**
 * Version 0.9:
 *
 *    20080715
 *    - Implementacion inicial
 *
 *================================================
 * TODO:
 *
 */

/***
 *
 * @author Gonzalo A. Aranda Corral
 * @version 0.9.0
 * @since   12/07/2008
 *
 */
@SuppressWarnings({"unchecked"})
public class ElementoLista implements Comparable {

	String  Nombre;
	boolean Consist;

	public ElementoLista(String _nombre, boolean _consist) {
		Nombre  = _nombre;
		Consist = _consist;
	}

	public ElementoLista(String _nombre) {
		Nombre  = _nombre;
		Consist = true;
	}

	public String getNombre() {
		return Nombre;
	}

	public boolean getConsistente() {
		return Consist;
	}

	@Override
	public String toString() {
		return Nombre;
	}

	public int compareTo(Object o) {
		return this.getNombre().compareTo( ((ElementoLista)o).getNombre());
	}

}
```

```
1  package es.us.ccia.dummy.dibujo.ventanas;
2  /**
3   * Version 0.9:
4   *
5   *    20080715
6   *    − Implementacion inicial
7   *
8   *================================================
9   * TODO:
10  *
11  */
12
13 import java.awt.BorderLayout;
14 import java.awt.Dimension;
15 import java.util.Iterator;
16
17 import javax.swing.JComponent;
18 import javax.swing.JScrollPane;
19
20 import es.us.ccia.dummy.dibujo.GUI;
21 import es.us.ccia.dummy.dibujo.eventos.Movimientos;
22 import es.us.ccia.dummy.dibujo.ventanas.AreaDibujo;
23 import es.us.ccia.dummy.teoria.regiones.Elemento;
24 import es.us.ccia.dummy.teoria.regiones.ListaElementos;
25 import es.us.ccia.dummy.teoria.regiones.Punto;
26 import es.us.ccia.dummy.teoria.regiones.Region;
27
28 /***
29  *
30  * @author Gonzalo A. Aranda Corral
31  * @version 0.9.0
32  * @since   12/07/2008
33  *
34  */
35
36 @SuppressWarnings("serial")
37 public class VentanaDibujo extends JComponent {
38
39         public static final int TAMANOX = 700;
40         public static final int TAMANOY = 600;
41
42     VentanaRelaciones vrelaciones;
43         VentanaHistorial vhist;
44
45         JScrollPane panelCls;
46         AreaDibujo midibujo;
47         Movimientos raton;
48         ListaElementos ListaE = null;
49
50
51         public VentanaDibujo(GUI g) {
52                 super();
53 //              vrelaciones  = g.getVentanaRelaciones();
54 //              vhist = g.getVentanaHistorial();
55
56                 Dimension dim = new Dimension(TAMANOX+100,TAMANOY+100);
57
58
59         midibujo = new AreaDibujo();
60         midibujo.setMinimumSize(dim);
61         midibujo.setPreferredSize(dim);
62         midibujo.setSize(dim);
63
64         raton = new Movimientos(g,ListaE);
65         midibujo.addMouseListener(raton);
66         midibujo.addMouseMotionListener(raton);
67
68
69
70
71
72 //        labelClases = new LabeledComponent("DRAWING AREA", midibujo);
73 //        labelClases.addHeaderButton(new CrearClaseB(this));
74 //        labelClases.addHeaderButton(new CrearIndivB(this));
75
76
77         setLayout(new BorderLayout());
78 //      add(labelClases,BorderLayout.CENTER);
79         add(midibujo,BorderLayout.CENTER);
80         }
81
82         public AreaDibujo getDibujo() {
```

```
83                      return this.midibujo;
84              }
85
86
87              public boolean estaModificada() {
88                      return raton.estaModificado();
89              }
90
91              public void setModificado(boolean m) {
92                      raton.setModificado(m);
93              }
94
95              @SuppressWarnings("unchecked")
96              public void actualizar(ListaElementos _lista) {
97
98                      // Aqui hay que duplicar la lista _lista
99
100                     ListaElementos ll = new ListaElementos();
101
102                     Iterator i = _lista.iterator();
103                     while(i.hasNext()) {
104                             Elemento e = (Elemento)i.next();
105                             Elemento nuevo = null;
106                             if (e.getTipo() == Elemento.PUNTO) {
107                                     nuevo = new Punto(e.getArgumento(),e.getX1(),e.getY1());
108
109                             } else if (e.getTipo() == Elemento.REGION) {
110                                     Region r = (Region)e;
111                                     nuevo = new Region(r.getArgumento(),r.getX1(),r.getY1(),
112                                                     r.getX2(),r.getY2());
113                             } else {
114                                     System.out.println("Error en la lista ");
115                                     nuevo = null;
116                             }
117                             ll.add(nuevo);
118                     }
119
120                     ListaE = ll;
121                     this.midibujo.actualizar(ll);
122                     raton.actualizar(ll);
123
124             }
125
126             public ListaElementos getListaElementos() {
127                     return ListaE;
128             }
129
130
131
132     }
```

```
1  package es.us.ccia.dummy.dibujo.ventanas;
2  /**
3   * Version 0.9:
4   *
5   *    20080715
6   *    − Implementacion inicial
7   *
8   *================================================
9   * TODO:
10  *
11  */
12
13 import java.awt.BorderLayout;
14 import java.awt.Color;
15 import java.awt.Component;
16 import java.awt.Font;
17
18 import javax.swing.DefaultListModel;
19 import javax.swing.JComponent;
20 import javax.swing.JLabel;
21 import javax.swing.JList;
22 import javax.swing.JScrollPane;
23 import javax.swing.ListCellRenderer;
24
25
26 /***
27  *
28  * @author Gonzalo A. Aranda Corral
29  * @version 0.9.0
30  * @since   12/07/2008
31  *
32  */
33 @SuppressWarnings({ "serial" })
34 public class VentanaHistorial extends JComponent {
35
36         int numpasos = 0;
37
38 //      ListaRelacionRCC8 viejo;
39
40         JList lista;
41         DefaultListModel Lmodelo;
42         JScrollPane panelInd;
43 //    LabeledComponent labelInd;
44
45         public VentanaHistorial () {
46                 super();
47
48 //              viejo    = new ListaRelacionRCC8();
49
50                 lista = new JList();
51                 lista.setCellRenderer(new HistoriaRenderer());
52                 Lmodelo = new DefaultListModel();
53                 lista.setModel(Lmodelo);
54
55         panelInd = new JScrollPane(lista);
56         panelInd.setSize(100, 100);
57 //        labelInd = new LabeledComponent("HISTORY", panelInd);
58         setLayout(new BorderLayout());
59         add(panelInd,BorderLayout.CENTER);
60
61
62         }
63
64         public void actualizar() {
65
66 //              Lmodelo.clear();
67 //
68 //              LinkedList lk = DummyPaella.getTeoria().getLCambios();
69 //
70 //
71 //              // AHORA HAY QUE RECORRER LA LISTA
72 //
73 //              Iterator i = lk.iterator();
74 //              while (i.hasNext()){
75 //                      ListaCambios lc = (ListaCambios)i.next();
76 //                      Lmodelo.addElement(new ElementoLista(lc.toString()));
77 //              }
78         }
79
80         public void limpiar() {
81                 Lmodelo.clear();
82         }
```

```
83
84
85  }
86
87
88
89
90
91
92
93  /*********************************************
94   *
95   *
96   * @author garanda
97   *
98   */
99
100 @SuppressWarnings("serial")
101 class HistoriaRenderer extends JLabel implements ListCellRenderer {
102
103 private static final Color HIGHLIGHT_COLOR = new Color(0, 0, 128);
104
105   public HistoriaRenderer() {
106     setOpaque(true);
107     setIconTextGap(12);
108   }
109
110   public Component getListCellRendererComponent(JList list, Object value,
111       int index, boolean isSelected, boolean cellHasFocus) {
112
113           if (value != null) {
114 //                Salida.Imprimirln(value.getClass().getName());
115
116                   ElementoLista entry = (ElementoLista) value;
117
118                   //setIcon(Icons.getClsesAndInstancesIcon()   );
119                   setText(entry.getNombre().toUpperCase());
120
121                   setBackground(Color.white);
122                   setForeground(Color.black);
123
124                   if (isSelected) {
125                           setBackground(HIGHLIGHT_COLOR);
126                           setForeground(Color.white);
127                           this.setFont(new Font("Default",Font.BOLD,12));
128                   } else {
129                           setBackground(Color.white);
130                           setForeground(Color.black);
131                           this.setFont(new Font("Default",Font.PLAIN,12));
132                   }
133
134           }
135
136           return this;
137   }
138 }
```

```
1  package es.us.ccia.dummy.dibujo.ventanas;
2
3  import java.awt.BorderLayout;
4  import java.awt.Color;
5  import java.awt.Component;
6  import java.awt.Font;
7  import java.util.ArrayList;
8  import java.util.Collection;
9  import java.util.Iterator;
10 import java.util.Set;
11 import java.util.TreeSet;
12
13 import javax.swing.DefaultListModel;
14 import javax.swing.JComponent;
15 import javax.swing.JLabel;
16 import javax.swing.JList;
17 import javax.swing.JScrollPane;
18 import javax.swing.ListCellRenderer;
19
20 import es.us.ccia.dummy.dibujo.GUI;
21 import es.us.ccia.dummy.dibujo.eventos.EventosClases;
22 import es.us.ccia.dummy.dibujo.eventos.EventosObjetos;
23
24
25 /***
26  *
27  * @author  Gonzalo A. Aranda Corral
28  * @version 0.1.0
29  * @since   28/02/2009
30  * @see
31  *    20090228 Carga de las objetos
32  *
33  *
34  */
35
36 @SuppressWarnings({"serial","unchecked"})
37 public class VentanaObjetos extends JComponent {
38
39
40         JList lista;
41         DefaultListModel Lmodelo;
42         JScrollPane panelInd;
43
44         public VentanaObjetos (GUI g) {
45                 super();
46
47                 lista = new JList();
48                 lista.setCellRenderer(new IndividRenderer());
49                 Lmodelo = new DefaultListModel();
50                 lista.setModel(Lmodelo);
51
52         lista.addListSelectionListener(new EventosObjetos(g));
53
54                 reiniciar(null);
55
56         panelInd = new JScrollPane(lista);
57
58         setLayout(new BorderLayout());
59         add(panelInd,BorderLayout.CENTER);
60 //        addIndividuo("individuos");
61
62         }
63
64         public void reiniciar(Collection lista) {
65
66                 if (lista != null) {
67
68                         Set orden = new TreeSet();
69
70                         Lmodelo.clear();
71
72                 for (Iterator i= lista.iterator();i.hasNext();) {
73                         ElementoLista cl =
74                                 new ElementoLista((i.next()).toString(),true);
75                         orden.add(cl);
76                 }
77
78                 Iterator i = orden.iterator();
79                 while (i.hasNext())
80                         Lmodelo.addElement(i.next());
81
82         //      lista.setSelectedValue(null, false);
```

```
                }



        }

        public ArrayList getSeleccion() {

                ArrayList devolver = new ArrayList();
                int tamano = lista.getSelectedIndices().length;

                for(int indice=0;indice<tamano;indice++) {
                        devolver.add(lista.getSelectedValues()[indice].toString());
                }

                return devolver;
        }


        public void addIndividuo(String nombre) {
                ElementoLista cl = new ElementoLista(nombre,true);
                Lmodelo.addElement(cl);
        }

}

/*********************************************
 *
 * @author garanda
 *
 */

@SuppressWarnings("serial")
class IndividRenderer extends JLabel implements ListCellRenderer {

private static final Color HIGHLIGHT_COLOR = new Color(0, 0, 128);

  public IndividRenderer() {
    setOpaque(true);
    setIconTextGap(12);
  }

  public Component getListCellRendererComponent(JList list, Object value,
      int index, boolean isSelected, boolean cellHasFocus) {
          ElementoLista entry = (ElementoLista) value;

//        setIcon(Icons.getInstanceIcon());
          setText(entry.getNombre());

        setBackground(Color.white);
        setForeground(Color.black);

    if (isSelected) {
                        setBackground(HIGHLIGHT_COLOR);
                        setForeground(Color.white);
                        this.setFont(new Font("Default",Font.BOLD,12));
    } else {
                setBackground(Color.white);
                setForeground(Color.black);
                this.setFont(new Font("Default",Font.PLAIN,12));

    }
    return this;
  }
}
```

```
1  package es.us.ccia.dummy.dibujo.ventanas;
2
3  /**
4   * Version 0.9:
5   *
6   *    20080715
7   *    − Implementacion inicial
8   *
9   *================================================
10  * TODO:
11  *
12  */
13
14 import java.awt.Color;
15 import java.awt.Graphics;
16 import java.util.Iterator;
17
18 import javax.swing.JPanel;
19
20 import es.us.ccia.dummy.teoria.regiones.Elemento;
21 import es.us.ccia.dummy.teoria.regiones.ListaElementos;
22
23 //import es.us.ccia.dummy.teoria.regiones.Elemento;
24 //import es.us.ccia.dummy.teoria.regiones.I_Elemento;
25
26 /***
27  *
28  * @author Gonzalo A. Aranda Corral
29  * @version 0.9.0
30  * @since   12/07/2008
31  *
32  */
33 @SuppressWarnings({})
34 public class AreaDibujo extends JPanel //implements I_Elemento
35 {
36         private static final long serialVersionUID = 1L;
37         ListaElementos ListaE = null;
38
39
40
41
42         public AreaDibujo() {
43                 setBackground(Color.WHITE);
44         }
45
46
47         /****************************
48          * METODOS DE PINTAR
49          ****************************/
50         @SuppressWarnings("unchecked")
51         @Override
52         public void paint(Graphics g)
53     {
54         super.paint(g);
55         g.clearRect(0, 0, this.getWidth(), this.getHeight());
56
57         if (ListaE != null) {
58                 Iterator i = ListaE.iterator();
59                 while (i.hasNext()) {
60                         Elemento ele = (Elemento)i.next();
61 //                      System.out.println(ele.toString());
62
63                         ele.pintar(g);
64                         }
65         }
66     }
67
68         public void actualizar(ListaElementos _lista) {
69                 ListaE = _lista;
70                 repaint();
71
72         }
73
74 }
```

```
1  package es.us.ccia.dummy.dibujo.ventanas;
2  /**
3   * Version 0.9:
4   *
5   *    20080715
6   *    − Implementacion inicial
7   *
8   *================================================
9   * TODO:
10  *
11  */
12
13 import java.awt.BorderLayout;
14 import java.awt.Color;
15 import java.awt.Component;
16 import java.awt.Font;
17 import java.util.ArrayList;
18 import java.util.Iterator;
19 import java.util.LinkedList;
20
21 import javax.swing.DefaultListModel;
22 import javax.swing.JComponent;
23 import javax.swing.JLabel;
24 import javax.swing.JList;
25 import javax.swing.JScrollPane;
26 import javax.swing.ListCellRenderer;
27
28 import es.us.ccia.dummy.dibujo.GUI;
29
30
31 /***
32  *
33  * @author Gonzalo A. Aranda Corral
34  * @version 0.9.0
35  * @since   12/07/2008
36  *
37  */
38 @SuppressWarnings({"serial","unchecked"})
39 public class VentanaRelaciones extends JComponent {
40
41         JList lista;
42         DefaultListModel Lmodelo;
43         JScrollPane panelCls;
44
45         public VentanaRelaciones(GUI g) {
46                 super();
47                 lista   = new JList();
48                 lista.setCellRenderer(new RelacionRenderer());
49         Lmodelo = new DefaultListModel();
50         lista.setModel(Lmodelo);
51
52         lista.addListSelectionListener(null);
53
54         actualizar(null);
55
56         panelCls = new JScrollPane(lista);
57         panelCls.setSize(100, 300);
58
59         setLayout(new BorderLayout());
60         add(panelCls,BorderLayout.CENTER);
61
62         }
63
64
65         public void actualizar(LinkedList lista) {
66
67                 Lmodelo.clear();
68
69                 if (lista != null) {
70                         Iterator i = lista.iterator();
71
72                         while(i.hasNext()) {
73                                 String rr = (String)i.next();
74                                 ElementoLista cl = new ElementoLista(rr);
75 //                              cl.Consist = rr.estaSeleccionada();
76                                 Lmodelo.addElement(cl);
77                         }
78                 }
79
80         }
81
82 //      public void addRelacion(Relacion relacion) {
```

```
//              if (relacion.getTipo()>0) {
//                      ElementoLista cl = new ElementoLista(relacion.toString());
//                      Lmodelo.addElement(cl);
//              }
//                      //Lmodelo.addElement(relacion.toString());
//      }


        public void vaciar() {
                Lmodelo.clear();
        }


        public ArrayList getSeleccion() {

                ArrayList devolver = new ArrayList();
                int tamano = lista.getSelectedIndices().length;

                for(int indice=0;indice<tamano;indice++) {
                        //devolver.add(lista.getSelectedValues()[indice].toString());
                        devolver.add(lista.getSelectedValues()[indice]);
                }

                return devolver;
        }




}

/*********************************************
 *
 *
 * @author garanda
 *
 */

@SuppressWarnings("serial")
class RelacionRenderer extends JLabel implements ListCellRenderer {

private static final Color HIGHLIGHT_COLOR = new Color(0, 0, 128);

  public RelacionRenderer() {
    setOpaque(true);
    setIconTextGap(12);
  }

  public Component getListCellRendererComponent(JList list, Object value,
      int index, boolean isSelected, boolean cellHasFocus) {

          if (value != null) {

          ElementoLista entry = (ElementoLista) value;

          //setIcon(Icons.getClsesAndInstancesIcon()   );
          setText(entry.getNombre().toUpperCase());

        setBackground(Color.white);
        setForeground(Color.black);

//    if (isSelected) {
        if (isSelected || entry.Consist) {
                        setBackground(HIGHLIGHT_COLOR);
                        setForeground(Color.white);
                this.setFont(new Font("Default",Font.BOLD,12));

    } else {
                setBackground(Color.white);
                setForeground(Color.black);
                this.setFont(new Font("Default",Font.PLAIN,12));
    }
          }
    return this;
  }
}
```

```
1  package es.us.ccia.dummy.razonGrafico;
2
3  import java.awt.geom.Rectangle2D;
4
5  import es.us.ccia.dummy.teoria.*;
6  import es.us.ccia.dummy.teoria.regiones.*;
7  import es.us.ccia.dummy.teoria.relaciones.ListaRelacionRCC;
8  import es.us.ccia.dummy.teoria.relaciones.Relacion;
9  import es.us.ccia.rcc.RCC;
10
11 public class RazonG_RCC8 extends RazonGBase  {
12
13         /**
14          * MÃ©todo generico para el calculo de las posiciones relativas de
15          * dos regiones.
16          * @param e1 Region 1
17          * @param e2 Region 2
18          * @return nos devuelve un entero que sera la constante perteneciente
19          * a la posicion relativa de los dos elementos.
20          */
21         public static int PosicionRelativa(Elemento e1, Elemento e2){
22                 int devolver = -1;
23
24                 if (e1.getClass().getSimpleName().equalsIgnoreCase("Region")) {
25                         if (e2.getClass().getSimpleName().equalsIgnoreCase("Region")) {
26                                 devolver = PosicionRelativaRR((Region)e1,(Region) e2);
27                         } else if (e2.getClass().getSimpleName().equalsIgnoreCase("Punto")) {
28                                 devolver = PosicionRelativaRP((Region)e1,(Punto) e2);
29                         }
30                 } else if (e1.getClass().getSimpleName().equalsIgnoreCase("Punto")) {
31                         if (e2.getClass().getSimpleName().equalsIgnoreCase("Region")) {
32                                 devolver = PosicionRelativaRP((Region)e2,(Punto) e1);
33                         } else if (e2.getClass().getSimpleName().equalsIgnoreCase("Punto")) {
34                                 devolver = PosicionRelativaPP((Punto)e1,(Punto) e2);
35                         }
36                 }
37                 return devolver;
38         }
39
40
41         /*****************************************************
42          *****************************************************
43          *
44          *   POSICIONES RELATIVAS DE REGIONES !!!!
45          *
46          *****************************************************
47          *****************************************************/
48
49
50         public static int PosicionRelativaRR(Region r1, Region r2){
51                 int devolver = RCC.NODEF;
52
53 //              Salida.Imprimirln("Region - Region");
54
55
56                 if (esIgual(r1,r2)) {
57                         devolver = RCC.EQ;
58                 } else {
59
60                         if (!esInterseccion(r1,r2)){
61                         //Salida.Escribirln("NOO Se cortan");
62                                 if (externaConectadas(r1,r2)) {
63                                 devolver = RCC.EC;
64                                 } else {
65                                         devolver = RCC.DC;
66                                 }
67
68                 } else {
69                         //Salida.Escribirln("Se cortan");
70                         if (esSubconjuntoDe(r1,r2)){
71                                 if (esTangente(r1,r2)) {
72                                         devolver = RCC.TPP;
73                                 } else {
74                                         devolver = RCC.NTPP;
75                                 }
76                         } else {
77                                 if (esSubconjuntoDe(r2,r1)) {
78                                         if (esTangente(r1,r2)) {
79                                                 devolver = RCC.TPPi;
80                                         } else {
81                                                 devolver = RCC.NTPPi;
82                                         }
```

```
                                } else {
                                        devolver = RCC.PO;
                                }
                        }

                }
                }

        return devolver;
        }

        public static int PosicionRelativaRP(Region r1, Punto p){
                int devolver = RCC.NODEF;

                //Salida.Imprimirln("Region − Punto");

                Rectangle2D r = new Rectangle2D.Double(
                                r1.getX1(), r1.getY1(),
                                r1.getX2() − r1.getX1(),
                                r1.getY2() − r1.getY1());
                if (r.contains(p.getX1(), p.getY1())) {
                        if (    (r1.getX1() == p.getX1()) ||
                                        (r1.getX2() == p.getX1()) ||
                                        (r1.getY1() == p.getY1()) ||
                                        (r1.getY2() == p.getY1())
                        )
                        {
                                devolver = RCC.FRONT_P;
                        } else {
                                devolver = RCC.IN_P;
                        }
                } else {
                        devolver = RCC.OUT_P;
                }



                return devolver;
        }

    /**
     * Posicion relativa de dos puntos
     *
     * @param p1 Punto 1
     * @param p2 Punto 2
     * @return Devuelve un entero para con la posicion relativa
     */
        public static int PosicionRelativaPP(Punto p1, Punto p2) {
        int devolver = RCC.DIFER_P;

        if (p1.getX1() == p2.getX1() && p1.getY1() == p2.getY1()) {
                devolver = RCC.IGUAL_P;
        }
        return devolver;
    }




        /*************************************************
         *************************************************
         *
         *   CONVERSION DE LISTA REGIONES A RCC8
         *
         *************************************************
         *************************************************/

        public static ListaRelacionRCC Dibujo_a_RCC(ListaElementos _lE) {
                ListaRelacionRCC devolver = new ListaRelacionRCC();

                int longitud = _lE.size();
                for(int i=0;i<longitud−1;i++) {
                        for(int j=i+1;j<longitud;j++) {
                                Elemento e1 = (Elemento)_lE.get(i);
                                Elemento e2 = (Elemento)_lE.get(j);

                                Relacion rr = new Relacion(
                                                RazonG_RCC8.PosicionRelativa(e1, e2),
                                                (e1.getNombre()),
                                                (e2.getNombre()));

                                devolver.add(rr);
```

```
165                             }
166                     }
167
168
169                     return devolver;
170             }
171
172
173
174
175     }
```

```
1  package es.us.ccia.dummy.razonGrafico;
2
3  import java.awt.geom.Rectangle2D;
4  import java.util.Iterator;
5
6  import es.us.ccia.dummy.teoria.regiones.Elemento;
7  import es.us.ccia.dummy.teoria.regiones.Punto;
8  import es.us.ccia.dummy.teoria.regiones.Region;
9  import es.us.ccia.dummy.teoria.relaciones.ListaRelacionRCC;
10 import es.us.ccia.dummy.teoria.relaciones.Relacion;
11 import es.us.ccia.rcc.RCC;
12
13 public class RazonGBase {
14
15
16
17
18         /**
19          * Metodo que nos informa si el conjunto r es subconjunto de Ã©ste
20          * @param r Rectangulo2D que representa al subconjunto
21          * @return True si r es subconjunto de este conjunto, False en caso contrario
22          */
23         public static boolean esIgual(Region r1,Region r2) {
24                 Rectangle2D a = new Rectangle2D.Double(r1.getX1(),r1.getY1(),r1.getAncho(),r1.
   getAlto());
25         Rectangle2D b = new Rectangle2D.Double(r2.getX1(),r2.getY1(),r2.getAncho(),r2.getAlto(
   ));
26
27         return b.contains(a) && a.contains(b);
28         }
29
30         /**
31          * MÃ©todo que nos informa si el conjunto r es subconjunto de Ã©ste
32          * @param r Rectangulo2D que representa al subconjunto
33          * @return True si r es subconjunto de este conjunto, False en caso contrario
34          */
35         public static boolean esSubconjuntoDe(Region r1, Region r2) {
36
37         Rectangle2D a = new Rectangle2D.Double(r1.getX1(),r1.getY1(),r1.getAncho(),r1.getAlto(
   ));
38         Rectangle2D b = new Rectangle2D.Double(r2.getX1(),r2.getY1(),r2.getAncho(),r2.getAlto(
   ));
39
40         return b.contains(a);
41         }
42
43         /**
44      * Metodo que nos informa si dos conjuntos interseccionan
45      * @param r Rectangulo2D que representa al otro conjunto
46      * @return True si r es subconjunto de este conjunto, False en caso contrario
47      */
48         public static boolean esInterseccion(Region r1, Region r2) {
49
50         Rectangle2D a = new Rectangle2D.Double(r1.getX1(),r1.getY1(),r1.getAncho(),r1.getAlto(
   ));
51         Rectangle2D b = new Rectangle2D.Double(r2.getX1(),r2.getY1(),r2.getAncho(),r2.getAlto(
   ));
52
53         return a.intersects(b);
54     }
55
56     /**
57      * Metodo que nos informa si dos conjuntos son tangentes en algÃºn lado
58      * @param r Rectangulo2D que representa al otro conjunto
59      * @return True si r es tangente a este conjunto, False en caso contrario
60      */
61         public static boolean esTangente(Region r1,Region r2) {
62         return
63                 r1.getX1() == r2.getX1() ||
64                 r1.getX1() == r2.getX2() ||
65
66                 r1.getX2() == r2.getX1() ||
67                 r1.getX2() == r2.getX2() ||
68
69                 r1.getY1() == r2.getY1() ||
70                 r1.getY1() == r2.getY2() ||
71
72                 r1.getY2() == r2.getY1() ||
73                 r1.getY2() == r2.getY2();
74         }
75
76         public static boolean externaConectadas(Region r1,Region r2) {
```

```
77                      return
78          (
79                      ( r1.getX1() == r2.getX2() || r1.getX2() == r2.getX1() ) &&
80                              (r2.getY1() >= r1.getY1() − r2.getAlto()) &&
81                              (r2.getY1() <= r1.getY2())
82
83              ) ||
84
85              (
86                      ( r1.getY1() == r2.getY2() || r1.getY2() == r2.getY1() ) &&
87                              (r2.getX1() >= r1.getX1() − r2.getAncho()) &&
88                              (r2.getX1() <= r1.getX2())
89
90                      );
91
92
93
94
95
96
97
98
99              }
100
101
102 //
103 //      public int relacionAnterior(FCAElement a, FCAElement b){
104 //              int devolver = RCC.NODEF;
105 //              ListaRelacionRCC tope = PilaRelacRCC.pop();
106 //
107 //              Iterator it = PilaRelacRCC.peek().iterator();
108 //              while (it.hasNext()) {
109 //                      Relacion r = (Relacion)it.next();
110 //                      if ((r.getA().equalsIgnoreCase(a.getNombre())) &&
111 //                                      (r.getB().equalsIgnoreCase(b.getNombre()))) {
112 //                              devolver = r.getTipo();
113 //                      } else if ((r.getA().equalsIgnoreCase(b.getNombre()) &&
114 //                                      (r.getB().equalsIgnoreCase(a.getNombre())))) {
115 //                              devolver = RCC.reflexiva(r.getTipo());
116 //                      }
117 //              }
118 //              PilaRelacRCC.push(tope);
119 //              return devolver;
120 //      }
121
122
123 }
```

```
1  package es.us.ccia.dummy.razonGrafico;
2
3  import java.awt.geom.Rectangle2D;
4
5  import es.us.ccia.dummy.teoria.*;
6  import es.us.ccia.dummy.teoria.regiones.*;
7  import es.us.ccia.dummy.teoria.relaciones.ListaRelacionRCC;
8  import es.us.ccia.dummy.teoria.relaciones.Relacion;
9  import es.us.ccia.rcc.RCC;
10
11 public class RazonG_RCC5 extends RazonGBase  {
12
13         /**
14          * MÃ©todo generico para el calculo de las posiciones relativas de
15          * dos regiones.
16          * @param e1 Region 1
17          * @param e2 Region 2
18          * @return nos devuelve un entero que sera la constante perteneciente
19          * a la posicion relativa de los dos elementos.
20          */
21         public static int PosicionRelativa(Elemento e1, Elemento e2){
22                 int devolver = -1;
23
24                 if (e1.getClass().getSimpleName().equalsIgnoreCase("Region")) {
25                         if (e2.getClass().getSimpleName().equalsIgnoreCase("Region")) {
26                                 devolver = PosicionRelativaRR((Region)e1,(Region) e2);
27                         } else if (e2.getClass().getSimpleName().equalsIgnoreCase("Punto")) {
28                                 devolver = PosicionRelativaRP((Region)e1,(Punto) e2);
29                         }
30                 } else if (e1.getClass().getSimpleName().equalsIgnoreCase("Punto")) {
31                         if (e2.getClass().getSimpleName().equalsIgnoreCase("Region")) {
32                                 devolver = PosicionRelativaRP((Region)e2,(Punto) e1);
33                         } else if (e2.getClass().getSimpleName().equalsIgnoreCase("Punto")) {
34                                 devolver = PosicionRelativaPP((Punto)e1,(Punto) e2);
35                         }
36                 }
37                 return devolver;
38         }
39
40
41         /*****************************************************
42          *****************************************************
43          *
44          *   POSICIONES RELATIVAS DE REGIONES !!!!
45          *
46          *****************************************************
47          *****************************************************/
48
49
50         public static int PosicionRelativaRR(Region r1, Region r2){
51                 int devolver = RCC.NODEF;
52
53 //              Salida.Imprimirln("Region - Region");
54
55
56                 if (esIgual(r1,r2)) {
57                         devolver = RCC.EQ5;
58
59                 } else if (!esInterseccion(r1,r2)){
60                         devolver = RCC.DR;
61
62                 } else if (esSubconjuntoDe(r1,r2)){
63                                 devolver = RCC.PP;
64
65                 } else if (esSubconjuntoDe(r2,r1)) {
66                         devolver = RCC.PPi;
67
68                 } else {
69                         devolver = RCC.PO;
70                 }
71
72         return devolver;
73         }
74
75         public static int PosicionRelativaRP(Region r1, Punto p){
76                 int devolver = RCC.NODEF;
77
78                 //Salida.Imprimirln("Region - Punto");
79
80                 Rectangle2D r = new Rectangle2D.Double(
81                                 r1.getX1(), r1.getY1(),
82                                 r1.getX2() - r1.getX1(),
```

```
                                r1.getY2() - r1.getY1());
                if (r.contains(p.getX1(), p.getY1())) {
                        if (    (r1.getX1() == p.getX1()) ||
                                        (r1.getX2() == p.getX1()) ||
                                        (r1.getY1() == p.getY1()) ||
                                        (r1.getY2() == p.getY1())
                        )
                        {
                                devolver = RCC.FRONT_P;
                        } else {
                                devolver = RCC.IN_P;
                        }
                } else {
                        devolver = RCC.OUT_P;
                }



                return devolver;
        }

    /**
     * Posicion relativa de dos puntos
     *
     * @param p1 Punto 1
     * @param p2 Punto 2
     * @return Devuelve un entero para con la posicion relativa
     */
        public static int PosicionRelativaPP(Punto p1, Punto p2) {
        int devolver = RCC.DIFER_P;

        if (p1.getX1() == p2.getX1() && p1.getY1() == p2.getY1()) {
                devolver = RCC.IGUAL_P;
        }
        return devolver;
    }




        /*****************************************************
         *****************************************************
         *
         *   CONVERSION DE LISTA REGIONES A RCC8
         *
         *****************************************************
         *****************************************************/

        public static ListaRelacionRCC Dibujo_a_RCC(ListaElementos _lE) {
                ListaRelacionRCC devolver = new ListaRelacionRCC();

                int longitud = _lE.size();
                for(int i=0;i<longitud-1;i++) {
                        for(int j=i+1;j<longitud;j++) {
                                Elemento e1 = (Elemento)_lE.get(i);
                                Elemento e2 = (Elemento)_lE.get(j);

                                Relacion rr = new Relacion(
                                                RazonG_RCC5.PosicionRelativa(e1, e2),
                                                (e1.getNombre()),
                                                (e2.getNombre()));

                                devolver.add(rr);
                        }
                }

                return devolver;
        }


//  Esto para borrar... si no hace falta en otro lado.
//      private static FCAElement convertirElementoFCA(Elemento _e) {
//              FCAElement devolver = null;
//
//              if (_e.getTipo() == Elemento.PUNTO) {
//                      devolver = Contexto.getObject(_e.getNombre());
//              } else if (_e.getTipo() == Elemento.REGION) {
//                      devolver = Contexto.getAttribute(_e.getNombre());
//              }
//
//
//              return devolver;
```

```
164 //        }
165 //
166
167
168 }
```