➢ Class Aliada

```
1  /*****************************************************************************************************
2   * Clase Aliada.                                                                                     *
3   * ------------------------------------------------------------------------------------------------- *
4   * En esta clase se definen los atributos de la nave aliada, movimiento y se trata el manejo         *
5   * de la misma.                                                                                      *
6   * <<< Borja DelgadoAngulo >>>                                                                       *
7   *****************************************************************************************************/
8
9   // Lista de bibliotecas.
10  import java.awt.Image;
11  import java.awt.Rectangle;
12  import java.awt.event.KeyEvent;
13
14  import java.util.ArrayList;
15
16  import javax.swing.ImageIcon;
17
18  // Clase Aliada.
19  public class Aliada {
20       // Atributos.
21       // Posicion.
22       public int x, y;
23       // Avance.
24       public int dx, dy;
25       // Medidas del objeto.
26       private int width, height;
27       //Variable para posicionar el origen del disparo laser en la nave.
28       private final int naveLaser = 20;
29
30       //Variable para la imagen de la Nave Aliada.
31       public static Image image;
32       //Variable para el array de lasers creados.
33       private ArrayList lasers;
34
35       // Constructor de la clase RType.
36       public Aliada() {
37            // Se recibe la imagen de la nave Aliada de una imagen externa
     .
38            ImageIcon ii = new ImageIcon(this.getClass().getResource("alia da.png"));
39            image = ii.getImage();
40            // Se obtienen las medidas de la imagen de la nave Aliada.
41            width = image.getWidth(null);
42            height = image.getHeight(null);
43
44            // Se define un array para los disparos laser.
45            lasers = new ArrayList();
46
47            // Se define el origen de la nave aliada al comienzo del juego
       .
48            x = 40;
49            y = 250;
50       }
51
52       // Metodo para el movimiento de la nave.
53       public void move() {
54            // La nave aliada avanza una posicion.
55            x += dx;
56            y += dy;
57
58            // Se controla que la nave aliada no se salga de los límites q ue se indican.
```

```
59            if (x <= 10 && x > -1000) {
60                 x = 10;
61            }
62
63            if (x >= 740) {
64                 x = 740;
65            }
66
67            if (y <= 50 && y > -1000) {
68                 y = 50;
69            }
70
71            if (y >= 500) {
72                 y = 500;
73            }
74       }
75
76       // Metodo para obtener la posición de la nave en ese momento en el eje X.
77       public int getX() {
78            return x;
79       }
80
81       // Metodo para obtener la posición de la nave en ese momento en el eje Y.
82       public int getY() {
83            return y;
84       }
85
86       // Metodo para obtener la imagen de la nave Aliada.
87       public Image getImage() {
88            return image;
89       }
90
91       // Metodo para obtener la posicion de los lasers en curso.
92       public ArrayList getLasers() {
93            return lasers;
94       }
95
96       // Metodo para obtener los limites de la imagen de la nave aliada.
97       public Rectangle getBounds() {
98            return new Rectangle(x, y, width-40, height-40);
99       }
100
101      // Metodo para manejar la acción de pulsar una tecla.
102      public void keyPressed(KeyEvent e) {
103
104           int key = e.getKeyCode();
105
106           // Tecla Disparo.
107           if (key == KeyEvent.VK_SPACE) {
108                disparar();
109           }
110
111           // Tecla movimiento: Izquierda.
112           if (key == KeyEvent.VK_A) {
113                dx = -1;
114           }
115
116           // Tecla movimiento: Derecha.
117           if (key == KeyEvent.VK_D) {
118                dx = 1;
```

```
119          }
120
121          // Tecla movimiento: Arriba.
122          if (key == KeyEvent.VK_W) {
123               dy = -1;
124          }
125
126          // Tecla movimiento: Abajo.
127          if (key == KeyEvent.VK_S) {
128               dy = 1;
129          }
130
131          // Tecla nueva partida (Se lanza nueva ventana de seleccion de juego).
132          if (key == KeyEvent.VK_ENTER && PantallaJuego.desactEnter) {
133               PantallaJuego.desactEnter = false;
134
135               RType.finPartida();
136          }
137
138          // Tecla salir de la aplicacion.
139          if (key == KeyEvent.VK_ESCAPE) {
140               System.exit(0);
141          }
142     }
143
144     // Método para crear un nuevo laser y posicionar el origen de este en la nave aliada.
145     public void disparar() {
146          lasers.add(new Laser(x + naveLaser+5, y + naveLaser));
147     }
148
149     // Método para manejar la acción de soltar una tecla y detener el movimiento de la nave
    aliada.
150     public void keyReleased(KeyEvent e) {
151          int key = e.getKeyCode();
152
153          if (key == KeyEvent.VK_A) {
154               dx = 0;
155          }
156
157          if (key == KeyEvent.VK_D) {
158               dx = 0;
159          }
160
161          if (key == KeyEvent.VK_W) {
162               dy = 0;
163          }
164
165          if (key == KeyEvent.VK_S) {
166               dy = 0;
167          }
168     }
169 }
```

➢ Class Alienigena

```
1   /*********************************************************************************************
2    * Clase Alienígena.                                                                          *
3   * ------------------------------------------------------------------------------------------ *
4   * En esta clase se definen los atributos de las diferentes naves alienigenas, movimiento y se   *
5   * trata el manejo estas.                                                                     *
6   * <<< Borja Delgado Angulo >>>                                                               *
7    *********************************************************************************************/
8
9   // Lista de bibliotecas.
10  import java.awt.Image;
11  import java.awt.Rectangle;
12
13  import javax.swing.ImageIcon;
14
15  // Clase Alienígena.
16  public class Alienigena {
17       // Atributos.
18       // Posicion.
19       public static int xA, yA, xB, yB;
20       // Avance.
21       public static int dx;
22       public static int dy;
23       // Medidas del objeto.
24       public int widthA, heightA, widthB, heightB;
25       // Guardar posicion en caso de colision.
26       public static int avanzaA;
27       public static int avanzaB;
28
29       // Valida si una nave alienigena ha sido eliminada.
30       public boolean      muertoA;
31       public boolean muertoB;
32       // Valida si se debe crear una nueva nave alienigena, para no suma r al contador cuando lo
    sea
33       // necesario.
34       public boolean nuevoA = true;
35       public boolean nuevoB = true;
36
37       public static boolean seMueve = true;
38
39       //Variable para las imagenes de las naves alienigenas.
40       public static Image imageA;
41       private Image imageAlienA;
42       public static Image imageB;
43       private Image imageAlienB;
44
45       // Constructor de la clase Alienigena.
46       public Alienigena() {
47            // Se recibe la imagen de la nave Alienigena tipo A de una ima gen externa.
48            ImageIcon ii = new ImageIcon(this.getClass().getResource("spac eshipA.gif"));
49            // Se obtienen las medidas de la imagen de la nave Alienigena tipo A.
50            imageA = ii.getImage();
51            imageAlienA = ii.getImage();
52
53            // Se recibe la imagen de la nave Alienigena tipo B de una ima gen externa.
54            ImageIcon iii = new ImageIcon(this.getClass().getResource("spa ceshipB.gif"));
55            imageB = iii.getImage();
56            imageAlienB = iii.getImage();
57            // Se obtienen las medidas de las imagenes de las naves Alieni genas tipo A y B.
58            widthA = imageA.getWidth(null);
59            heightA = imageA.getHeight(null);
60            widthB = imageB.getWidth(null);
```

```
61            heightB = imageB.getHeight(null);
62
63            // Se define el origen de las naves alienigenas al comienzo de l juego.
64            xA = 800;
65            yA = 100;
66            xB = 800;
67            yB = 300;
68       }
69
70      // Metodo para el movimiento de las Naves tipo A.
71      public void moveA() {
72           // Hasta que no comience una nueva partida, la nave alienigena
           A se queda sin moverse.
73           if (seMueve == true) {
74                 // La nave alienigena tipo B avanza una posicion.
75                 xA -= dx+1;
76
77                 // Si la nave alienigena tipo A llega un poco mas lejos de l limite izquierdo de
   la
78                 // pantalla, esta aparece de nuevo por el lado derecho en una posicion diferente
   en el
79                 // eje "y".
80                 if ((getXA() > -500) && (getXA() <= -50)) {
81                      xA = 1000;
82                      yA = yA+182;
83                      if (yA >= 510 || yA <= 50) {
84                           yA = 100;
85                      }
86
87                      imageA = imageAlienA;
88                 }
89
90                 // Si la nave alienigena tipo A es eliminada, esta avanza unas posiciones en el
   eje "x" e "y",
91                 // para dar tiempo a que se cargue la imagen de explosion.
92                 // Despues, esta aparece de nuevo por el lado derecho en u na posicion diferente
   en el eje "y",
93                 // con la imagen recargada de nuevo, como si se tratase de una nueva nave
    alienigena.
94                 if (muertoA) {
95                      if (xA < avanzaA-300) {
96                           xA += 1000;
97                           yA += 182;
98                           if (yA >= 510 && yA <= 50) {
99                                yB = yB+92;
100                          }else yB = 100;
101
102                          imageA = imageAlienA;
103                          muertoA = false;
104                          nuevoA = true;
105                     }
106                }
107                }
108     }
109
110
111     // Metodo para devolver la posicion en la que la nave alienigena ti po A es eliminada.
112     public void muertoA() {
113          avanzaA = xA;
114          muertoA = true;
115     }
116
117     // Obtenemos la posición de la nave alienigena tipo A en ese moment o en el eje X.
```

```
118      public int getXA() {
119           return xA;
120      }
121
122     // Obtenemos la posición de la nave alienigena tipo A en ese moment o en el eje Y.
123      public int getYA() {
124           return yA;
125      }
126
127      // Metodo para obtener la imagen de la nave alienigena tipo A.
128       public Image getImageA() {
129            return imageA;
130      }
131
132      // Metodo para obtener los limites de la imagen de la nave alienig ena tipo A.
133      public Rectangle getBoundsA() {
134           return new Rectangle(xA, yA, widthA-10, heightA-10);
135      }
136
137      // Metodo para el movimiento de las Naves tipo B.
138     public void moveB() {
139          // Hasta que no comience una nueva partida, la nave alienigena
           B se queda sin moverse.
140           if (seMueve == true) {
141                // Se controla que si la nave allienigena tipo B llega a u no de los limites de
   la pantalla
142                // en el eje "y", esta cambia el sentido.
143                if (getYB() >= 510) {
144                     dy = -2;
145                }
146
147                if (getYB() <= 50) {
148                     dy = 2;
149                }
150
151                // La nave alienigena tipo B avanza una posicion.
152                xB -= dx;
153                yB += dy;
154
155                // Si la nave alienigena tipo B llega un poco mas lejos de l limite izquierdo de
   la
156                // pantalla, esta aparece de nuevo por el lado derecho en una posicion diferente
   en el
157                // eje "y".
158                if ((getXB() > -500) &&         (getXB() <= -50)) {
159                     xB = 1000;
160                     if (yB <= 550) {
161                          yB = yB+92;
162                     }else yB = 10;
163
164                     imageB = imageAlienB;
165                }
166
167                // Si la nave alienigena tipo B es eliminada, esta avanza unas posiciones en el
   eje "x",
168                // para dar tiempo a que se cargue la imagen de explosion.
169                // Despues, esta aparece de nuevo por el lado derecho en u na posicion diferente
   en el
170                // eje "y", con la imagen recargada de nuevo, como si se t ratase de una nueva
   nave alienigena.
171                if (muertoB) {
172                     dy = 0;
173                     if (xB < avanzaB-150) {
```

```
                        xB += 1000;
                        dy = 2;
                        if (yB <= 550) {
                             yB = yB+92;
                        }else yB = 10;

                        imageB = imageAlienB;
                        muertoB = false;
                        nuevoB = true;
                   }
              }
         }
    }

    // Metodo para devolver la posicion en la que la nave alienigena ti po B es eliminada.
    public void muertoB() {
         avanzaB = xB;
         muertoB = true;
    }

    // Obtenemos la posición de la nave alienigena tipo B en ese moment o en el eje X.
    public int getXB() {
         return xB;
    }

    // Obtenemos la posición de la nave alienigena tipo B en ese moment o en el eje Y.
    public int getYB() {
         return yB;
    }

    // Metodo para obtener la imagen de la nave alienigena tipo B.
    public Image getImageB() {
         return imageB;
    }

    // Metodo para obtener los limites de la imagen de la nave alienige na tipo B.
    public Rectangle getBoundsB() {
         return new Rectangle(xB, yB, widthB-10, heightB);
     }
}
```

Class JuegoNuevo

```
/*********************************************************************************************
 *                                Clase                                JuegoNuevo.
 *
 * ------------------------------------------------------------------------------------------ *
 * Se definen los detalles de la ventana de juego y se carga el limite de muertes alienigenas
 *
 *       segun        el        modo        de        juego        seleccionado.
 *
 *          <<<          Borja          Delgado          Angulo          >>>
 *
 *********************************************************************************************/

// Lista de bibliotecas.
import javax.swing.JFrame;

// Clase JuegoNuevo.
public class JuegoNuevo extends JFrame {
    //Atributos.
    public static int modoJuego;

    // Constructor de la clase JuegoNuevo.
    public JuegoNuevo() {
        add(new PantallaJuego());

        // Se activa la opcion de salir de la aplicacion al pulsar el boton de salir(x) de la ventana.
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        // Tamaño de la ventana de juego.
        setSize(800, 600);
        // Se establece la posicion de la ventana en el centro.
        setLocationRelativeTo(null);
        // Se desactiva la opcion de modificar el tamaño de la ventana.
        setResizable(false);
        // Se activa la visibilidad de la ventana.
        setVisible(true);

        // Dependiendo del valor pasado en la clase RType, se cargan l os titulos y limite de navesa
        // eliminar correspondientes.
        switch (modoJuego) {
            case 1:
                setTitle("R - Type <<< Modo Facil >>>");
                PantallaJuego.maxMuertes = 10;
                break;
            case 2:
                setTitle("R - Type <<< Modo Normal >>>");
                PantallaJuego.maxMuertes = 15;
                break;
            case 3:
                setTitle("R - Type <<< Modo Complicado >>>");
                PantallaJuego.maxMuertes = 20;
                break;
            case 4:
                setTitle("R - Type <<< Modo IMPOSIBLE >>>");
                PantallaJuego.maxMuertes = 30;
                break;
        }
    }

    // Método para cargar una partida nueva.
```

```
55      public static void main(String[] args) {
56           JuegoNuevo juegoNuevo = new JuegoNuevo();
57      }
58  }
```

Class Laser

```
1  /*********************************************************************************************
2  * Clase Laser.                                                                               *
3  * ------------------------------------------------------------------------------------------ *
4  * En esta clase se definen los atributos de los lasers, movimiento y se trata el manejo de   *
   * los mismos.                                                                                *
6  * <<< Borja Delgado Angulo >>>                                                               *
7    *********************************************************************************************/
8
9  // Lista de bibliotecas.
10 import java.awt.Image;
11 import java.awt.Rectangle;
12
13 import javax.swing.ImageIcon;
14
15 // Clase Laser.
16 public class Laser {
17       // Atributos.
18       // Posicion.
19       private int x, y;
20       // Medidas del objeto.
21       public int width, height;
22       // Velocidad del disparo laser.
23       private final int velocidadLaser = 2;
24
25       //Variable para la imagen del laser.
26       private Image image;
27
28       //Variable para validar si el laser es visible o no.
29       boolean visible;
30
31       // Constructor de la clase Laser.
32       public Laser(int x, int y) {
33            // Se recibe la imagen del laser de una imagen externa.
34            ImageIcon ii = new ImageIcon(this.getClass().getResource("lase r.jpg"));

35            image = ii.getImage();
36            // Se obtienen las medidas de la imagen del laser.
37            width = image.getWidth(null);
38            height = image.getHeight(null);
39
40            // Se define a visible por defecto.
41            visible = true;
42
43            // Se define la posicion del laser la que tiene en ese momento
      .
              this.x = x;
44
```

```
45            this.y = y;
46        }
47
48        // Método para el movimiento del láser.
49        public void move() {
50            // El laser avanza una posicion.
51            x += velocidadLaser;
52
53            //Si el laser pasa del limite izquierdo de la pantalla, este d eja de verse.
54            if (x > 810){
55                visible = false;
56            }
57        }
58
59        // Metodo para obtener la posición del laser en ese momento en el eje X.
60        public int getX() {
61            return x;
62        }
63
64        // Metodo para obtener la posición del laser en ese momento en el eje Y.
65        public int getY() {
66            return y;
67        }
68
69        // Metodo para obtener la imagen del laser.
70        public Image getImage() {
71            return image;
72        }
73
74        // Metodo para devolver visible al laser.
75        public boolean isVisible() {
76            return visible;
77        }
78
79        // Metodo para obtener los limites de la imagen del laser.
80        public Rectangle getBounds() {
81            return new Rectangle(x, y, width-5, height-5);
82        }
83    }
```

Class Pantalla juego

```
1   /**********************************************************************************************
2    * Clase PantallaJuego.                                                                       *
3    * -------------------------------------------------------------------------------------------*
4    * En esta clase se dibuja la imágen de fondo del juego, las imágenes de las naves aliada y   *
5    * alienígenas, los disparos láser y el manejo de estos segun las cond iciones que se indican en
     * cada caso.                                                                                 *
6    * <<< Borja Delgado Angulo >>>                                                               *
8    **********************************************************************************************/
9
10  // Lista de bibliotecas.
11  import java.awt.Graphics;
12  import java.awt.Graphics2D;
13  import java.awt.Image;
14  import java.awt.Rectangle;
15  import java.awt.Color;
16  import java.awt.Toolkit;
17  import java.awt.event.ActionEvent;
18  import java.awt.event.ActionListener;
19  import java.awt.event.KeyAdapter;
20  import java.awt.event.KeyEvent;
21
22  import java.util.ArrayList;
23
24  import javax.swing.ImageIcon;
25  import javax.swing.JPanel;
26  import javax.swing.Timer;
27
28  // Clase PantallaJuego.
29  public class PantallaJuego extends JPanel implements ActionListener {
30      // Atributos.
31      // Variables para las imagenes del fondo y del contador de naves a lienigenas eliminadas.
32      Image imagen, contNaves;
33      // Variable timer para manejar cada momento de la ejecucion.
34      private Timer timer;
35      // Variable para utilizar la nave aliada.
36      private Aliada aliada;
37      // Variable para utilizar las naves alienigenas.
38      private Alienigena alienigenaA;
39      private Alienigena alienigenaB;
40
41      // Variables rectangulo para manejar colisiones.
42      Rectangle rAliada;
43      Rectangle rAlienigenaA;
44      Rectangle rAlienigenaB;
```

```
45
46      // Variable para validar si el laser colisiona con una nave alieni gena.
47      public static boolean colision;
48
49      // Variable contador de naves alienigenas eliminadas.
50      public int muertes;
51      // Variable para controlas el maximo numero de naves a eliminar se gun modo de juego
    seleccionado.
52      public static int maxMuertes;
53
54      // Variable para modificar la imagen contador de naves alienigenas eliminadas.
55      public static String alienEliminados;
56
57      // Variable para validar si la tecla Enter puede pulsarse, solo en caso de fin de partida.
58      public static boolean desactEnter;
59
60      // Variable auxiliar de inicio de juego nuevo.
61      public static boolean vuelta2;
62
63      // Constructor de la clase PantallaJuego.
64      public PantallaJuego() {
65           // Se recibe la imagen del fondo de pantalla de una imagen ext erna.
66           ImageIcon ii = new ImageIcon(this.getClass().getResource("univ erse2.gif"));
67           imagen = ii.getImage();
68
69           // Se recibe la imagen de origen del contador de naves alienig enas eliminadas de una
70           // imagen externa.
71           ImageIcon iii = new ImageIcon(this.getClass().getResource("x0. jpg"));
72           contNaves = iii.getImage();
73
74           // Se añade la opcion de "escuchar" la tecla pulsada en cada m omento mediante la
75           // clase interna TAdapter.
76           addKeyListener(new TAdapter());
77           // Para que un objeto JPanel reciba las notificaciones del tec lado es necesario
    incluir la
78           // siguiente instrucción.
79           setFocusable(true);
80
81           // Se activa el color negro por defecto en el fondo de pantall a.
82           setBackground(Color.BLACK);
83
84           // Esta opcion dibuja primero en memoria, y luego dibuja todo junto en pantalla.
85           setDoubleBuffered(true);
```

```
86
87            // Se crea un nuevo objeto Nave Aliada.
88            aliada = new Aliada();
89            // Se crea un nuevo objeto alienigena tipo A.
90            alienigenaA = new Alienigena();
91            // Se crea un nuevo objeto alienigena tipo B.
92            alienigenaB = new Alienigena();
93
94            // Se inicializa el contador de muertes, y las variable auxili ares de validar tecla
    enter y
95            // de inicio de juego nuevo.
96            muertes = 0;
97            desactEnter = false;
98            vuelta2 = false;
99
100           // Se crea un nuevo timer para el manejo de cada momento de ej ecucion.
101           timer = new Timer(4, this);
102           timer.start();
103     }
104
105     // Método para dibujar en el JPanel todos los elementos que partic ipan en la ejecucion.
106     public void paint(Graphics g) {
107           super.paint(g);
108
109           Graphics2D g2d = (Graphics2D)g;
110
111           g2d.drawImage(imagen, 0, 0, null);
112           g2d.drawImage(contNaves, 650, 20, null);
113           g2d.drawImage(aliada.getImage(), aliada.getX(), aliada.getY(), this);
114           g2d.drawImage(alienigenaA.getImageA(), alienigenaA.getXA(), al ienigenaA.getYA(),
    this);
115           g2d.drawImage(alienigenaB.getImageB(), alienigenaB.getXB(), al ienigenaB.getYB(),
    this);
116
117           // Se crea un array con los lasers creados en el momento y se recorre, dibujandolos y
118           // controlando si colisionan con las naves alienigenas.
119           // Mediante el parametro rectangulo, manejamos los limites de los disparos laser y
    las
120           // naves alienigenas y si estos "intersectan", se produce la e xplosion de la nave
121           // alienigena. Esto da lugar al sumatorio del contador de nave s eliminadas.
122           ArrayList lsr = aliada.getLasers();
123           for (int i = 0; i < lsr.size(); i++ ) {
124                 Laser ls = (Laser) lsr.get(i);
125                 g2d.drawImage(ls.getImage(), ls.getX(), ls.getY(), this);
126
127                 Rectangle rLaser = ls.getBounds();
128                 Rectangle rAlienigenaA = alienigenaA.getBoundsA();
```

```
129
130             if (rLaser.intersects(rAlienigenaA) && alienigenaA.nuevoA) {
131                     ImageIcon ii = new ImageIcon(this.getClass().getResour
    ce("explosionNAVE.gif"));
132                     Alienigena.imageA = ii.getImage();
133
134                     colision = true;
135
136                     alienigenaA.muertoA();
137
138                     muertes +=1;
139
140                     eliminados();
141
142                     ImageIcon iiii = new ImageIcon(this.getClass().getReso
    urce(alienEliminados));
143                     contNaves = iiii.getImage();
144
145                     alienigenaA.nuevoA = false;
146                 }
147
148                 Rectangle rAlienigenaB = alienigenaB.getBoundsB();
149
150                 if (rLaser.intersects(rAlienigenaB) && alienigenaB.nuevoB)
     {
151                     ImageIcon iii = new ImageIcon(this.getClass().getResou
    rce("explosionNAVE.gif"));
152                     Alienigena.imageB = iii.getImage();
153
154                     colision = true;
155
156                     alienigenaB.muertoB();
157
158                     muertes +=1;
159
160                     eliminados();
161
162                     ImageIcon iiiii = new ImageIcon(this.getClass().getRes
    ource(alienEliminados));
163                     contNaves = iiiii.getImage();
164
165                     alienigenaB.nuevoB = false;
166                 }
167             }
168
169             // Se manejan las colisiones entre la nave Aliada y naves alie nigenas.
170             // Si estas "intersectan", se produce el final de partida, se carga una nueva imagen
    de fondo
171             // indicando "final de partida" y la nave aliada desaparece de la pantalla.
172             Rectangle rAliada = aliada.getBounds();
173             Rectangle rAlienigenaA = alienigenaA.getBoundsA();
```

```
174          Rectangle rAlienigenaB = alienigenaB.getBoundsB();
175
176          if (rAliada.intersects(rAlienigenaA)) {
177               ImageIcon i = new ImageIcon(this.getClass().getResource("e xplosionNAVE.gif"));
178               Alienigena.imageA = i.getImage();
179               ImageIcon ii = new ImageIcon(this.getClass().getResource(" naveExpl.gif"));
180               Aliada.image = ii.getImage();
181
182               ImageIcon iii = new ImageIcon(this.getClass().getResource( "gameover.jpg"));
183               imagen = iii.getImage();
184               // Se desactiva la imagen contador de naves eliminadas.
185               contNaves = null;
186               aliada.x = -9000;
187               aliada.y = -9000;
188
189               desactEnter = true;
190          }
191
192          if (rAliada.intersects(rAlienigenaB)) {
193               ImageIcon iii = new ImageIcon(this.getClass().getResource( "explosionNAVE.gif"));
194               Alienigena.imageB = iii.getImage();
195
196               ImageIcon i = new ImageIcon(this.getClass().getResource("g ameover.jpg"));
197               imagen = i.getImage();
198               contNaves = null;
199               aliada.x = -9000;
200               aliada.y = -9000;
201
202               desactEnter = true;
203          }
204
205          Toolkit.getDefaultToolkit().sync();
206          g.dispose();
207     }
208
209     // Método para manejar el avance de los elementos en ejecucion.
210     public void actionPerformed(ActionEvent e) {
211          if (vuelta2 == true) {
212               muertes = 0;
213               aliada.x = 40;
214               aliada.y = 250;
215
216               ImageIcon ii = new ImageIcon(this.getClass().getResource(" universe2.gif"));
217               imagen = ii.getImage();
218
219               ImageIcon iii = new ImageIcon(this.getClass().getResource( "aliada.png"));
```

```
220                 Aliada.image = iii.getImage();
221
222                 alienEliminados = "x0.jpg";
223                 ImageIcon iiii = new ImageIcon(this.getClass().getResource
        (alienEliminados));
224                 contNaves = iiii.getImage();
225
226                 vuelta2 = false;
227            }
228
229            ArrayList lsr = aliada.getLasers();
230            // Mediante la llamada al metodo isVisible de la clase laser, se define si el laser
    avanza una
231            // posicion o si por el contrario debe desaparecer de la pantalla.
232            for (int i = 0; i < lsr.size(); i++) {
233                 Laser ls = (Laser) lsr.get(i);
234                 if (ls.isVisible())
235                      ls.move();
236                 else lsr.remove(i);
237
238                 // El disparo laser desaparece cuando colisiona con una na ve alienigena, sino
    avanza una
239                 // posicion.
240                 if (colision == false) {
241                      ls.move();
242                 }
243
244                 if (colision == true) {
245                      lsr.remove(i);
246                      colision = false;
247                 }
248            }
249
250            // La nave aliada avanza una posicion.
251            aliada.move();
252            // La nave alienigena tipo A avanza una posicion.
253            alienigenaA.moveA();
254            // La nave alienigena tipo B avanza una posicion.
255            alienigenaB.moveB();
256            // Se llama a este metodo para volver a pintar los elementos e n su nueva posicion a
    cada golpe
257            // de timer.
258            repaint();
259       }
260
261       // Metodo para actualizar la imagen contador de naves eliminadas s egun el numero de naves
262       // alienigenas eliminadas.
263       // Si se alcanza el limite indicado en cada modo de juego, se carg a una nueva imagen de
    fondo
264       // indicando que la partida ha sido ganada y colocando las naves a lienigenas fuera de
    pantalla.
```

```
265     public void eliminados() {
266          switch (muertes) {
267               case 1:
268                    alienEliminados = "x1.jpg";
269                    break;
270               case 2:
271                    alienEliminados = "x2.jpg";
272                    break;
273               case 3:
274                    alienEliminados = "x3.jpg";
275                    break;
276               case 4:
277                    alienEliminados = "x4.jpg";
278                    break;
279               case 5:
280                    alienEliminados = "x5.jpg";
281                    break;
282               case 6:
283                    alienEliminados = "x6.jpg";
284                    break;
285               case 7:
286                    alienEliminados = "x7.jpg";
287                    break;
288               case 8:
289                    alienEliminados = "x8.jpg";
290                    break;
291               case 9:
292                    alienEliminados = "x9.jpg";
293                    break;
294               case 10:
295                    if (maxMuertes == 10 && JuegoNuevo.modoJuego == 1) {
296                         ImageIcon i = new ImageIcon(this.getClass().getResource("youwin.jpg"));
297                         imagen = i.getImage();
298                         Alienigena.xA=3000;
299                         Alienigena.xB=3000;
300                         Alienigena.seMueve = false;
301                         Alienigena.dx=0;
302                         Alienigena.dy=0;
303                         alienEliminados = "aliensad.jpg";
304
305                         desactEnter = true;
306                    }else
307                         alienEliminados = "x10.jpg";
308                    break;
309               case 11:
310                    alienEliminados = "x11.jpg";
311                    break;
312               case 12:
313                    alienEliminados = "x12.jpg";
314                    break;
315               case 13:
316                    alienEliminados = "x13.jpg";
```

```
317                 break;
318             case 14:
319                 alienEliminados = "x14.jpg";
320                 break;
321             case 15:
322                 if (maxMuertes == 15 && JuegoNuevo.modoJuego == 2) {
323                     ImageIcon i = new ImageIcon(this.getClass().getResource("youwin.jpg"));
324                     imagen = i.getImage();
325                     Alienigena.xA=3000;
326                     Alienigena.xB=3000;
327                     Alienigena.seMueve = false;
328                     Alienigena.dx=0;
329                     Alienigena.dy=0;
330                     alienEliminados = "aliensad.jpg";
331 
332                     desactEnter = true;
333                 }else
334                     alienEliminados = "x15.jpg";
335                 break;
336             case 16:
337                 alienEliminados = "x16.jpg";
338                 break;
339             case 17:
340                 alienEliminados = "x17.jpg";
341                 break;
342             case 18:
343                 alienEliminados = "x18.jpg";
344                 break;
345             case 19:
346                 alienEliminados = "x19.jpg";
347                 break;
348             case 20:
349                 if (maxMuertes == 20 && JuegoNuevo.modoJuego == 3) {
350                     ImageIcon i = new ImageIcon(this.getClass().getResurce("youwin.jpg"));
351                     imagen = i.getImage();
352                     Alienigena.xA=3000;
353                     Alienigena.xB=3000;
354                     Alienigena.seMueve = false;
355                     Alienigena.dx=0;
356                     Alienigena.dy=0;
357                     alienEliminados = "aliensad.jpg";
358 
359                     desactEnter = true;
360                 }else
361                     alienEliminados = "x20.jpg";
362                 break;
363             case 21:
364                 alienEliminados = "x21.jpg";
365                 break;
366             case 22:
367                 alienEliminados = "x22.jpg";
```

```
                    break;
               case 23:
                    alienEliminados = "x23.jpg";
                    break;
               case 24:
                    alienEliminados = "x24.jpg";
                    break;
               case 25:
                    alienEliminados = "x25.jpg";
                    break;
               case 26:
                    alienEliminados = "x26.jpg";
                    break;
               case 27:
                    alienEliminados = "x27.jpg";
                    break;
               case 28:
                    alienEliminados = "x28.jpg";
                    break;
               case 29:
                    alienEliminados = "x29.jpg";
                    break;
               case 30:
                    if (maxMuertes == 30 && JuegoNuevo.modoJuego == 4) {
                         ImageIcon i = new ImageIcon(this.getClass().getRes
    ource("youwin.jpg"));
                         imagen = i.getImage();
                         Alienigena.xA=3000;
                         Alienigena.xB=3000;
                         Alienigena.seMueve = false;
                         Alienigena.dx=0;
                         Alienigena.dy=0;
                         alienEliminados = "aliensad.jpg";

                         desactEnter = true;
                    }else
                         alienEliminados = "x30.jpg";
                    break;
          }
     }

     // Clase interna TAdapter, se encarga de manejar las acciones puls ar/soltar una tecla.
     private class TAdapter extends KeyAdapter {

          public void keyReleased(KeyEvent e) {
               aliada.keyReleased(e);
          }

          public void keyPressed(KeyEvent e) {
               aliada.keyPressed(e);
          }
     }

}
```

➢ Class Rtype

```
1  /*********************************************************************************************
2  * Clase RType.                                                                               *
3  * ------------------------------------------------------------------------------------------ *
4  * Clase principal de la aplicación java.                                                     *
5  * Se definen los detalles de la pantalla principal de la aplicacion (tamaño, títulos, modos de  *
6  * juego) y se carga la partida seleccionada por el jugador.                                  *
7  * <<< Borja Delgado Angulo >>>                                                               *
8   *********************************************************************************************/
9
10 // Lista de bibliotecas.
11 import javax.swing.JFrame;
12 import javax.swing.JButton;
13 import javax.swing.JLabel;
14 import javax.swing.ImageIcon;
15
16 import java.awt.FlowLayout;
17 import java.awt.event.ActionEvent;
18 import java.awt.event.ActionListener;
19
20 // Clase RType.
21 public class RType extends JFrame implements ActionListener {
22      // Atributos.
23      // Etiquetas para el texto y la imagen de la ventana.
24      JLabel texto, imag;
25      // Botones de seleccion de la ventana principal.
26      JButton botonFacil, botonNormal, botonComplicado, botonImposible, botonSalir;
27
28      public static boolean vuelta = true;
29
30      // Constructor de la clase RType.
31      public RType() {
32           setTitle("Practica POO 2013: R-Type");
33           // Tamaño de la ventana principal.
34           setBounds(500,200,250,450);
35           // Contenedor que pone los elementos en linea.
36           setLayout (new FlowLayout());
37           // Se desactiva la opcion de modificar el tamaño de la ventana
.
38           setResizable(false);
39
40           // Se definen los elementos que contendra el contenedor.
41           // Imagen de la ventana principal.
42           imag = new JLabel(new ImageIcon(getClass().getResource("imagen
Principal.jpg")));
```

```
43
44            // Etiqueta con texto seleccion.
45            texto = new JLabel("Seleccionar dificultad");
46
47            // Botones de seleccion.
48            botonFacil = new JButton("Facil (10 enemigos)");
49            botonNormal = new JButton("Normal (15 enemigos)");
50            botonComplicado = new JButton("Complicado (20 enemigos)");
51            botonImposible = new JButton("          IMPOSIBLE (30 enemigos)        ");
52            botonSalir = new JButton("Salir");
53
54            // Se añaden los elementos una vez definidos.
55            botonFacil.addActionListener (this);
56            botonNormal.addActionListener (this);
57            botonComplicado.addActionListener (this);
58            botonImposible.addActionListener (this);
59            botonSalir.addActionListener (this);
60
61            add(imag);
62
63            add(texto);
64
65            add(botonFacil);
66            add(botonNormal);
67            add(botonComplicado);
68            add(botonImposible);
69            add(botonSalir);
70
71            // Se activa la opcion de salir de la aplicacion al pulsar el boton de salir(x) de la
    ventana.
72            this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
73       }
74
75       // Metodo para manejar el boton pulsado.
76       // Segun la seleccion, se cargaran las opciones definidas por el m odo de juego (cantidad y
77       // velocidad de las naves alienigenas) o se sale de la aplicacion.
78       public void actionPerformed (ActionEvent evento) {
79            if (evento.getSource() == botonFacil) {
80                 JuegoNuevo.modoJuego = 1;
81                 Alienigena.xA = 800;
82                 Alienigena.yA = 100;
83                 Alienigena.xB = 800;
84                 Alienigena.yB = 300;
85                 Alienigena.dx = 1;
86                 Alienigena.dy = 2;
87                 Alienigena.seMueve = true;
88                 ImageIcon i = new ImageIcon(this.getClass().getResource("s paceshipB.gif"));
89                 Alienigena.imageB = i.getImage();
90                 ImageIcon ii = new ImageIcon(this.getClass().getResource(" spaceshipA.gif"));
91                 Alienigena.imageA = ii.getImage();
```

```
92                setVisible(false);
93
94                if (vuelta == true) {
95                     new JuegoNuevo();
96                     vuelta = false;
97                }else {
98                     PantallaJuego.vuelta2 = true;
99                }
100          }
101          else if (evento.getSource() == botonNormal) {
102               JuegoNuevo.modoJuego = 2;
103               Alienigena.xA = 800;
104               Alienigena.yA = 100;
105               Alienigena.xB = 800;
106               Alienigena.yB = 300;
107               Alienigena.dx = 2;
108               Alienigena.dy = 3;
109               Alienigena.seMueve = true;
110               ImageIcon i = new ImageIcon(this.getClass().getResource("s paceshipB.gif"));
111               Alienigena.imageB = i.getImage();
112               ImageIcon ii = new ImageIcon(this.getClass().getResource(" spaceshipA.gif"));
113               Alienigena.imageA = ii.getImage();
114               setVisible(false);
115
116               if (vuelta == true) {
117                    new JuegoNuevo();
118                    vuelta = false;
119               }else {
120                    PantallaJuego.vuelta2 = true;
121               }
122          }
123          else if (evento.getSource() == botonComplicado) {
124               JuegoNuevo.modoJuego = 3;
125               Alienigena.xA = 800;
126               Alienigena.yA = 100;
127               Alienigena.xB = 800;
128               Alienigena.yB = 300;
129               Alienigena.dx = 4;
130               Alienigena.dy = 4;
131               Alienigena.seMueve = true;
132               ImageIcon i = new ImageIcon(this.getClass().getResource("s paceshipB.gif"));
133               Alienigena.imageB = i.getImage();
134               ImageIcon ii = new ImageIcon(this.getClass().getResource(" spaceshipA.gif"));
135               Alienigena.imageA = ii.getImage();
136               setVisible(false);
137
138               if (vuelta == true) {
139                    new JuegoNuevo();
140                    vuelta = false;
```

```
141                 }else {
142                       PantallaJuego.vuelta2 = true;
143                 }
144           }
145           else if (evento.getSource() == botonImposible) {
146                 JuegoNuevo.modoJuego = 4;
147                 Alienigena.xA = 800;
148                 Alienigena.yA = 100;
149                 Alienigena.xB = 800;
150                 Alienigena.yB = 300;
151                 Alienigena.dx = 5;
152                 Alienigena.dy = 5;
153                 Alienigena.seMueve = true;
154                 ImageIcon i = new ImageIcon(this.getClass().getResource("s paceshipB.gif"));
155                 Alienigena.imageB = i.getImage();
156                 ImageIcon ii = new ImageIcon(this.getClass().getResource(" spaceshipA.gif"));
157                 Alienigena.imageA = ii.getImage();
158                 setVisible(false);
159
160                 if (vuelta == true) {
161                       new JuegoNuevo();
162                       vuelta = false;
163                 }else {
164                       PantallaJuego.vuelta2 = true;
165                 }
166           }
167           else if (evento.getSource() == botonSalir) {
168                 System.exit(0);
169           }
170     }
171
172     // Si la partida finaliza, ya sea por haber vencido o por haber si do derrotado, se carga
    una nueva
173     // pantalla de seleccion.
174     public static void finPartida() {
175           RType menu = new RType();
176           menu.setLocation(500, 200);
177           menu.setVisible(true);
178     }
179     public static void main(String[] args) {
180           finPartida();
181     }
182 }
183
184
```