“R-Type”

Práctica de Programación Orientada a Objetos
Junio 2013

Escuela Técnica Superior de Ingeniería Informática – UNED

Centro asociado: Madrid –Las Tablas

Alumno: Borja Delgado Angulo

Correo Electrónico UNED: bdelgado56@alumno.uned.es

Correo Electrónico Personal: borja.delan88@gmail.com

Teléfono: 669203560

Indice

1. Descripción de la práctica:

La práctica del presente curso va a estar basada en el legendario arcade “R-Type”. Esto nos servirá para estudiar y practicar los mecanismos de la Programación Orientada a Objetos y hacer uso de sus características gráficas.

➢ Historia del Juego

Según la descripción en Wikipedia, R-Type es un videojuego de género mata marcianos creado por Irem en 1987 para las máquinas recreativas o Arcade.

En este juego el jugador controla a un caza espacial llamado R-9 que se caracteriza por un láser que se puede cargar para aumentar la fuerza de impacto contra los enemigos y a la vez, con una cápsula, llamada Force (fuerza) que se deja anclar a la nave y otorga un arma definida de acuerdo con el color o la letra que tiene el Power-Up que se recoge a los siguientes. El poder de "Fuerza" aumenta conforme se recogen más armas. También existen extras para aumentar la velocidad de despliegue de la nave, cápsulas fijas sobre el techo y debajo de la base de R-9 y misiles detectores (homing) como añadido eficiente al disparo que ya poseemos. Lo que más distinguió a R-Type de los juegos de la competencia era la novedad de poder utilizar la cápsula "Force" tanto para atacar como para defenderse de los disparos enemigos y que esta se podía lanzar y separar de la nave para utilizarla como un satélite, además de forma libre, lo que habilitaba al jugador de anclar "Fuerza" en la cola de la nave y así dispara hacia atrás, es decir hacia la izquierda de la pantalla. R-Type fue el único juego que disponía de estas características y de un apartado técnico impecable. Daba igual la cantidad de enemigos o su tamaño en pantalla, R-Type no se ralentizaba ni se dañaba la apariencia en otros objetos que simultáneamente se movían sobre la pantalla. Este era un error muy común en aquellos tiempos pero Irem demostró su maestría en crear un shooter horizontal sin estos errores comunes. Este último hecho solo se entiende para la versión original de R-Type en máquina recreativa. Las conversiones para consolas y ordenadores solían tener algunos fallos pero era debido a que el hardware no cumplía con los requisitos para poder programar a un R-Type vistoso y sin ralentizaciones.

Puedes ver el juego en acción en este vídeo en YouTube:
http://www.youtube.com/watch?v=pVWtI0426mU

Figura 1. Imagen del juego original

➢ Implementación Obligatoria

El alumno deberá implementar un juego del estilo R-Type satisfaciendo los siguientes requisitos:

1- El juego comenzará con una pantalla de bienvenida a partir de la cual se podrá seleccionar el modo de juego (FACIL, NORMAL, COMPLICADO, IMPOSIBLE) y comenzar a jugar.
2- El juego constará de un único nivel donde el jugador deberá acabar con una horda de naves alienígenas. El número de alienígenas con los que acabar dependerá del modo de juego seleccionado. Fácil=10, Normal=15, Complicado=20, Imposible=30.
3- El jugador controlará la nave aliada y dispondrá de 1 sola vida.
4- Las naves alienígenas serán controladas por el ordenador.
5- Las naves alienígenas no disparan.
6- No hay que implementar relieve. Es decir, no hay que mostrar ningún tipo de suelo o techo como en el juego original.
7- La nave aliada podrá moverse arriba (Tecla Q), abajo (Tecla A), izquierda (Tecla O) y derecha (Tecla P). Así mismo podrá disparar su laser utilizando la tecla ESPACIO.
8- El área de movimiento permitido para la nave será toda la pantalla, aunque habrá que comprobar que la nave no salga de estos límites.
9- El disparo que realiza la nave aliada es continuo, es decir, no es necesario esperar a que el misil disparado abandone la pantalla para que la nave aliada pueda volver a disparar.
10- La nave aliada sólo puede realizar un tipo de disparo que se desplazará horizontalmente hacia la derecha de la pantalla, sin variar su trayectoria y a velocidad constante.
11- Las naves alienígenas se mueven a velocidad constante y podrán ser de dos tipos:
    a. **Nave Alienígena Tipo A.** Aparecen por la parte derecha de la pantalla y se mueven horizontalmente hacia la izquierda a velocidad constante sin variar su trayectoria, es decir, su coordenada "y" no varía en todo el desplazamiento.
    b. **Nave alienígena Tipo B.** Aparecen por la parte derecha de la pantalla y se mueven horizontalmente hacia la izquierda a velocidad constante. La principal diferencia con las Naves de Tipo A es que éstas pueden variar su trayectoria, es decir, en su desplazamiento horizontal pueden variar su coordenada "y" de manera aleatoria.
12- La velocidad a la que se mueven las naves alienígenas dependerá del modo de juego seleccionado. Todas las naves se mueven a la misma velocidad.
13- Cuando las naves alienígenas alcancen la parte izquierda de la pantalla volverán a aparecer por la parte derecha de ésta.
14- Se deberán de detectar dos tipos de colisiones.
    a. Las colisiones entre la nave aliada y las naves alienígenas, lo que supondrá el final del juego.
    b. Las colisiones entre los misiles disparados por la nave aliada y las naves alienígenas, lo que supondrá la destrucción de la nave alienígena contra la que ha chocado el misil.
15- Si el jugador finaliza el nivel del juego deberá aparecer un mensaje de felicitación y se volvería a mostrar el menú inicial.

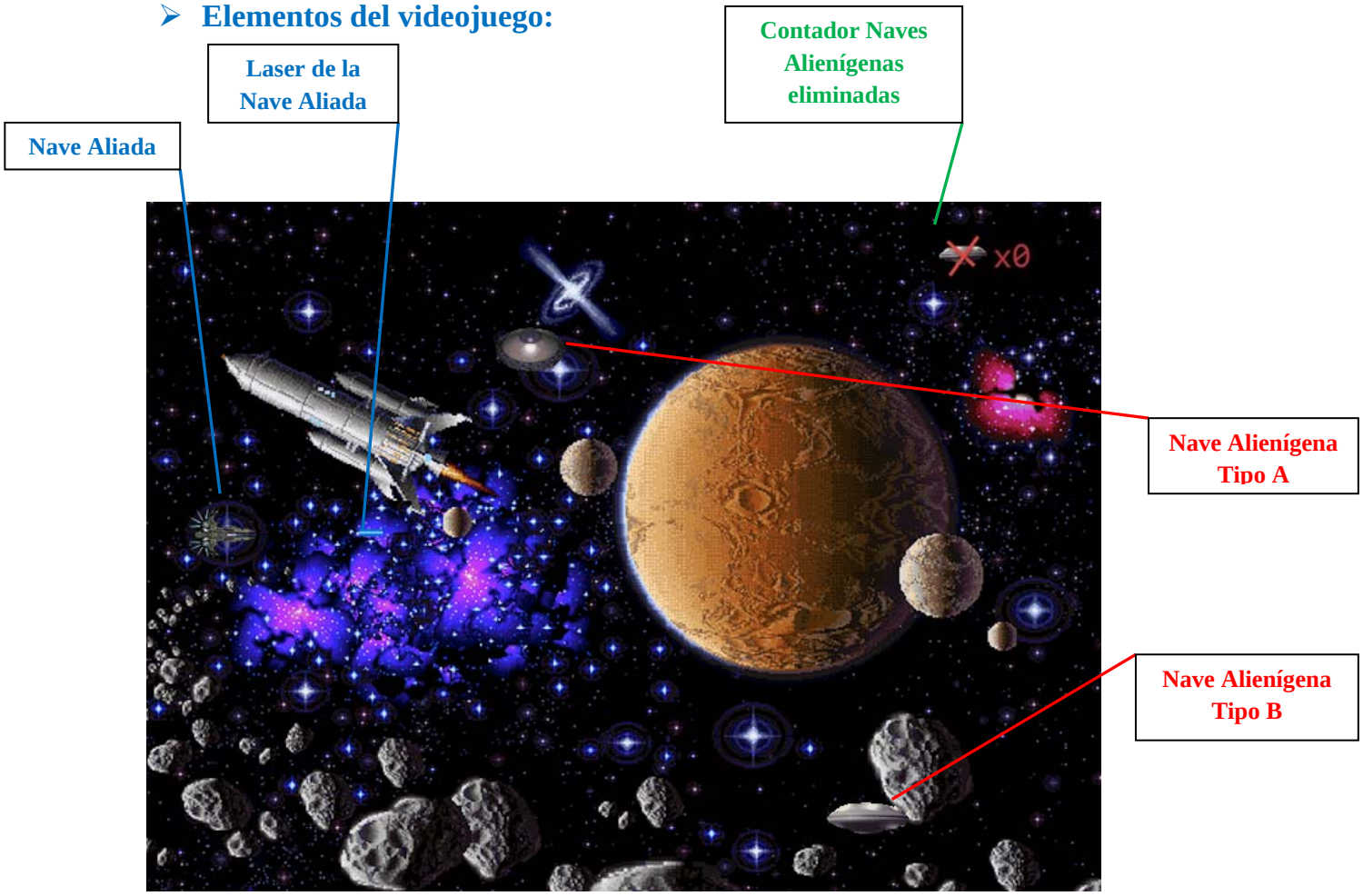
2. Funcionamiento y detalle de la aplicación:

➢ Pantalla principal del juego:

➢ Reglas del videojuego:

1. Una horda de naves alienígenas, avanza hacia el límite izquierdo de la pantalla.
2. El número de naves alienígenas y la velocidad de estas dependerá del modo seleccionado en la pantalla inicial.
3. La nave aliada dispara tantos láseres como el jugador pulse la tecla disparo.
4. Las naves alienígenas no disparan a la nave aliada.
5. La nave aliada no debe chocar con ninguna de las naves alienígenas ni con la explosión generada al destruir estas, o la partida se dará por finalizada.
6. La nave aliada dispone de una sola vida.
7. La nave aliada debe destruir todas las naves alienígenas para ganar la partida.

➢ Elementos del videojuego:



➢ Controles:

3. Análisis de la aplicación:

Para la realización de esta práctica, he seguido una serie de pasos que he ido anotando a modo diario, para el correcto desarrollo de este punto de la memoria. A continuación procedo a determinar el camino y decisiones que sigue la aplicación.

La aplicación empieza desde el siguiente método de la clase principal **RType**:

```
public static void main(String[] args) {
        finPartida();
}
```

En un principio este método se encargaba de crear una instancia de la clase principal **RType**, pero más adelante fue modificado para que llamase a un nuevo método y este se encargara de dicha función. La decisión fue tomada para la implementación de la ventana principal (mediante un *JFrame*), que se muestra en la página número 5 de esta práctica, para así poder relanzar dicha ventana tantas veces como sea necesario según la lógica del juego.

```
public static void finPartida() {
        RType menu = new RType();
        menu.setLocation(500, 200);
        menu.setVisible(true);
}
```

Una vez se muestra la ventana principal, a modo de menú del juego, se deberá pulsar uno de los botones y se creará una nueva instancia de la clase **JuegoNuevo**. Al pulsar uno de los botones, el juego se creará con unas características constantes durante el tiempo de partida (dependiendo del modo seleccionado, varían la velocidad en el eje “x” e “y” de las naves alienígenas).

```
public void actionPerformed (ActionEvent evento) {
        if (evento.getSource() == botonFacil) {
            JuegoNuevo.modoJuego = 1;
            Alienigena.xA = 800;
            Alienigena.yA = 100;
            Alienigena.xB = 800;
            Alienigena.yB = 300;
            Alienigena.dx = 1;
            Alienigena.dy = 2;
            setVisible(false);

            if (vuelta == true) {
                new JuegoNuevo();
                vuelta = false;
            }else {
             PantallaJuego.vuelta2 = true;
        }
}
```

.

.

.

Si se pulsa el botón salida, se cerrará la aplicación.

```
        else if (evento.getSource() == botonSalir) {
                    System.exit(0);
        }
```

La clase **JuegoNuevo** es la encargada de crear la pantalla principal de juego.

Mediante el método " `main(String[] args)`" de esta clase, se crea una nueva instancia de la clase **JuegoNuevo**.

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

Tras su creación, esta clase crea una pantalla (*JPanel*) haciendo uso de una nueva clase (**PantallaJuego**), con un formato prefijado y dependiendo del modo seleccionado, se muestra en la parte superior de la pantalla un título acorde y se establecen el número de alienígenas a destruir.

```
        add(new PantallaJuego());
// Tamaño de la ventana de juego.
        setSize(800, 600);
// Se establece la posición de la ventana en el centro.
        setLocationRelativeTo(null);
// Se desactiva la opción de modificar el tamaño de la ventana
.       setResizable(false);
// Se activa la visibilidad de la ventana.
        setVisible(true);

        switch (modoJuego) {
        case 1:
        setTitle("R - Type <<< Modo Facil >>>");
        PantallaJuego.maxMuertes = 10;
                        break;
                                                .

                                                .

                                                .
```

La clase **PantallaJuego,** gestiona el funcionamiento de la aplicación (fondo de pantalla, nave aliada, naves alienígenas, disparos laser…).

El constructor se encarga de crear todos los elementos necesarios: fondo de pantalla, nave aliada, naves alienígenas, disparos laser…, así como un timer, que marcará el lapso de tiempo con el que se va a realizar las tareas del juego.

```
public PantallaJuego() {
// Se recibe la imagen del fondo de pantalla de una imagen externa.
        ImageIcon ii = new ImageIcon(this.getClass().getResource("universe2.gif"));
        imagen = ii.getImage();
// Se recibe la imagen de origen del contador de naves alienígenas eliminadas de una
imagen externa.
        ImageIcon iii = new ImageIcon(this.getClass().getResource("x0. jpg"));
        contNaves = iii.getImage();
// Se añade la opción de "escuchar" la tecla pulsada en cada m omento mediante la clase
interna TAdapter.
        addKeyListener(new TAdapter());

// Para que un objeto JPanel reciba las notificaciones del teclado es necesario incluir la
siguiente instrucción.
        setFocusable(true);
// Se activa el color negro por defecto en el fondo de pantalla.
```

```
        setBackground(Color.BLACK);
// Esta opción dibuja primero en memoria, y luego dibuja todo junto en pantalla.
        setDoubleBuffered(true);
 // Se crea un nuevo objeto Nave Aliada.
        aliada = new Aliada();
 // Se crea un nuevo objeto alienigena tipo A.
        alienigenaA = new Alienigena();
 // Se crea un nuevo objeto alienigena tipo B.
        alienigenaB = new Alienigena();
 // Se inicializa el contador de muertes, y las variable auxiliares de validar tecla enter y de
     inicio de juego nuevo.
        muertes = 0;
        desactEnter = false;
        vuelta2 = false;
 // Se crea un nuevo timer para el manejo de cada momento de ejecución.
        timer = new Timer(4, this);
        timer.start();
 }
```

A partir de aquí, se ejecuta el cuerpo del juego, siguiendo las reglas explicadas en la página 5. Una vez creada la pantalla principal se procede al manejo de los elementos a cada “golpe de timer”. Al final de este documento/memoria, adjunto el resto del código fuente con cada línea debidamente explicada y comentada.

Hay varios procedimientos que creo necesarios explicar:

* En el método *paint(Graphics g)*, el cual va dibujando todos los elementos en cada instante, he implementado un procedimiento para manejar las colisiones nave aliada/naves alienígenas y laser/naves alienígenas, haciendo uso del método *getLasers()* de la clase **Laser** y la utilidad “*Rectangle*” junto al método *getBounds*() de cada objeto.

La variable *alienigenaA.nuevoA*, se encarga de que las naves no puedan ser intersectadas por el láser hasta que estas no hayan reaparecido por los límites derechos.

```
 // Se crea un array con los láser creados en el momento y se recorre, dibujándolos y
 controlando si colisionan con las naves alienígenas.
 // Mediante el parámetro rectángulo, manejamos los limites de los disparos laser y las naves
 alienígenas y si estos "intersectan", se produce la explosión de la nave alienígena. Esto da
 lugar al sumatorio del contador de nave s eliminadas.


 ArrayList lsr = aliada.getLasers();

      for (int i = 0; i < lsr.size(); i++ ) {
           Laser ls = (Laser) lsr.get(i);
           g2d.drawImage(ls.getImage(), ls.getX(), ls.getY(), this);
           Rectangle rLaser = ls.getBounds();
           Rectangle rAlienigenaA = alienigenaA.getBoundsA();

          if (rLaser.intersects(rAlienigenaA) && alienigenaA.nuevoA) {
              ImageIcon ii = new ImageIcon(this.getClass().getResour
ce("explosionNAVE.gif"));
              Alienigena.imageA = ii.getImage();
```

```
            colision = true;
            alienigenaA.muertoA();
            muertes +=1;
            eliminados();
            ImageIcon iiii = new ImageIcon(this.getClass().getReso urce(alienEliminados));
            contNaves = iiii.getImage();
            alienigenaA.nuevoA = false;
        }
```

.

.

.

.

.

.

```
// Se manejan las colisiones entre la nave Aliada y naves alienígenas.
// Si estas "intersectan", se produce el final de partida, se carga una nueva imagen de fondo
// indicando "final de partida" y la nave aliada desaparece de la pantalla.

            Rectangle rAliada = aliada.getBounds();
            Rectangle rAlienigenaA = alienigenaA.getBoundsA();
            if (rAliada.intersects(rAlienigenaA)) {
                ImageIcon i = new ImageIcon(this.getClass().getResource("e
xplosionNAVE.gif"));
                Alienigena.imageA = i.getImage();
                ImageIcon ii = new ImageIcon(this.getClass().getResource(" naveExpl.gif"));
                Aliada.image = ii.getImage();
                ImageIcon iii = new ImageIcon(this.getClass().getResource( "gameover.jpg"));
                imagen = iii.getImage();

// Se desactiva la imagen contador de naves eliminadas.

                contNaves = null;
                aliada.x = -9000;
                aliada.y = -9000;
                desactEnter = true;
            }
```


```
// Método para actualizar la imagen contador de naves eliminadas según el numero de naves
// Alienígenas eliminadas.
// Si se alcanza el limite indicado en cada modo de juego, se carga una nueva imagen de fondo
// Indicando que la partida ha sido ganada y colocando las naves alienígenas fuera de pantalla.
```

* En el método *eliminados()*, se van controlando el número de naves alienígenas eliminadas, a medida que se van destruyendo, se actualiza el contador de naves. Si se llega al límite de naves destruidas del modo de juego seleccionado, se gestiona la partida conseguida.

```
public void eliminados() {
     switch (muertes) {
          case 1:
              alienEliminados = "x1.jpg";
              break;

                                        .

                                        .

                                        .

          case 10:
               if (maxMuertes == 10 && JuegoNuevo.modoJuego == 1) {
                  ImageIcon i = new ImageIcon(this.getClass().getResource("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 = "x10.jpg";
              break;

                                        .

                                        .

                                        .
```

*He implementado una condición en el método *actionPerformed(ActionEvent)* para controlar el "reinicio" de los elementos del juego cada vez que se vuelve a jugar una partida nueva.

```
public void actionPerformed(ActionEvent e) {
    if (vuelta2 == true) {
        muertes = 0;
        aliada.x = 40;
        aliada.y = 250;
        ImageIcon ii = new ImageIcon(this.getClass().getResource(" universe2.gif"));
        imagen = ii.getImage();
        ImageIcon iii = new ImageIcon(this.getClass().getResource( "aliada.png"));
        Aliada.image = iii.getImage();
        alienEliminados = "x0.jpg";
        ImageIcon iiii = new ImageIcon(this.getClass().getResource(alienEliminados));
        contNaves = iiii.getImage();
        vuelta2 = false;
    }
```

.

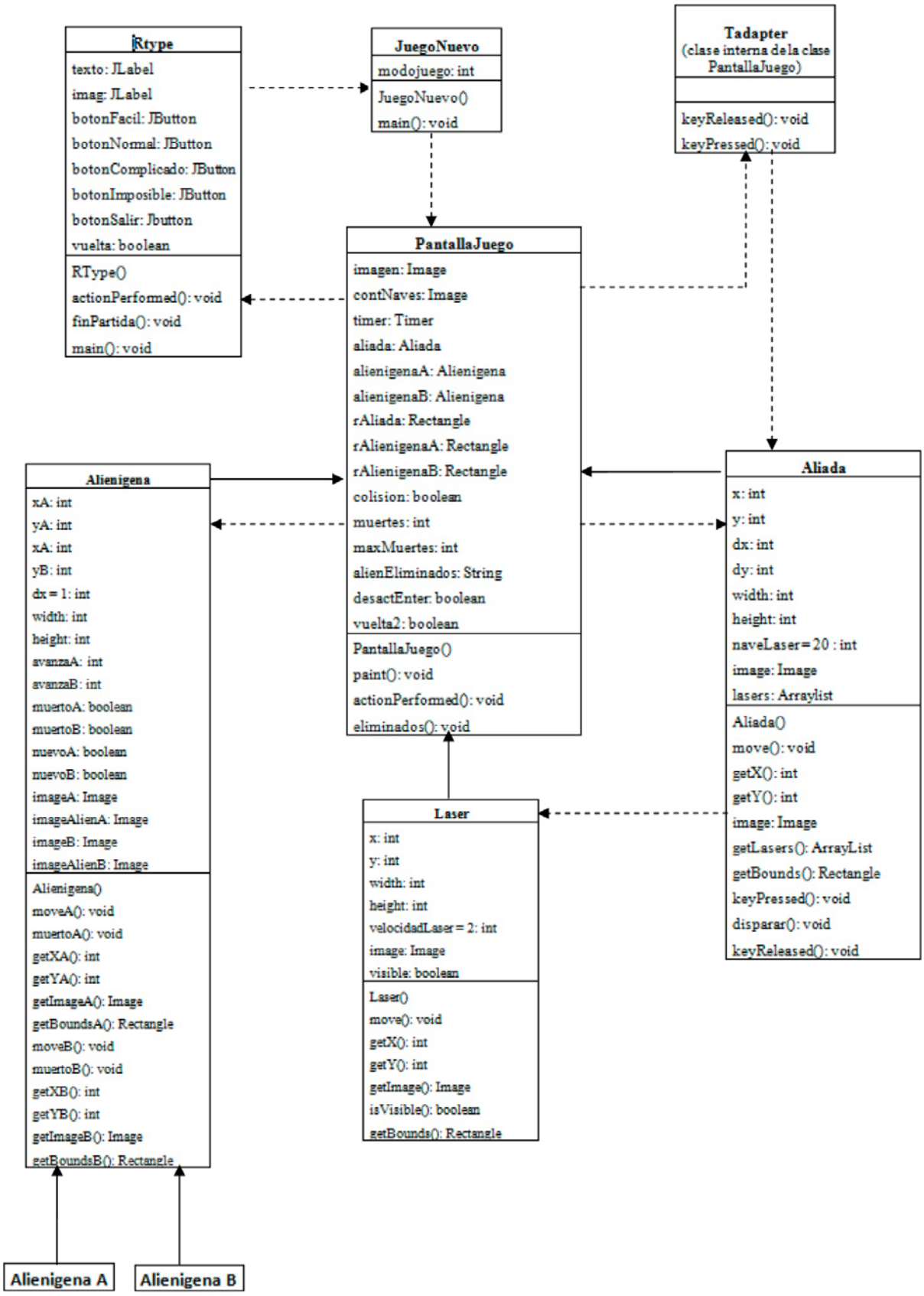
.

.

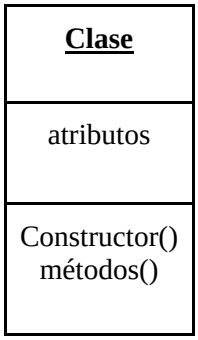
*Respecto a las naves alienígenas, he decidido crear dos objetos: alienigenaA y alienigenaB, cada una con su movimiento y desarrollo propio. Estas siguen la lógica del juego como si se tratasen de diferentes naves en cada ocasión, lo que crea la "ilusión" de parecer un horda de naves alienígenas.

4. Diseño y Diagrama de clases:

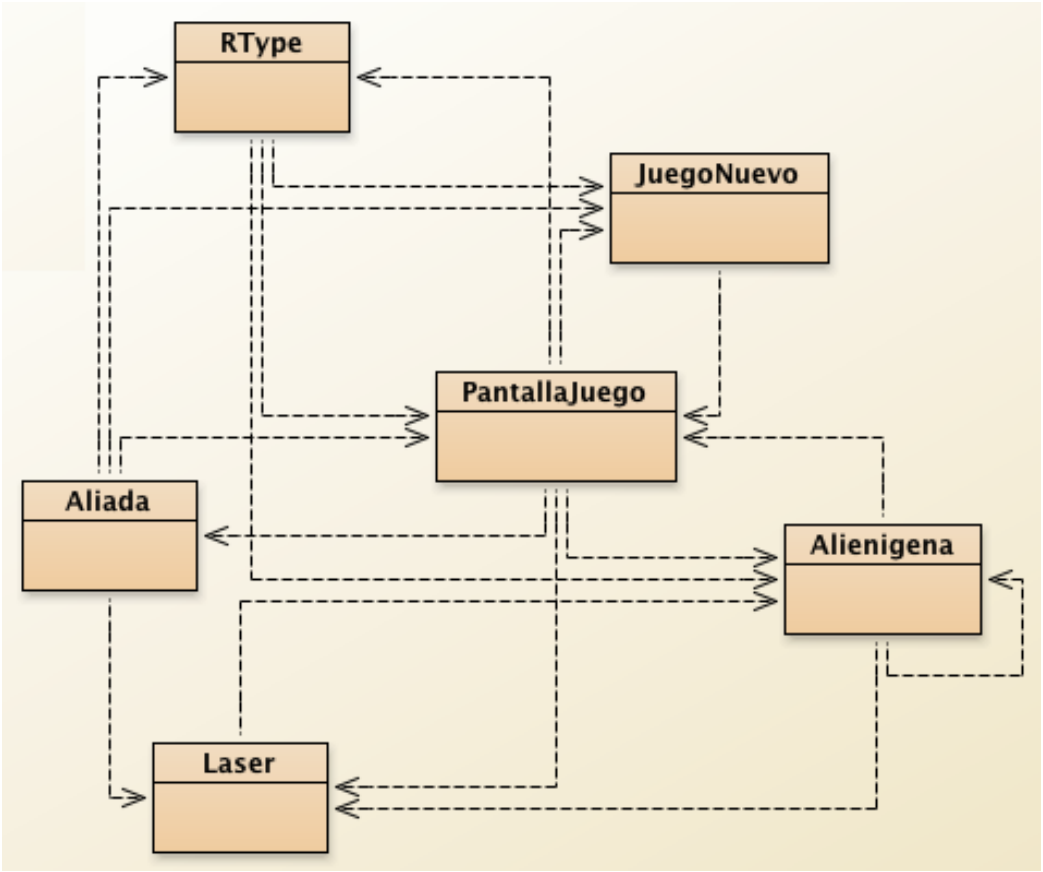
➢ Diseño



Antes de comenzar a desarrollar cualquier aplicación, debe realizarse previamente un paso muy importante. Este paso analítico son los *diagramas de clases*, que plasman la arquitectura propia de la aplicacion y se estructuran de la siguiente manera:

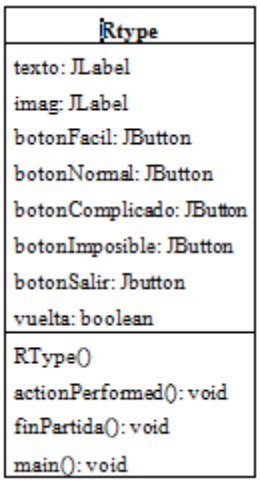


➢ Diagrama de Clases aplicación RType:



5. Clases/Objetos y sus métodos

➢ Clase Rtype:

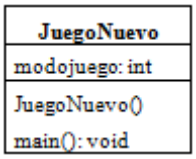



```
/*************************************************************************************************
 * Clase RType.                                                                                  *
 * ----------------------------------------------------------------------------------------------*
 * Clase principal de la aplicacion java.                                                        *
 * Se definen los detalles de la pantalla principal de la aplicacion (tamaño, títulos, modos de  *
 * juego) y se carga la partida seleccionada por el jugador.                                     *
 * <<< Borja Delgado Angulo >>>                                                                  *
 *************************************************************************************************/
```

Constructor():	Descripción:
RType()	Constructor de la clase Rtype. Constructor de la clase Rtype. Crea una nueva ventana de menú opciones.

Métodos():	Descripción:
actionPerformed():	Maneja el botón pulsado. Según la selección, se cargaran las opciones definidas por el modo de juego (cantidad y velocidad de las naves alienígenas) o se sale de la aplicacion.
finPartida():	Si la partida finaliza, ya sea por haber vencido o por haber sido derrotado, se carga una nueva pantalla de selección.
main():	

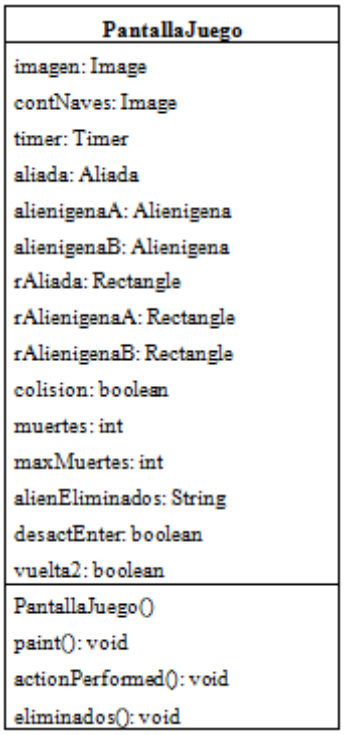
➢ Clase JuegoNuevo:




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

Constructor():	Descripción:
JuegoNuevo()	Constructor de la clase JuegoNuevo. Crea una nueva pantalla de juego.

➢ Clase PantallaJuego:



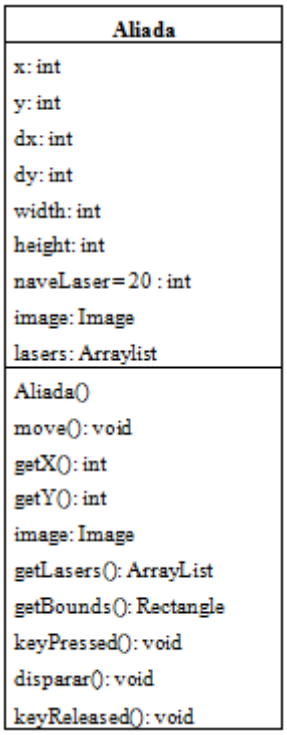

```
/*************************************************************************************************
 * Clase PantallaJuego.                                                                          *
 * ----------------------------------------------------------------------------------------------*
 * En esta clase se dibuja la imagen de fondo del juego, las imágenes de las naves aliada y      *
 * alienígenas, los disparos láser y el manejo de estos según las condiciones que se indican en  *
 * cada caso.                                                                                    *
 * <<< Borja Delgado Angulo >>>                                                                  *
*************************************************************************************************/
```

Constructor():	Descripción:
PantallaJuego()	Constructor de la clase PantallaJuego. Crea una nueva pantalla de juego.

Métodos():	Descripción:
PantallaJuego()	Gestiona los detalles de la pantalla (JPanel) principal del juego y crea los elementos necesarios para el correcto funcionamiento de la aplicación.
paint()	Dibuja en el Jpanel todos los elementos que participan en la ejecución.
actionPerformed()	Maneja el avance de los elementos en ejecución.
eliminados()	Controla el contador de naves eliminadas.

-

➢ Clase Aliada:

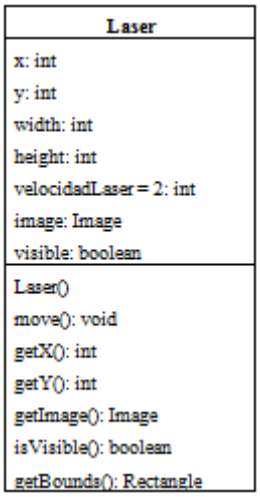



```
/**************************************************************************************************
 * Clase Aliada.                                                                                  *
 * ---------------------------------------------------------------------------------------------*
 * En esta clase se definen los atributos de la nave aliada, movimiento y se trata el manejo     *
 * de la misma.                                                                                   *
 * <<< Borja Delgado Angulo >>>                                                                   *
 **************************************************************************************************/
```

Constructor():	Descripción:
Aliada()	Constructor de la clase Aliada Crea una nueva nave aliada.

Métodos():	Descripción:
move()	Gestiona el movimiento de la nave aliada.
getX()	Método para obtener la posición de la nave en ese momento en el eje X.
getY()	Método para obtener la posición de la nave en ese momento en el eje Y.
Image()	Método para obtener la imagen de la nave Aliada.
getLasers()	Método para obtener la posición de los láser en curso.
getBounds()	Método para obtener los límites de la imagen de la nave aliada.
keyPressed()	Método para manejar la acción de pulsar una tecla.
disparar()	Método para crear un nuevo laser y posiciónar el origen de este en la nave aliada.
keyReleased()	Método para manejar la acción de soltar una tecla y detener el movimiento de la nave aliada.

➢ Clase Laser:

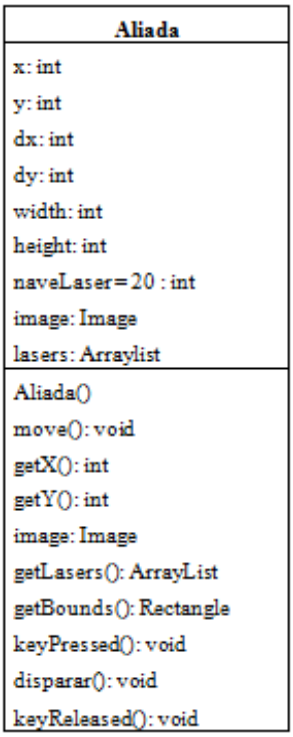



```
/**************************************************************************************************
 * Clase Laser.                                                                                   *
 * ---------------------------------------------------------------------------------------------*
 * En esta clase se definen los atributos de los láser, movimiento y se trata el manejo de los    *
 * mismos.                                                                                        *
 * <<< Borja Delgado Angulo >>>                                                                   *
 **************************************************************************************************/
```

Constructor():	Descripción:
Laser()	Constructor de la clase Laser. Crea un nuevo disparo laser.

Métodos():	Descripción:
move()	Gestiona el movimiento del laser disparado.
getX()	Método para obtener la posición del laser en ese momento en el eje X.
getY()	Método para obtener la posición del laser en ese momento en el eje Y.
getImage()	Método para obtener la imagen del laser.
isVisible()	Método para devolver visible al laser.
getBounds()	Método para obtener los límites de la imagen del laser.

➢ Clase Alienígena:




```
/**************************************************************************************************
 * Clase Alienígena.                                                                              *
 * ---------------------------------------------------------------------------------------------*
 * En esta clase se definen los atributos de las diferentes naves alienígenas, movimiento y se   *
 * trata el manejo estas.                                                                         *
 * <<< Borja Delgado Angulo >>>                                                                   *
 **************************************************************************************************/
```

Constructor():	Descripción:
Alienígena()	Constructor de la clase Laser. Crea nuevos alienígenas.

Métodos():	Descripción:
moveA()	Método para el movimiento de las Naves tipo A.
muertoA()	Método para devolver la posición en la que la nave alienígena tipo A es eliminada.
getXA()	Obtenemos la posición de la nave alienígena tipo A en ese momento en el eje X.
getYA()	Obtenemos la posición de la nave alienígena tipo A en ese momento en el eje Y.
getImageA()	Método para obtener la imagen de la nave alienígena tipo A.
getBoundsA()	Método para obtener los límites de la imagen de la nave alienígena tipo A.
moveB()	Método para el movimiento de las Naves tipo B.
muertoB()	Método para devolver la posición en la que la nave alienígena tipo B es eliminada.
getXB()	Obtenemos la posición de la nave alienígena tipo B en ese momento en el eje X.
getYB()	Obtenemos la posición de la nave alienígena tipo B en ese momento en el eje Y.
getImageB()	Método para obtener la imagen de la nave alienígena tipo B.
getBoundsB()	Método para obtener los límites de la imagen de la nave alienígena tipo B.

6. Modo de empleo y contenido de la aplicación:

Para poder hacer uso de la aplicación de forma óptima, estos son los requerimientos con los que he trabajado:

- Sistema Operativo: Mac OS X Versión 10.7.5.
- BlueJ versión 3.0.4 (Java versión 1.6.0_45).
- Virtual machine: Java HotSpot(TM) 64-Bit Server VM 20.45-b01-451 (Apple Inc.).

➢ Contenido de la entrega:

- Memoria con la información requerida en el plan de trabajo de la asignatura y el código fuente incluido.
- Archivo RType.jar ejecutable.

7. Código fuente de la aplicación:

➢ Clase RType

```
/*****************************************************************************************************
 * Clase RType.                                                                                      *
 * ------------------------------------------------------------------------------------------------- *
 * Clase principal de la aplicación java.                                                            *
 * Se definen los detalles de la pantalla principal de la aplicacion (tamaño, títulos, modos de      *
 * juego) y se carga la partida seleccionada por el jugador.                                         *
 * <<< Borja Delgado Angulo >>>                                                                      *
 *****************************************************************************************************/

      // Lista de bibliotecas.
      import javax.swing.JFrame;
      import javax.swing.JButton;
      import javax.swing.JLabel;
      import javax.swing.ImageIcon;

      import java.awt.FlowLayout;
      import java.awt.event.ActionEvent;
      import java.awt.event.ActionListener;

      // Clase RType.
      public class RType extends JFrame implements ActionListener {
           // Atributos.
           // Etiquetas para el texto y la imagen de la ventana.
           JLabel texto, imag;
           // Botones de selección de la ventana principal.
           JButton botonFacil, botonNormal, botonComplicado, botonImposible, botonSalir;

           public static boolean vuelta = true;

           // Constructor de la clase RType.
           public RType() {
                setTitle("Practica POO 2013: R-Type");
                // Tamaño de la ventana principal.
                setBounds(500,200,250,450);
                // Contenedor que pone los elementos en línea.
                setLayout (new FlowLayout());
                // Se desactiva la opción de modificar el tamaño de la ventana.
                setResizable(false);

                // Se definen los elementos que contendrá el contenedor.
                // Imagen de la ventana principal.
                imag = new JLabel(new ImageIcon(getClass().getResource("imagenPrincipal.jpg")));

                // Etiqueta con texto selección.
                texto = new JLabel("Seleccionar dificultad");

                // Botones de selección.
                botonFacil = new JButton("Facil (10 enemigos)");
                botonNormal = new JButton("Normal (15 enemigos)");
                botonComplicado = new JButton("Complicado (20 enemigos)");
                botonImposible = new JButton("          IMPOSIBLE (30 enemigos)        ");
                botonSalir = new JButton("Salir");

                // Se añaden los elementos una vez definidos.
                botonFacil.addActionListener (this);
                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 opción de salir de la aplicación al pulsar el botón de
salir(x) de la ventana.
72                this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
73           }
74
75           // Método para manejar el botón pulsado.
76           // Según la selección, se cargaran las opciones definidas por el modo de juego
77           // (cantidad y velocidad de las naves alienígenas) o se sale de la aplicación.
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 sido
173          // derrotado, se carga una nueva pantalla de selección.
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     }
```

Clase JuegoNuevo

```
1       /*********************************************************************************************
2       * Clase JuegoNuevo.                                                                          *
3       * -------------------------------------------------------------------------------------------*
4       * Se definen los detalles de la ventana de juego y se carga el limite de muertes alienígena  *
5       * según el modo de juego seleccionado.                                                       *
6       * <<< Borja Delgado Angulo >>>                                                               *
7       *********************************************************************************************/
8
9         // Lista de bibliotecas.
10        import javax.swing.JFrame;
11
12        // Clase JuegoNuevo.
13        public class JuegoNuevo extends JFrame {
14             //Atributos.
15             public static int modoJuego;
16
17             // Constructor de la clase JuegoNuevo.
18             public JuegoNuevo() {
19                  add(new PantallaJuego());
20
21                       // Se activa la opción de salir de la aplicación al pulsar el botón
de salir(x) de la ventana.
22                  setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
23                  // Tamaño de la ventana de juego.
24                  setSize(800, 600);
25                  // Se establece la posición de la ventana en el centro.
26                  setLocationRelativeTo(null);
27                  // Se desactiva la opción de modificar el tamaño de la ventana
.
28                  setResizable(false);
29                  // Se activa la visibilidad de la ventana.
30                  setVisible(true);
31
32                  // Dependiendo del valor pasado en la clase RType, se cargan los títulos y
33                  // límite de navea eliminar correspondientes.
34                  switch (modoJuego) {
35                       case 1:
36                            setTitle("R - Type <<< Modo Facil >>>");
37                            PantallaJuego.maxMuertes = 10;
38                            break;
39                       case 2:
40                            setTitle("R - Type <<< Modo Normal >>>");
41                            PantallaJuego.maxMuertes = 15;
42                            break;
43                       case 3:
44                            setTitle("R - Type <<< Modo Complicado >>>");
45                            PantallaJuego.maxMuertes = 20;
46                            break;
47                       case 4:
48                            setTitle("R - Type <<< Modo IMPOSIBLE >>>");
49                            PantallaJuego.maxMuertes = 30;
50                            break;
51                  }
52             }
53
54             // Método para cargar una partida nueva.
55             public static void main(String[] args) {
56                  JuegoNuevo juegoNuevo = new JuegoNuevo();
57             }
58        }
```

➢ Clase Pantalla juego

```
1  /**********************************************************************************************
2  * Clase PantallaJuego.                                                                        *
3  * ------------------------------------------------------------------------------------------  *
4  * En esta clase se dibuja la imagen de fondo del juego, las imágenes de las naves aliada y    *
5  * alienígenas, los disparos láser y el manejo de estos según las condiciones que se indican en *
6  * cada caso.                                                                                   *
7  * <<< 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 imágenes del fondo y del contador de naves alienígenas
eliminadas.
32            Image imagen, contNaves;
33            // Variable timer para manejar cada momento de la ejecución.
34            private Timer timer;
35            // Variable para utilizar la nave aliada.
36            private Aliada aliada;
37            // Variable para utilizar las naves alienígenas.
38            private Alienigena alienigenaA;
39            private Alienigena alienigenaB;
40
41            // Variables rectángulo para manejar colisiones.
42            Rectangle rAliada;
43            Rectangle rAlienigenaA;
44            Rectangle rAlienigenaB;
45
46            // Variable para validar si el láser colisiona con una nave alienígena.
47            public static boolean colision;
48
49            // Variable contador de naves alienígenas eliminadas.
50            public int muertes;
51            // Variable para controlas el máximo número de naves a eliminar según modo de
juego seleccionado.
52            public static int maxMuertes;
53
54            // Variable para modificar la imagen contador de naves alienígenas 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 externa.
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 alienigenas
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 opción 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 teclado es
necesario
78                 // incluir la siguiente instrucción.
79                 setFocusable(true);
80
81                 // Se activa el color negro por defecto en el fondo de pantalla.
82                 setBackground(Color.BLACK);
83
84                 // Esta opción 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 alienígena tipo A.
90                 alienigenaA = new Alienigena();
91                 // Se crea un nuevo objeto alienígena tipo B.
92                 alienigenaB = new Alienigena();
93
94                 // Se inicializa el contador de muertes, y las variable auxiliares de
validar tecla
95                 // enter y 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 ejecución.
101                timer = new Timer(4, this);
102                timer.start();
103           }
104
105           // Método para dibujar en el JPanel todos los elementos que participan en la
ejecución.
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 lásers creados en el momento y se recorre,
118                 // dibujándolos y controlando si colisionan con las naves alienígenas.
119                 // Mediante el parámetro rectángulo, manejamos los límites de los disparos
120                 // laser y las naves alienígenas y si estos "intersectan", se produce la
121                 // explosión de la nave alienígena. 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().getResource("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 alienígenas.
170                 // Si estas "intersectan", se produce el final de partida, se carga una
171                 // nueva imagen de fondo 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 ejecución.
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 método isVisible() de la clase laser, se define
231                 // si el láser avanza una posición 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 nave
239                     // alienígena, sino avanza una posición.
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 posición.
251                aliada.move();
252                // La nave alienígena tipo A avanza una posición.
253                alienigenaA.moveA();
254                // La nave alienígena tipo B avanza una posición.
255                alienigenaB.moveB();
256                // Se llama a este método para volver a pintar los elementos en su nueva
257                // posición a cada golpe de timer.
258                repaint();
259           }
260
261           // Método para actualizar la imagen contador de naves eliminadas según el número
262           // de naves alienígenas eliminadas.
263           // Si se alcanza el limite indicado en cada modo de juego, se carga una nueva
264           // imagen de fondo indicando que la partida ha sido ganada y colocando las naves
alienígenas 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().getResource("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";
368                          break;
369                     case 23:
370                          alienEliminados = "x23.jpg";
371                          break;
372                     case 24:
373                          alienEliminados = "x24.jpg";
374                          break;
375                     case 25:
376                          alienEliminados = "x25.jpg";
377                          break;
378                     case 26:
379                          alienEliminados = "x26.jpg";
380                          break;
381                     case 27:
382                          alienEliminados = "x27.jpg";
383                          break;
384                     case 28:
385                          alienEliminados = "x28.jpg";
386                          break;
387                     case 29:
388                          alienEliminados = "x29.jpg";
389                          break;
390                     case 30:
391                          if (maxMuertes == 30 && JuegoNuevo.modoJuego == 4) {
392                               ImageIcon i = new ImageIcon(this.getClass().getRes ource("youwin.jpg"));
393                               imagen = i.getImage();
394                               Alienigena.xA=3000;
395                               Alienigena.xB=3000;
396                               Alienigena.seMueve = false;
397                               Alienigena.dx=0;
398                               Alienigena.dy=0;
399                               alienEliminados = "aliensad.jpg";
400
401                               desactEnter = true;
402                          }else
403                               alienEliminados = "x30.jpg";
404                          break;
405               }
406           }
407
408          //Clase interna TAdapter, se encarga de manejar las acciones pulsar/soltar una tecla.
409           private class TAdapter extends KeyAdapter {
410
411                public void keyReleased(KeyEvent e) {
412                     aliada.keyReleased(e);
413                }
414
415               public void keyPressed(KeyEvent e) {
416                     aliada.keyPressed(e);
417               }
418           }
419  }
```

➢ Clase 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 Delgado Angulo >>>                                                                  *
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             // Posición.
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 lásers 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             // Método para el movimiento de la nave.
53             public void move() {
54                  // La nave aliada avanza una posición.
55                  x += dx;
56                  y += dy;
57
58                  // Se controla que la nave aliada no se salga de los límites que 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             // Método para obtener la posición de la nave en ese momento en el eje X.
77             public int getX() {
78                  return x;
79             }
80
81             // Método para obtener la posición de la nave en ese momento en el eje Y.
82             public int getY() {
83                  return y;
84             }
85
86             // Método para obtener la imagen de la nave Aliada.
87             public Image getImage() {
88                  return image;
89             }
90
91             // Método para obtener la posición de los lásers en curso.
92             public ArrayList getLasers() {
93                  return lasers;
94             }
95
96             // Método para obtener los límites de la imagen de la nave aliada.
97             public Rectangle getBounds() {
98                  return new Rectangle(x, y, width-40, height-40);
99             }
100
101            // Método para manejar la acción de pulsar una tecla.
102            public void keyPressed(KeyEvent e) {
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 selección 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 aplicación.
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  }
```

➢ Clase Laser

```
/*****************************************************************************************************
* Clase Laser.                                                                                       *
* ---------------------------------------------------------------------------------------------      *
* En esta clase se definen los atributos de los lasers, movimiento y se trata el manejo de los *
* mismos.                                                                                            *
* <<< Borja Delgado Angulo >>>                                                                       *
*****************************************************************************************************/

        // Lista de bibliotecas.
        import java.awt.Image;
        import java.awt.Rectangle;

        import javax.swing.ImageIcon;

        // Clase Láser.
        public class Laser {
             // Atributos.
             // Posición.
             private int x, y;
             // Medidas del objeto.
             public int width, height;
             // Velocidad del disparo laser.
             private final int velocidadLaser = 2;

             //Variable para la imagen del láser.
             private Image image;

             //Variable para validar si el láser es visible o no.
             boolean visible;

             // Constructor de la clase Laser.
             public Laser(int x, int y) {
                  // Se recibe la imagen del láser de una imagen externa.
                  ImageIcon ii = new ImageIcon(this.getClass().getResource("lase r.jpg"));
                  image = ii.getImage();
                  // Se obtienen las medidas de la imagen del láser.
                  width = image.getWidth(null);
                  height = image.getHeight(null);

                  // Se define a visible por defecto.
                  visible = true;

                  // Se define la posición del láser la que tiene en ese momento
                  this.x = x;

                 this.y = y;
             }

             // Método para el movimiento del láser.
             public void move() {
                  // El láser avanza una posición.
                  x += velocidadLaser;

                  //Si el láser pasa del límite izquierdo de la pantalla, este deja de verse.
                  if (x > 810){
                       visible = false;
                  }
             }

             // Método para obtener la posición del láser en ese momento en el eje X.
```

```
60              public int getX() {
61                   return x;
62              }
63
64              // Método para obtener la posición del láser en ese momento en el eje Y.
65              public int getY() {
66                   return y;
67              }
68
69              // Método para obtener la imagen del láser.
70              public Image getImage() {
71                   return image;
72              }
73
74              // Método para devolver visible al láser.
75              public boolean isVisible() {
76                   return visible;
77              }
78
79              // Método para obtener los límites de la imagen del láser.
80              public Rectangle getBounds() {
81                   return new Rectangle(x, y, width-5, height-5);
82              }
83         }
```

➢ Clase Alienígena

```
1  /*********************************************************************************************
2  *      Clase Alienígena.                                                                     *
3  * ------------------------------------------------------------------------------------------ *
4  * En esta clase se definen los atributos de las diferentes naves alienígenas, 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            // Posición.
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 posición en caso de colisión.
26            public static int avanzaA;
27            public static int avanzaB;
28
29            // Valida si una nave alienígena ha sido eliminada.
30            public boolean      muertoA;
31            public boolean muertoB;
32            // Valida si se debe crear una nueva nave alienígena, para no suma r al contador
cuando
33            // lo sea necesario.
34            public boolean nuevoA = true;
35            public boolean nuevoB = true;
36
37            public static boolean seMueve = true;
38
39            //Variable para las imágenes de las naves alienígenas.
40            public static Image imageA;
41            private Image imageAlienA;
42            public static Image imageB;
43            private Image imageAlienB;
44
45            // Constructor de la clase Alienígena.
46            public Alienigena() {
47                 // Se recibe la imagen de la nave Alienígena tipo A de una imagen externa.
48                 ImageIcon ii = new ImageIcon(this.getClass().getResource("spac eshipA.gif"));
49                 // Se obtienen las medidas de la imagen de la nave Alienígena tipo A.
50                 imageA = ii.getImage();
51                 imageAlienA = ii.getImage();
52
53                 // Se recibe la imagen de la nave Alienígena tipo B de una imagen 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 imágenes de las naves Alienígenas 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 alienígenas al comienzo del juego.
64                 xA = 800;
65                 yA = 100;
66                 xB = 800;
67                 yB = 300;
68            }
69
70           // Método para el movimiento de las Naves tipo A.
71           public void moveA() {
72                // Hasta que no comience una nueva partida, la nave alienígena A se queda
sin moverse.
73                if (seMueve == true) {
74                      // La nave alienígena tipo B avanza una posición.
75                      xA -= dx+1;
76
77                      // Si la nave alienígena tipo A llega un poco más lejos del límite
78                      // izquierdo de la pantalla, esta aparece de nuevo por el lado
79                      // derecho en una posición diferente en el 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 alienígena tipo A es eliminada, esta avanza unas
91                      // posiciones en el eje "x" e "y", para dar tiempo a que se cargue la
92                      // imagen de explosión. Después, esta aparece de nuevo por el lado
93                      // derecho en una posición diferente en el eje "y", con la imagen
recargada de nuevo, como si se tratase de una nueva nave alienígena.
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          // Método para devolver la posición en la que la nave alienígena tipo A es
eliminada.
112          public void muertoA() {
113               avanzaA = xA;
114               muertoA = true;
115          }
116
117          // Obtenemos la posición de la nave alienígena tipo A en ese momento en el eje X.
118           public int getXA() {
119                return xA;
```

```
120          }
121
122         // Obtenemos la posición de la nave alienígena tipo A en ese momento en el eje Y.
123          public int getYA() {
124               return yA;
125          }
126
127          // Método para obtener la imagen de la nave alienígena tipo A.
128           public Image getImageA() {
129                return imageA;
130          }
131
132          // Método para obtener los límites de la imagen de la nave alienígena tipo A.
133          public Rectangle getBoundsA() {
134               return new Rectangle(xA, yA, widthA-10, heightA-10);
135          }
136
137          // Método 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 alienígena tipo B llega a u no de los
142                    // límites de la pantalla 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 alienígena tipo B avanza una posición.
152                    xB -= dx;
153                    yB += dy;
154
155                    // Si la nave alienígena tipo B llega un poco más lejos del límite
156                    // izquierdo de la pantalla, esta aparece de nuevo por el lado
157                    // derecho en una posición diferente en el 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 alienígena tipo B es eliminada, esta avanza unas
168                    // posiciones en el eje "x" para dar tiempo a que se cargue la imagen
169                    // de explosión. Después, esta aparece de nuevo por el lado derecho en
170                    // una posición diferente en el eje "y", con la imagen recargada de
nuevo, como si se tratase de una nueva nave alienígena.
171                    if (muertoB) {
172                         dy = 0;
173                         if (xB < avanzaB-150) {
174                              xB += 1000;
175                              dy = 2;
176                              if (yB <= 550) {
177                                   yB = yB+92;
178                              }else yB = 10;
179
```

```
180                              imageB = imageAlienB;
181                              muertoB = false;
182                              nuevoB = true;
183                         }
184                    }
185               }
186          }
187
188          // Método para devolver la posición en la que la nave alienígena tipo B es
eliminada.
189          public void muertoB() {
190               avanzaB = xB;
191               muertoB = true;
192          }
193
194          // Obtenemos la posición de la nave alienígena tipo B en ese momento en el eje X.
195          public int getXB() {
196               return xB;
197          }
198
199          // Obtenemos la posición de la nave alienígena tipo B en ese momento en el eje Y.
200          public int getYB() {
201               return yB;
202          }
203
204          // Método para obtener la imagen de la nave alienígena tipo B.
205          public Image getImageB() {
206               return imageB;
207          }
208
209          // Método para obtener los límites de la imagen de la nave alienígena tipo B.
210          public Rectangle getBoundsB() {
211               return new Rectangle(xB, yB, widthB-10, heightB);
212           }
213  }
```