PRACTICA
PROGRAMACIÓN ORIENTADA A OBJETOS

OSCAR GONZÁLEZ TUR

kronosyo@gmail.com

Tlf.:678519113

JUNIO 2013

RTYPE

En este documento pretendo explicar como funciona y como está implementado este juego.

Podrá encontrar el diagrama de clases en el archivo DiagramaClases.pdf o pinchando aquí.

Adjunto el javadoc donde se explican los métodos de cada clase, puede abrirlo clickando aquí.

A continuación explicaré las clases que he creado y su función dentro del programa.

LAS CLASES

Menú

La clase principal del programa es la clase Menú, ya que contiene el método main y es donde se inicia la aplicación. El método main llama al constructor de Menú el cual crea 4 botones contenidos en un Jpanel y este a su vez contenido en un Jframe.

Cada botón tiene un texto que nos indica la dificultad de ese Nivel y al pulsar un botón el listener de dicho botón elimina la ventana de menú y crea un nuevo objeto rtype con los parámetros específicos de la dificultad que hemos seleccionado.

Rtype

Esta clase hereda de Jframe y va a ser el contenedor del Jpanel donde se mostrará la imagen del juego. Dentro de esta clase están las constantes del juego y algunos atributos principales. Los parámetros recibidos desde la llamada al constructor desde los botones del menú nos indican la velocidad de las naves alienígenas y el número que saldrán. El constructor también llamará a los métodos privados de la clase Rtype makeFrame() y centerScreen() los cuales se encargan de crear el Jframe y centrar el Jframe en la pantalla, respectivamente.

Vista

Esta clase hereda del Jpanel e implementa la interfaz ActionListener. Como atributos tiene un timer de la clase Timer y controller de la clase Controller.

El constructor de esta clase se encarga de establecer los ajustes del Jpanel. También añade un listener de la clase privada Tadapter que es la encargada de recibir las

pulsaciones del teclado. Después crea un objeto controller y otro objeto timer con los parámetros de delay y de observador y se inicia la variable aliensAdd de controller a 0 y se pone en marcha el timer.

A continuación se reescribe el método paint que se encarga de pintar todos los objetos que aparecen en pantalla y están activos con el método drawUfo que es un método privado implementado para no repetir código.

El método actionPerformed es el encargado de actuar a cada golpe de timer ejecutando los métodos que le hayamos implementado. En este caso son los siguientes: creamos un arraylist que coge los misiles lanzados por la nave y mediante un for each comprueba que no hayan colisionado con una de las naves aliens. También comprobamos mediante for each que la nave no colisione con ninguna nave alienigena. El método checkColision es de la clase controller y lo explicaré en la siguiente clase. Después mediante un if añadiremos naves alienígenas mediante el método addAlien siempre y cuando no haya mas naves que el numero de aliens del atributo numAliens de la clase Rtype.

Una vez comprobadas las colisiones y añadido el nuevo alien actualizamos los estados de los objetos que hay en pantalla mediante el método action, que moverá el objeto si está activo y lo eliminará si esta no activo. Actualizamos la posición de la nave y comprobamos si el juego ha terminado mediante el método finDelJuego parando el timer y creando la ventana final si ha terminado el juego, y si el juego no ha terminado llamamos al método repaint() para repintar todos los objetos que hemos modificado durante este actionPerformed.

El método privado drawUfo se encarga de pintar los objetos en pantalla mediante el método drawImage en su nueva ubicación y con su imagen correspondiente.

El método finalWindow es llamado en el actionPerformed cuando se comprueba que el juego a terminado y este método crea una ventana que te indica si has ganado o perdido según el mensaje que guarde en la variable mensajeFinal el método finDelJuego según sea el final. Es una ventana de opción JoptionPane con las opciones si y no y te pregunta si quieres volver a jugar, si le pulsas a si, marca el setVisible del Jframe como false y se llama al método dispose del Jframe para que sea eliminado por el garbage collector y crea un nuevo menu, en caso de que pulses no, cierra la aplicación mediante el método System.exit(0).

Está clase contiene una clase privada de soporte llamada Tadapter que hereda de la clase abstracta KeyAdapter. Su función es llamar al método adecuado según la pulsación de teclado que hagamos.

Y por último reescribimos el método getPreferredSize para ajustar el tamaño del

Jpanel con las constantes que establecimos en la clase Rtype.

Controller

Esta clase se encarga de las acciones generales del juego. Y contiene la nave aliada y el arraylist de aliens, los cuales crea el constructor.

Tenemos el método addAlien que crea un alien1, alien2 o ninguno según el resultado de un número aleatorio que creamos.

El método colision comprueba si dos objetos de la clase ufo han chocado, creando un rectángulo a partir del tamaño de la imagen que los representa y mediante el método intersects comprueba si dichos rectángulos se cortan por algún punto, es decir, intersectan. Como ampliación al anterior método está checkColision que comprueba un objeto ufo con toda una arraylist de objetos ufo usando el método colision.

Para actualizar los estados de los objetos existe el método action que comprueba en un arraylist de objetos ufo su estado y si están activos actualiza su posición con el método move de dichos objetos y si están no activos los elimina.

Para comprobar si el juego a terminado está el método booleano finDelJuego que nos devuelve verdad si ha terminado y false si no ha terminado, además hay dos formas de terminar, ganando o perdiendo y según sea guardará un mensaje en la variable string mensajeFinal de la clase Rtype. Por último un getter de arraylist de aliens.

Ufo

La clase Ufo es una clase abstracta que tiene los atributos y métodos generales de sus clases hijas Alien, Rocket y Craft. Los atributos son sus coordenadas x e y, su desplazamiento dx y dy. La altura “height” y el ancho “width” del objeto. La imagen tipo Image y tipo ImageIcon llamadas image e im respectivamente. Un string ufo el cual contendrá el nombre del objeto que es. Una variable de estado booleana llamada activo que nos indicará si el objeto debe aparecer en pantalla o debe ser eliminado.

Y por último un HashMap<String,String> llamado pictures que guardará la imagen de cada objeto según sea su nombre contenido en la variable ufo. Estos atributos tendrán el modificador protected.

El constructor marcará como activo el objeto y creará el HashMap de clave valor con los nombres de los objeto posibles como clave y el nombre de la imagen de cada objeto como valor.

El método move es abstracto y será implementado en cada hijo de la clase Ufo.

ImageResource es un método que entrando el nombre del objeto como parámetro nos asignará la imagen correspondiente al objeto.

Por último tenemos 4 getters de los atributos x, y, activo e image y un setter del atributo activo.

Como clases hijas de Ufo tenemos las siguientes:

Alien

Para esta clase que hereda de Ufo le añadimos un atributo booleano que es typeAlien, y nos indicará si la nave es tipo 1 con true o tipo 2 con false. Lo pasaremos como parámetro al constructor a la hora de crear un alien para definir el tipo de alien que queremos crear. El constructor llamará al constructor de la clase padre Ufo y además inicializará los atributos x, y(este de forma aleatoria), dx, dy, width, height y ufo. Y llamaremos al método imageResource para asignar la imagen correspondiente.

Por último sobrescribiremos la el método move con la implementación del movimiento que realizará la nave, según sea tipo 1 o tipo 2. La única diferencia es que la tipo dos además de desplazarse horizontalmente como la tipo 1 se desplazará verticalmente arriba y abajo de forma aleatoria, sin salirse de los margenes de la pantalla superior e inferior. Cuando una nave salga de la pantalla por el lateral izquierdo reaparecerá por el lateral derecho de nuevo.

Rocket

Al igual que la clase Alien el constructor de rocket llamará al constructor de la clase padre y llamará al método imageResource con el parámetro ufo que habrá inicializado antes junto a los atributos x e y.

También sobrescribiremos el método move indicando el desplazamiento horizontal y una condición de que si sale de la pantalla se marcará como inactivo para su posterior eliminación.

Craft

La clase Craft añade a parte de los atributos de su padre un arraylist de misiles el cual contendrá los misiles que ha disparado y no han sido eliminados.

El constructor llamará al constructor de ufo e inicializa los atributos, asigna la imagen mediante imageResource y crea el arraylist de misiles.

El método move es sobrescrito asignando x e y a través del método privado clamp el cual suma a nuestra coordenada x e y el desplazamiento dx y dy a cada pulsación de la tecla correspondiente, marcando un valor máximo y mínimo para cada coordenada para que se mantenga dentro de los limites. El método clamp funciona como limite, es decir, cuando el valor x+dx o y+dy supera alguno de los dos rangos máximo o mínimo devuelve el valor limite.

El método keyPressed modifica las variables de desplazamiento dx y dy según la tecla que presionemos asignando la constante CRAFTSPEED positiva o negativa según la dirección en la que se mueve. La tecla espacio llama al método shoot que dispara un misil creando un nuevo misil y añadiéndolo al arraylist de misiles.

Para que al dejar de pulsar se pare la nave implementamos el método keyReleased

poniendo las variables dx y dy a 0 al soltar las teclas que hemos pulsado.

Y por último tenemos un getter del arraylist de misiles de la nave.

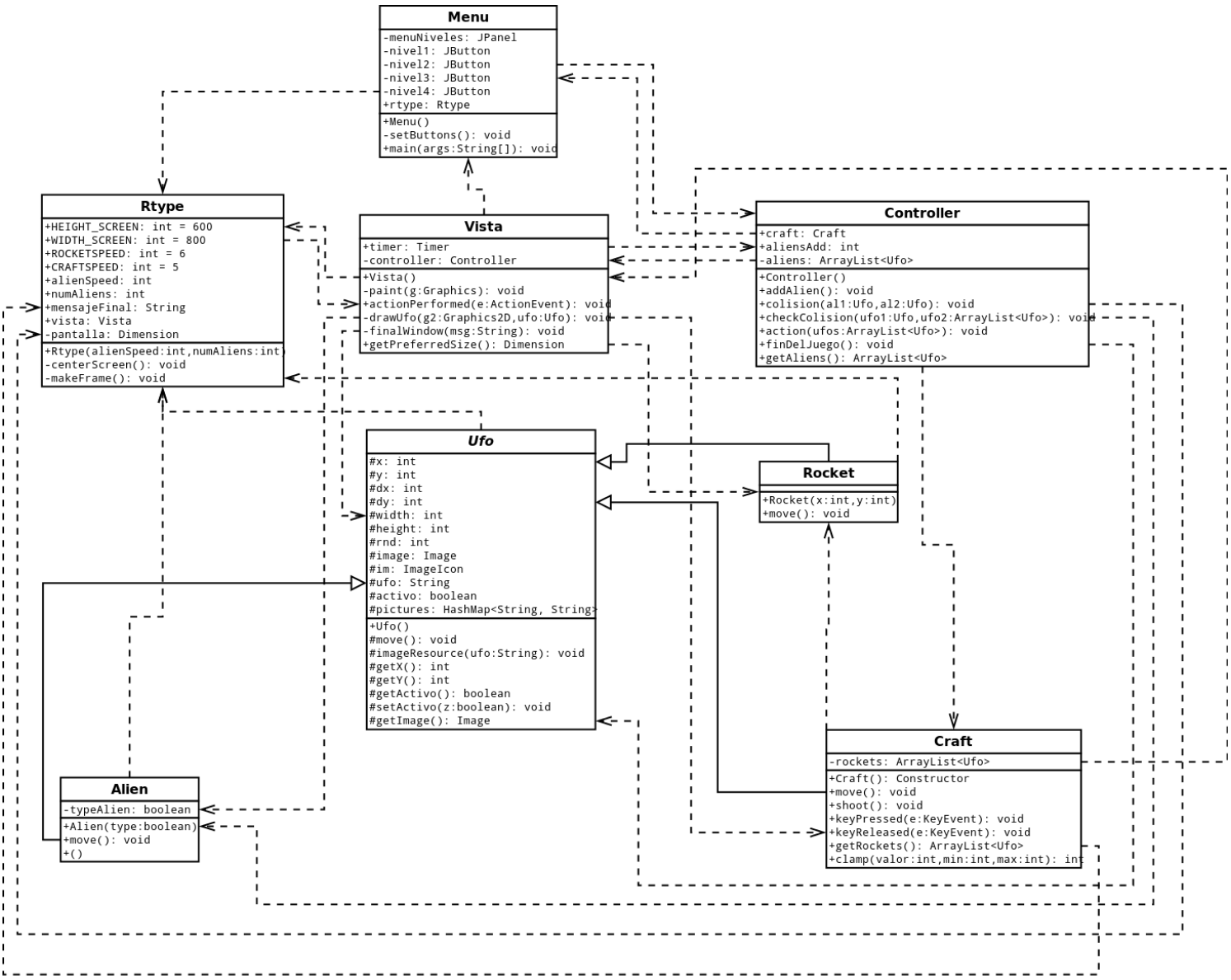
DISEÑO

Bueno el diseño está dividido de la siguiente manera. En primer lugar tenemos menu que es un Jframe con los botones de selección de nivel que nos crea un objeto de la clase Rtype. Luego rtype es el Jframe de la ventana de juego y en el se contiene el Jpanel vista y también los atributos generales del juego. En el jpanel vista se dibujan todos los objetos del juego(todos los hijos de la clase abstracta Ufo) y es el listener de los eventos de teclado y del timer. La interacción de los objetos del juego también se ejecuta desde vista, pero llamando a los metodos de la clase controller la cual se encarga de actualizar los atributos de los objetos del juego, para que vista los repinte en pantalla con sus nuevos estados.

Los objetos del juego son los hijos de la clase abstracta Ufo, la cual contiene los metodos y atributos comunes a los 3 hijos que son Alien, Rocket y Craft. Cada objeto contiene sus atributos de estado y sus metodos comunes y especificos.

DIAGRAMA DE CLASES

(Haga clic sobre la imagen para abrir el DiagramaClases.pdf)



ANEXO

CODIGO FUENTE DEL JUEGO

(se han omitido los comentarios del codigo fuente original)

CLASE MENU

```
import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.*;
```

```
import java.awt.*;
import java.awt.Dimension;

public class Menu extends JFrame
{
    private JPanel menuNiveles;
    private JButton nivel1, nivel2, nivel3, nivel4;
    public static Rtype rtype;

    public Menu()
    {
        menuNiveles = new JPanel();
        nivel1=new JButton();
        nivel2=new JButton();
        nivel3=new JButton();
        nivel4=new JButton();
        setButtons();
        menuNiveles.setLayout(new GridLayout(4,1));
        setResizable(false);
        setTitle("Rtype Nivel");
        setSize(new Dimension(200, 400));
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        add(menuNiveles);
        setVisible(true);
    }
```

```
private void setButtons()
{
    nivel1.setText("FACIL");
    nivel2.setText("NORMAL");
    nivel3.setText("COMPLICADO");
    nivel4.setText("IMPOSIBLE");

    nivel1.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent e){
            rtype = new Rtype(2,10);
            setVisible(false);
            dispose();
        }
    });
    nivel2.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent e){
            rtype = new Rtype(4,15);
            setVisible(false);
            dispose();
        }
    });
    nivel3.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent e){
            rtype = new Rtype(6,20);
            setVisible(false);
            dispose();
```

```
            }
        });
        nivel4.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent e){
                rtype = new Rtype(12,30);
                setVisible(false);
                dispose();
            }
        });
        menuNiveles.add(nivel1);
        menuNiveles.add(nivel2);
        menuNiveles.add(nivel3);
        menuNiveles.add(nivel4);

    }

    public static void main(String [] args){
        Menu menu = new Menu();
    }

}
```

CLASE RTYPE

```
import javax.swing.JFrame;
import java.awt.Color;
import java.awt.*;
import java.awt.Toolkit;

public class Rtype extends JFrame
{
    public static final int HEIGHT_SCREEN= 600;
    public static final int WIDTH_SCREEN = 800;
    public static final int ROCKETSPEED=6;
    public static final int CRAFTSPEED=5;
    public static int alienSpeed;
    public static int numAliens;
    public static String mensajeFinal;
    public static Vista vista;
    private Dimension pantalla;

    public Rtype(int alienSpeed,int  numAliens)
    {
      makeFrame();
      centerScreen();
      this.alienSpeed=alienSpeed;
      this.numAliens=numAliens;
    }
```

```
    private void centerScreen(){
        pantalla = Toolkit.getDefaultToolkit().getScreenSize();
      int height = (int) pantalla.getHeight()-HEIGHT_SCREEN;
      int width = (int) pantalla.getWidth()-WIDTH_SCREEN;
      this.setLocation(width/2,height/2);
    }

    private void makeFrame(){
      vista = new Vista();
      this.setLayout(new BorderLayout());
      this.getContentPane().add(vista, BorderLayout.CENTER);
      this.setTitle("R-Type");
      this.setResizable(false);
      this.setDefaultCloseOperation(EXIT_ON_CLOSE);
      this.pack();
      this.setVisible(true);
    }


}
```

CLASE VISTA

```
import javax.swing.*;
import java.awt.event.*;
import javax.swing.Timer;
import java.awt.*;
```

```
import java.util.ArrayList;
import java.util.Random;

public class Vista extends JPanel implements ActionListener
{
    public Timer timer;
    private Controller controller;

    public Vista()
    {
        setFocusable(true);
        setBackground(Color.black);
        setDoubleBuffered(true);
        setVisible(true);
        addKeyListener (new TAdapter());
        controller = new Controller();
        controller.aliensAdd = 0;
        timer = new Timer(5, this);
        timer.start();
     }

    public void paint(Graphics g)
    {
        super.paint(g);
        Graphics2D g2d = (Graphics2D)g;
        drawUfo(g2d,controller.craft);
```

```
    ArrayList<Ufo> ro = controller.craft.getRockets();
    for(Ufo rocket: ro){
        drawUfo(g2d,rocket);
    }
    for (Ufo alien: controller.getAliens()){
        drawUfo(g2d, alien);
    }
    Toolkit.getDefaultToolkit().sync();
    g.dispose();
}



public void actionPerformed(ActionEvent e)
{
    ArrayList<Ufo> ro = controller.craft.getRockets();
    for(Ufo rocket: ro){
        controller.checkColision(rocket,controller.getAliens());
    }
    controller.checkColision(controller.craft, controller.getAliens());
    if(controller.aliensAdd < Rtype.numAliens){
        controller.addAlien();
    }
    controller.action(ro);
    controller.action(controller.getAliens());
    controller.craft.move();
    if(controller.finDelJuego())
```

```
        {
            timer.stop();
            finalWindow(Rtype.mensajeFinal);
        }
        repaint();
    }

    private void drawUfo(Graphics2D g2,Ufo ufo){
        g2.drawImage(ufo.getImage(),ufo.getX(), ufo.getY(),this);
    }
    private void finalWindow(String msg)
    {
        int victory= JOptionPane.showConfirmDialog(this, msg + "¿Quieres volver a
jugar?","Fin del Juego", JOptionPane.YES_NO_OPTION);
        switch(victory){
            case 0:
            Menu.rtype.setVisible(false);
            Menu.rtype.dispose();
            Menu menu = new Menu();
            break;
            case 1:
            System.exit(0);
            break;
        }
    }
```

```
    private class TAdapter extends KeyAdapter
    {
        public void keyPressed(KeyEvent e){
            controller.craft.keyPressed(e);
        }
        public void keyReleased(KeyEvent e){ /
            controller.craft.keyReleased(e);
        }
    }

    @Override
    public Dimension getPreferredSize(){
        Dimension dimension = new
Dimension(Rtype.WIDTH_SCREEN,Rtype.HEIGHT_SCREEN);
        return dimension;
    }
}
```

CLASE CONTROLLER

```
import javax.swing.*;
import java.awt.*;
import java.util.ArrayList;
import java.util.Random;
```

```
public class Controller
{
    public Craft craft;
    public int aliensAdd;
    private ArrayList<Ufo> aliens;

    public Controller()
    {
        aliens = new ArrayList<Ufo>();
        aliensAdd = 0;
        craft = new Craft();
    }

    public void addAlien(){
      int rd;
      rd = new Random().nextInt(1000);
       if(rd<3){
         aliens.add(new Alien(false));
         aliensAdd++;
         } else if(rd>985) {
           aliens.add(new Alien(true));
           aliensAdd++;
        }
    }

    public void colision(Ufo al1, Ufo al2){
```

```
        Rectangle r1 = new Rectangle( al1.getX(), al1.getY(),
al1.getImage().getWidth(null), al1.getImage().getHeight(null));
        Rectangle r2 = new Rectangle( al2.getX(), al2.getY(),
al2.getImage().getWidth(null), al2.getImage().getHeight(null));
        if(r1.intersects(r2)){
            al1.setActivo(false);
            al2.setActivo(false);
        }
    }

    public void checkColision(Ufo ufo1, ArrayList<Ufo> ufo2){
        for(Ufo ufo:ufo2){
            colision(ufo1,ufo);
        }
    }

    public void action(ArrayList<Ufo> ufos){
        for(int i=0; i<ufos.size(); i++){
            Ufo ufo = (Ufo) ufos.get(i);
            if(ufo.getActivo()){
                ufo.move();
            }else {
                ufos.remove(i);
            }
        }
    }
```

```
    public boolean finDelJuego(){
        if(craft.activo==true && aliens.size()==0 && aliensAdd== Rtype.numAliens)
        {
            Rtype.mensajeFinal= " ¡¡ENHORABUENA, HAS GANADO!! ";
            return true;
        } else if(craft.activo==false){
            Rtype.mensajeFinal= " HAS PERDIDO ";
            return true;
        }else {
            return false;
        }
    }

    public ArrayList<Ufo> getAliens()
    {
        return aliens;
    }
}
```

CLASE ABSTRACTA UFO

```
import java.awt.Image;
import javax.swing.ImageIcon;
import java.util.Random;
import java.util.ArrayList;
import java.util.HashMap;

public abstract class Ufo
```

```
{
    protected int x;
    protected int y;
    protected int dx;
    protected int dy;
    protected int width;
    protected int height;
    protected int rnd;
    protected Image image;
    protected ImageIcon im;
    protected String ufo;
    protected boolean activo;
    protected HashMap<String,String> pictures;

    public Ufo()
    {
        activo=true;

        pictures = new HashMap<String,String>();
        pictures.put("craft", "nave.png");
        pictures.put("alien1", "alien1p.png");
        pictures.put("alien2", "alien2p.png");
        pictures.put("rocket", "misil.png");

    }
```

```
protected abstract void move();

protected void imageResource(String ufo)
{
    im = new ImageIcon(getClass().getResource(pictures.get(ufo)));
    image = im.getImage();
}

protected int getX(){
    return x;
}

protected int getY(){
    return y;
}

protected boolean getActivo(){
    return activo;
}

protected void setActivo(boolean z){
    activo=z;
}

protected Image getImage(){
    return image;
```

```
    }

}
```

CLASE ALIEN

```
import java.util.Random;
import javax.swing.ImageIcon;

public class Alien extends Ufo
{
    private boolean typeAlien;

    public Alien(boolean type)
    {
        super();
        typeAlien = type;
        x = Rtype.WIDTH_SCREEN;
        this.dx=Rtype.alienSpeed;
        if(type==true)
        {
            ufo = "alien1";
        } else {
            ufo = "alien2";
            dy = Rtype.alienSpeed;
        }
        imageResource(ufo);
        rnd = new Random().nextInt(Rtype.HEIGHT_SCREEN-
```

```
(im.getIconHeight()+50));
        y = rnd;
        width = im.getIconWidth();
        height = im.getIconHeight();
    }

    @Override
    public void move()
    {
      if(typeAlien==true)
      {
        if(x<-width){
            x=Rtype.WIDTH_SCREEN;
        }else {
        x-=dx;
        }
      } else {
        if((this.y<=0) || (this.y>Rtype.HEIGHT_SCREEN-height)){
            this.dy=-dy;
        } else {
          int rd = new Random().nextInt(100);
          if(rd<2){
            dy = -dy;
          }
       }
        if(this.x<-width){
```

```
        this.x=Rtype.WIDTH_SCREEN;
        }else {
        this.x-=dx;
        this.y+=dy;
        }

     }
   }
}
```

CLASE ROCKET

```
import javax.swing.ImageIcon;

public class Rocket extends Ufo
{

   public Rocket(int x, int y)
   {
      super();
      ufo = "rocket";
      imageResource(ufo);
      this.x = x;
      this.y = y;
   }
```

```
    @Override
    public void move()
    {
        if(this.x>Rtype.WIDTH_SCREEN){
            this.activo=false;
        } else {
        x+=Rtype.ROCKETSPEED;
        }
    }
}
```

CLASE CRAFT

```
import java.awt.event.KeyEvent;
import java.util.ArrayList;
import javax.swing.ImageIcon;

public class Craft extends Ufo
{
    private ArrayList<Ufo> rockets;

    public Craft()
    {
        super();
        ufo = "craft";
        imageResource(ufo);
```

```
    this.x=20;
    this.y = Rtype.HEIGHT_SCREEN / 2;
    rockets = new ArrayList<Ufo>();
    width=im.getIconWidth();
    height=im.getIconHeight();
}

@Override
public void move()
{
    x = clamp(x+dx, 0, Rtype.WIDTH_SCREEN - width);
    y = clamp(y+dy, 0, Rtype.HEIGHT_SCREEN - height);
}

public void shoot()
{
    Rocket rocket = new Rocket(getX()+width,getY()+height/2);
    rockets.add(rocket);
}

public void keyPressed ( KeyEvent e)
{
    int key = e.getKeyCode();

    if(key==KeyEvent.VK_Q){
        this.dy = -Rtype.CRAFTSPEED;
    }else
```

```
    if(key==KeyEvent.VK_A){
            this.dy = Rtype.CRAFTSPEED;
    }else
    if(key==KeyEvent.VK_O){
            this.dx=-Rtype.CRAFTSPEED;
    }else
    if(key==KeyEvent.VK_P){
            this.dx=Rtype.CRAFTSPEED;
    }else
    if(key==KeyEvent.VK_SPACE){
        shoot();
    }
}
```

```
public void keyReleased (KeyEvent e)
{
    int key = e.getKeyCode();

    if(key==KeyEvent.VK_Q){
        dy = 0;
```

```
        }else
        if(key==KeyEvent.VK_A){
            dy =0;
        }else
        if(key==KeyEvent.VK_O){
            dx=0;
        }else
        if(key==KeyEvent.VK_P){
            dx=0;
        }
    }

    public ArrayList<Ufo> getRockets(){
        return rockets;
    }

    private int clamp(int valor, int min, int max) {
    if (valor > max)
        return max;
    if (valor < min)
        return min;
    return valor;
    }
}
```