Código del juego

```
import greenfoot.Actor;
import greenfoot.Greenfoot;
import greenfoot.GreenfootImage;
import java.awt.Color;

import java.util.List;
import java.util.ArrayList;

/**
 * Escribe una descrición de la clase Barricada aquí.
 *
 * @autor (tu nombre)
 * @versión (Un número de versión o una fecha)
 */
public class Barricada extends Actor
{
   private int health = 100;
   private boolean hurt = true;

   /**
    * Act - hace lo que Barricada quiere hacer. Este método se llama "cuando quiera" o
whenever
    * los botones 'Actuar or 'Ejecutar' son presionados en el entorno.
    */
   public void act()
   {
    // Agrega tus códigos de acción aquí
      Actor a = getOneIntersectingObject(Necrofago.class);
      Actor b = getOneIntersectingObject(Zombie.class);
      Actor c = getOneIntersectingObject(ZRadiactivo.class);
      Actor d = getOneIntersectingObject(Jefe.class);

      if (a != null)
      {
       hurt(2);
       hurt = true;
      }

      if (b != null)
      {
       hurt(4);
       hurt = true;
      }

      if (c != null)
```

```
        {
         hurt(1);
         hurt = true;
        }

         if (d != null)
        {
         hurt(1);
         hurt = true;
        }
        if(health <= 0)
          //getWorld().removeObject(Barricada;
          removeTouching(Barricada.class);
    }

    public Barricada()
    {
     setImage("Barricada.png");

    }
    public void hurt(int amount)
    {
        health -= amount;
        int size = Greenfoot.getRandomNumber(5)+1;
        for (int i = 0; i < size; i++)
        {
            int rot = Greenfoot.getRandomNumber(360);
            int spd = 2;
        }
    }
    public int getHealth()
    {
     return health;
    }
}
import greenfoot.Actor;
import greenfoot.World;
import greenfoot.Greenfoot;
import greenfoot.GreenfootImage;

import java.awt.Color;

public class Bullet extends Actor
{
    private int speed = 150;
    private boolean fromShotgun = false;

    public Bullet(boolean shot, int speed)
```

```
    {
        GreenfootImage img = new GreenfootImage(4,2);
        img.setColor(Color.yellow);
        img.fill();
        setImage(img);
        fromShotgun = shot;
        this.speed = speed;
    }

    public void act()
    {
        Actor a = moved(speed);
        if (getX() > getWorld().getWidth() || getX() < 0 || getY() < 0 || getY() >
getWorld().getHeight())
            getWorld().removeObject(this);
    }

    public Actor moved(int amount)
    {
        int x1 = getX();
        int y1 = getY();
        move(amount);
        int x2 = getX();
        int y2 = getY();
        setLocation(x1,y1);
        Actor a = null;
        for (int i = 0; i < amount; i +=5)
        {
            move(5);
            a = getOneIntersectingObject(Zombie.class);
            if (a != null)
                break;
        }
        setLocation(x2,y2);
        return a;
    }
}
import greenfoot.*;


public class Buttons extends Actor
{
    private int speedX;
    private GreenfootSound clickSound;

    /**
     *
     */
```

```
    public Buttons()
    {
        speedX = 0;
        clickSound = new GreenfootSound("buttonSound.wav");
    }

    /**
     *
     */
    public void act()
    {
        efeitosEntrada();
    }

    /**
     *
     */
    private void efeitosEntrada()
    {
        if(getX() < getWorld().getWidth()/2)
            move(speedX);
        else
            setLocation(getWorld().getWidth()/2, getY());
        if(speedX < 10)
            speedX++;
    }

    /**
     *
     */
    protected GreenfootSound getClickSound()
    {
        return clickSound;
    }
}
import greenfoot.*;

/**
 * Escribe una descrición de la clase Credits aquí.
 *
 * @autor (tu nombre)
 * @versión (Un número de versión o una fecha)
 */
public class Credits extends Actor
{
    /**
     * Act - hace lo que Credits quiere hacer. Este método se llama "cuando quiera" o whenever
     * los botones 'Actuar or 'Ejecutar' son presionados en el entorno.
```

```
     */
    public void act()
    {
        // Agrega tus códigos de acción aquí.
    }
}
import greenfoot.*;
import java.awt.Color;
import java.awt.Font;

/**
 * Write a description of class Credits here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Creditsbt extends Buttons
{
    private GreenfootImage imgCredits;
    private Color amarelo;
    private Color laranja;
    private Color laranjaEscuro;
    private Credits credits;

    /**
     * Construtor da classe Credits
     */
    public Creditsbt()
    {
        drawBox();
        printText();
        imgCredits.setTransparency(0);
        credits = new Credits();
    }

    /**
     * Act - do whatever the Credits wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public void act()
    {
        super.act();
        imgCredits.setTransparency(255);
        click();
    }

    /**
     *
```

```
    */
    private void printText()
    {
        laranjaEscuro = new Color(214,95,0);
        imgCredits.setColor(laranjaEscuro);
        Font play = new Font("sanserif",Font.BOLD,30);
        imgCredits.setFont(play);
        int x = imgCredits.getWidth()-150;
        int y = imgCredits.getHeight()-16;
        imgCredits.drawString("Credits",x,y);
    }

    /**
     *
     */
    private void drawBox()
    {
        amarelo = new Color(255,188,0);
        laranja = new Color(255,133,0);
        imgCredits = getImage();
        imgCredits.setColor(laranja);
        imgCredits.fill();
        imgCredits.scale(200,50);
        imgCredits.setColor(amarelo);
        int margem = 5;
        int largura = imgCredits.getWidth()-2*margem;
        int altura = imgCredits.getHeight()-2*margem;
        imgCredits.fillRect(margem,margem,largura,altura);
    }

    /**
     *
     */
    public void click()
    {
        //processo analogo ao explicado na classe Helpbt
        if(Greenfoot.mouseClicked(this)){
            getClickSound().play();
            getWorld().addObject(credits,getWorld().getWidth()/2,getWorld().getHeight()/2);
            getWorld().addObject(new Exit(credits),credits.getX() -
credits.getImage().getWidth()/2,credits.getY() - credits.getImage().getHeight()/2);
        }
    }
}
import greenfoot.*;  // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)
import java.awt.Color;
import java.awt.Font;
/**
```

```
 * Write a description of class DesignScoreboard here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class DesignScoreboard extends Actor
{
   GreenfootImage titulo = new GreenfootImage(300,100);
   private Color verde;

   /**
    * Act - do whatever the DesignScoreboard wants to do. This method is called whenever
    * the 'Act' or 'Run' button gets pressed in the environment.
    */
   public void act()
   {
      // Add your action code here.
   }

   /**
    * Construtor da classe DesignScoreboard() - simplesmente faz display do titulo do
scoreboard
    */
   public DesignScoreboard()
   {
      printTitle();
   }

   /**
    * printTitle() - titulo do Scoreboard
    */
   private void printTitle()
   {
      Color verde = new Color(196,223,155);
      titulo.setColor(verde);
      titulo.setFont(new Font("sanserif",Font.BOLD,50));
      titulo.drawString("Scoreboard",10,50);
      setImage(titulo);
   }
}
import greenfoot.Actor;
import java.awt.Point;
import java.util.List;

public class Distance
{
   /**
    * Finds the distance between two Points
```

```
 * @param a Point a.
 * @param b Point b.
 * @return The distance between the two points.
 */
public static double distanceBetween(Point a, Point b)
{
   double x = a.getX() - b.getX();
   double y = a.getY() - b.getY();

   return Math.abs(Math.sqrt(x*x+y*y));
}

/**
 * Finds the distance between two coordinates
 * @param x1 The first x
 * @param y1 The first y
 * @param x2 The second x
 * @param y2 The second y
 */
public static double distanceBetween(int x1, int y1, int x2, int y2)
{
   return distanceBetween(new Point(x1, y1), new Point(x2, y2));
}

/**
 * Finds the distance between two actors' centers.
 * @param a Actor a.
 * @param b Actor b.
 * @return The distance between the center of Actor a, and the center of Actor b
 */
public static double distanceBetween(Actor a, Actor b)
{
   return distanceBetween(a.getX(), a.getY(), b.getX(), b.getY());
}

/**
 * Finds the Point the least amount away from a Point of reference
 * Precondition: pnts.size() > 0
 * @param a The reference Point.
 * @param pnts The points to find the distances of
 * @return The Point the closest to the reference Point
 */
public static Point closestTo(Point a, List<Point> pnts)
{
   Point closest = null;
   double min = Double.MAX_VALUE;

   for (Point p : pnts)
```

```
    {
        double distance = distanceBetween(a, p);
        if (distance < min)
        {
            min = distance;
            closest = p;
        }
    }

    return closest;
}

/**
 * Finds the Point farthest away from a Point of reference.
 * Precondition: pnts.size() > 0
 * @param a The reference Point
 * @param pnts The points to find the distances of
 * @return The Point the farthest away from the reference Point
 */
public static Point farthestFrom(Point a, List<Point> pnts)
{
    Point farthest = null;
    double max = Double.MIN_VALUE;

    for (Point p : pnts)
    {
        double distance = distanceBetween(a, p);
        if (distance > max)
        {
            max = distance;
            farthest = p;
        }
    }

    return farthest;
}

/**
 * Finds the Actor closest to an Actor of reference.
 * Precondition: pnts.size() > 0
 * @param a The reference Actor
 * @param acts The Actors to find the distances of
 * @return The Actor the closest to the Reference Actor
 */
public static Actor closestTo(Actor a, List<Actor> acts)
{
    Actor closest = null;
    double min = Double.MAX_VALUE;
```

```
        for (Actor act : acts)
        {
            double distance = distanceBetween(a, act);
            if (distance < min)
            {
                min = distance;
                closest = act;
            }
        }

        return closest;
    }

    /**
     * Finds the Actor farthest to an Actor of reference.
     * Precondition: pnts.size() > 0
     * @param a The reference Actor
     * @param acts The Actors to find the distances of
     * @return The Actor the farthest to the Reference Actor
     */
    public static Actor farthestFrom(Actor a, List<Actor> acts)
    {
        Actor farthest = null;
        double max = Double.MIN_VALUE;

        for (Actor act : acts)
        {
            double distance = distanceBetween(a, act);
            if (distance > max)
            {
                max = distance;
                farthest = act;
            }
        }

        return farthest;
    }
}
import greenfoot.*;


public class Exit extends Buttons
{
    private Actor actor;


    public Exit(Actor a)
```

```
    {
        actor = a;
    }


    public void act()
    {
        click();
    }


    private void click()
    {
        if(Greenfoot.mouseClicked(this)){
            getClickSound().play();
            getWorld().removeObject(actor);
            getWorld().removeObject(this);
        }
    }
}
import greenfoot.*;  // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)
import java.awt.Color;
import java.awt.Font;
/**
 * Write a description of class FirstPlace here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class FirstPlace extends ScoreLoad
{
    GreenfootImage primeiroLugar = new GreenfootImage(400,50);
    private Color verde;

    /**
     * Act - do whatever the DesignScoreboard wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public void act()
    {
        // Add your action code here.
    }

    /**
     * Construtor da clase FirstPlace, simplesmente faz clear na imagem predefinida e define
uma nova imagem
     */
    public FirstPlace()
```

```
    {
        display();
    }

    /**
     * display() - faz display da melhor pontuação alguma vez obtida
     */
    private void display()
    {
        Color verde = new Color(196,223,155);
        primeiroLugar.setColor(verde);
        primeiroLugar.setFont(new Font("sanserif",Font.BOLD,25));
        String score;
        if(getList().isEmpty()) //quando a lista está vazia metemos o score a 0.
            score = "0";
        else
            score = (String)getList().get(0);//caso contrario, vai buscar o 1 elemento da lista dando
obviamente a  melhor pontuacao de sempre porque a lista esta por ordem descrente
        primeiroLugar.drawString("Best score: " + score,150,primeiroLugar.getHeight()/2);

        setImage(primeiroLugar);
    }
}
import greenfoot.*;  // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)
import java.awt.Color;
import java.awt.Font;
/**
 * Write a description of class FirstPlace here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class FourthPlace extends ScoreLoad
{
    GreenfootImage quartoLugar = new GreenfootImage(400,50);
    private Color laranjaEscuro;

    /**
     * Act - do whatever the DesignScoreboard wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public void act()
    {
        // Add your action code here.
    }

    /**
```

```
    * Construtor da clase FourthPlace, simplesmente faz clear na imagem predefinida e define
uma nova imagem
    */
   public FourthPlace()
   {
      display();
   }

    /**
    * display() - faz display da quarta melhor pontuação alguma vez obtida
    */
   private void display()
   {
      laranjaEscuro = new Color(214,95,0);
      quartoLugar.setColor(laranjaEscuro);
      quartoLugar.setFont(new Font("sanserif",Font.BOLD,25));
      String score;
      if(getList().size()<4) //se o tamanho da lista for menor que 4 entao significa que ainda o
utilizador ainda nao jogou quatro vezes portanto nao existe quarta melhor pontuação
         score = "0";
      else
         score = (String)getList().get(3);//caso contrario, vai buscar o 4 elemento da lista dando
obviamente a quarta melhor pontuacao de sempre porque a lista esta por ordem descrente
      quartoLugar.drawString("Fourth best: "+ score,150,quartoLugar.getHeight()/2);

      setImage(quartoLugar);
   }
}
import greenfoot.*;
import greenfoot.World;
import greenfoot.Greenfoot;
import greenfoot.GreenfootImage;
import java.awt.Color;
import greenfoot.core.WorldHandler;
import javax.swing.JPanel;
import java.awt.Point;
import java.awt.Toolkit;
/**
 * Escribe una descrición de la clase Escenario aquí.
 *
 * @autor (tu nombre)
 * @versión (Un número de versión o una fecha)
 */
public class Game extends World
{
   GreenfootSound backgroundMusic = new GreenfootSound("Music.mp3");
   private int ztimer = 0;
   private int ttimer = 0;
```

```
   private int ctimer = 0;
   private int speed = 1;
   private int maxTime = 90;
   private int score = 0;

   //Posicion del marcador
   private static final int RADS_POS_X = 50;
   private static final int RADS_POS_Y = 15;

   private Rads rads = new Rads();
   int point = 0;
   //private Rads radiacion = new Radiacion();
   /**
    * Constructor para objetos de clase Ciudad.
    *
    */
   public Game()
   {
      // Crea un nuevo mundo de 600x400 celdas con un tamaño de celda de 1x1 pixeles.
      super(650, 750, 1);

      // Esta linea se encarga de poner el actor (objeto de la clase Nave) en su posicion inicial
      addObject(rads, RADS_POS_X, RADS_POS_Y);

      GreenfootImage img = new GreenfootImage(900, 600);
      img.fill();
      //Posicion del jugador
      addObject(new Player(),getWidth()/2,getHeight()/2);
      addObject(new Score(),79,61);
      addObject(new Barricada(),getWidth()/2,getHeight()/2);

setPaintOrder(Player.class,Zombie.class,Necrofago.class,ZRadiactivo.class,Jefe.class,Bullet.clas
s);
      setActOrder(Player.class,
Zombie.class,Necrofago.class,ZRadiactivo.class,Jefe.class,Bullet.class);
      for(int i=0; i<3; i++)
      {
         RadAway radAway = new RadAway();
         int x = Greenfoot.getRandomNumber(getWidth());
         int y = Greenfoot.getRandomNumber(getHeight());
         addObject(radAway, x, y);
      }

   }
   //Empieza el acto
   public void act()
   {
    getPlayer();
```

```
    if (Greenfoot.getMouseInfo() == null)
        return;
    //updateVars();
    if (ztimer <= 0)
        releaseASquare();
    else
        ztimer--;

    if (Greenfoot.getMouseInfo() == null)
        return;
    //updateVars();
    if (ttimer <= 0)
        releaseASquare();
    else
        ttimer--;

    if (Greenfoot.getMouseInfo() == null)
        return;
    //updateVars();
    if (ctimer <= 0)
        releaseASquare();
    else
        ctimer--;
}
//Comienza musica
public void started()
{
    backgroundMusic.playLoop();
}
//Nuevo jugador
public Player getPlayer()
{
    return new Player();
}
//Liberad los zombies XD
public void releaseASquare()
{
    Zombie z = new Zombie();
    Necrofago t = new Necrofago();
    ZRadiactivo c = new ZRadiactivo();
    Jefe b = new Jefe();
    Superviviente s = new Superviviente();

    addObject(z, 1, 1);
    while (z.getX() < getWidth() && z.getX() > 0 && z.getY() < getHeight() && z.getY() > 0)
    {
```

```
   int x = Greenfoot.getRandomNumber(700)+300;
   if (Greenfoot.getRandomNumber(2)+1 == 1)
      x = -x;
   int y = Greenfoot.getRandomNumber(700)+300;
   if (Greenfoot.getRandomNumber(2)+1 == 1)
      y = -y;
   z.setLocation(x,y);
}
ztimer = Greenfoot.getRandomNumber(400);

addObject(t, 1, 1);
while (t.getX() < getWidth() && t.getX() > 0 && t.getY() < getHeight() && t.getY() > 0)
{
   int x = Greenfoot.getRandomNumber(700)+300;
   if (Greenfoot.getRandomNumber(2)+1 == 1)
      x = -x;
   int y = Greenfoot.getRandomNumber(700)+300;
   if (Greenfoot.getRandomNumber(2)+1 == 1)
      y = -y;
   t.setLocation(x,y);
}
ttimer = Greenfoot.getRandomNumber(8000);

addObject(c, 1, 1);
while (c.getX() < getWidth() && c.getX() > 0 && c.getY() < getHeight() && c.getY() > 0)
{
   int x = Greenfoot.getRandomNumber(700)+300;
   if (Greenfoot.getRandomNumber(2)+1 == 1)
      x = -x;
   int y = Greenfoot.getRandomNumber(700)+300;
   if (Greenfoot.getRandomNumber(2)+1 == 1)
      y = -y;
   c.setLocation(x,y);
}
ctimer = Greenfoot.getRandomNumber(4000);

addObject(b, 1, 1);
while (b.getX() < getWidth() && b.getX() > 0 && b.getY() < getHeight() && b.getY() > 0)
{
   int x = Greenfoot.getRandomNumber(700)+300;
   if (Greenfoot.getRandomNumber(2)+1 == 1)
      x = -x;
   int y = Greenfoot.getRandomNumber(700)+300;
   if (Greenfoot.getRandomNumber(2)+1 == 1)
      y = -y;
   b.setLocation(x,y);
}
ctimer = Greenfoot.getRandomNumber(1000);
```

```
      addObject(s, 1, 1);
      while (s.getX() < getWidth() && s.getX() > 0 && s.getY() < getHeight() && s.getY() > 0)
      {
         int x = Greenfoot.getRandomNumber(700)+300;
         if (Greenfoot.getRandomNumber(2)+1 == 1)
            x = -x;
         int y = Greenfoot.getRandomNumber(700)+300;
         if (Greenfoot.getRandomNumber(2)+1 == 1)
            y = -y;
         s.setLocation(x,y);
      }
      ctimer = Greenfoot.getRandomNumber(10000);
   }
   //Puntos
   public void point(int amount)
   {
      score += amount;
      int size = 10;
   }
   //Obtener la puntuacion
   public int getScore()
   {
      return score;
   }
   //sobre pantalla GAME OVER
   public void gameOver()
   {
      removeObjects(getObjects(greenfoot.Actor.class));
      getBackground().drawImage(new GreenfootImage("Game
Over",40,Color.white,null),getWidth()/2-75,getHeight()/2-60);
      //getBackground().drawImage(new
GreenfootImage("Score:",40,Color.white,null),getWidth()/2-40,getHeight()/2-20);
      Greenfoot.stop();
   }
}
import greenfoot.*;

/**
 * Escribe una descrición de la clase Gameover aquí.
 *
 * @autor (tu nombre)
 * @versión (Un número de versión o una fecha)
 */
public class Gameover extends World
{

   /**
```

```
    * Constructor para objetos de clase Gameover.
    *
    */
   public Gameover()
   {
      // Crea un nuevo mundo de 600x400 celdas con un tamaño de celda de 1x1 pixeles.
      super(600, 400, 1);
   }
}
import greenfoot.*;  // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/**
 * Write a description of class GoGame here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class GoGame extends Buttons
{
   /**
    * Act - do whatever the GoGame wants to do. This method is called whenever
    * the 'Act' or 'Run' button gets pressed in the environment.
    */
   public void act()
   {
    click();
   }

   /**
    *
    */
   private void click()
   {
      if(Greenfoot.mouseClicked(this)){
         getClickSound().play();
         Greenfoot.setWorld(new Game());
      }
   }
}
import greenfoot.*;

/**
 * Write a description of class GoMenu here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class GoMenu extends Buttons
```

```
{
   /**
    * Act - do whatever the GoMenu wants to do. This method is called whenever
    * the 'Act' or 'Run' button gets pressed in the environment.
    */
   public void act()
   {
      click();
   }

   /**
    * click() - Quando clicamos neste botão produz um som e leva-nos para o Menu.
    */
   private void click()
   {
      if(Greenfoot.mouseClicked(this)){
         getClickSound().play();
         Greenfoot.setWorld(new Menu());
      }
   }
}
import greenfoot.*;

/**
 * Escribe una descrición de la clase Help aquí.
 *
 * @autor (tu nombre)
 * @versión (Un número de versión o una fecha)
 */
public class Help extends Actor
{
   /**
    * Act - hace lo que Help quiere hacer. Este método se llama "cuando quiera" o whenever
    * los botones 'Actuar or 'Ejecutar' son presionados en el entorno.
    */
   public void act()
   {
      // Agrega tus códigos de acción aquí.
   }
}
import greenfoot.*;  // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)
import java.awt.Color;
import java.awt.Font;

/**
 * Write a description of class Help here.
 *
 * @author (your name)
```

```
 * @version (a version number or a date)
 */
public class Helpbt extends Buttons
{
   private GreenfootImage imgHelp;
   private Color amarelo;
   private Color laranja;
   private Color laranjaEscuro;
   private Help help;

   /**
    * Construtor, desenha uma caixa, faz print do texto e inicializa o objeto help
    */
   public Helpbt()
   {
      drawBox();
      printText();
      help = new Help();
      imgHelp.setTransparency(0);
   }

   /**
    * Act - do whatever the Help wants to do. This method is called whenever
    * the 'Act' or 'Run' button gets pressed in the environment.
    */
   public void act()
   {
      super.act();
      imgHelp.setTransparency(255);
      click();
   }

   /**
    *
    */
   private void drawBox()
   {
      amarelo = new Color(255,188,0);
      laranja = new Color(255,133,0);
      imgHelp = getImage();
      imgHelp.setColor(laranja);
      imgHelp.fill();
      imgHelp.scale(200,50);
      imgHelp.setColor(amarelo);
      int margem = 5;
      int largura = imgHelp.getWidth()-2*margem;
      int altura = imgHelp.getHeight()-2*margem;
      imgHelp.fillRect(margem,margem,largura,altura);
```

```
    }

    /**
     *
     */
    private void printText()
    {
        laranjaEscuro = new Color(214,95,0);
        imgHelp.setColor(laranjaEscuro);
        Font play = new Font("sanserif",Font.BOLD,30);
        imgHelp.setFont(play);
        int x = imgHelp.getWidth()-130;
        int y = imgHelp.getHeight()-16;
        imgHelp.drawString("Help",x,y);
    }

    /**
     *
     */

    public void click()
    {
        if(Greenfoot.mouseClicked(this)){
            getClickSound().play(); //quando é carregado neste botão é produzido um som de click
            getWorld().addObject(help,getWorld().getWidth()/2,getWorld().getHeight()/2);
//adicionamos num objeto da Classe help, este objeto tem a função de ser um popup que
poderá ser fechado pelo utilizador
            getWorld().addObject(new Exit(help), help.getX() - help.getImage().getWidth()/2,
help.getY() - help.getImage().getHeight()/2); //botão que fecha o popup, adiciona um objeto
da classe Exit que recebe como o objeto o Actor que quer remover, neste caso o objeto
adicionado acima deste
        }
    }
}
import greenfoot.Actor;
import greenfoot.World;
import greenfoot.Greenfoot;
import greenfoot.GreenfootImage;

import java.awt.Color;

public class ImpBullet extends Actor
{
    private int speed = 100;
    private boolean fromShotgun = false;

    public ImpBullet(boolean shot, int speed)
    {
```

```
        GreenfootImage img = new GreenfootImage(4,2);
        img.setColor(Color.yellow);
        img.fill();
        setImage(img);
        fromShotgun = shot;
        this.speed = speed;
    }

    public void act()
    {
        Actor a = moved(speed);
        if (getX() > getWorld().getWidth() || getX() < 0 || getY() < 0 || getY() >
getWorld().getHeight())
            getWorld().removeObject(this);
    }

    public Actor moved(int amount)
    {
        int x1 = getX();
        int y1 = getY();
        move(amount);
        int x2 = getX();
        int y2 = getY();
        setLocation(x1,y1);
        Actor a = null;
        for (int i = 0; i < amount; i +=5)
        {
            move(5);
            a = getOneIntersectingObject(Zombie.class);
            if (a != null)
                break;
        }
        setLocation(x2,y2);
        return a;
    }
}
import greenfoot.Actor;
import greenfoot.World;
import greenfoot.Greenfoot;
import greenfoot.GreenfootImage;
import java.util.List;

import java.awt.Color;

public class Jefe extends Actor
{
    public int speed;
    private int counter = 5;
```

```
   private int thealth = 200;
   private boolean thurt = true;

   public Jefe()
   {
    setImage("Jefe.gif");
   }

   public Jefe(int spd)
   {
      speed = spd;
   }

   public void act()
   {
      Actor a = getOneIntersectingObject(Bullet.class);
      Actor b = getOneIntersectingObject(ImpBullet.class);
      if (a != null)
      {
         thurt(Greenfoot.getRandomNumber(2)+1);
         thurt = true;
         World w = getWorld();
         int size = Greenfoot.getRandomNumber(10)+5;
         for (int i = 0; i < size; i++)
         {
            Color col = Color.red;
            int rot = getRotation()+180;
            double spd = (double)Greenfoot.getRandomNumber(4)-
.1*Greenfoot.getRandomNumber(5);
            w.addObject(new Particle(rot,spd,col),getX(),getY());
         }
         return;
      }
      if (b != null)
      {
         thurt(Greenfoot.getRandomNumber(2)+3);
         thurt = true;
         World w = getWorld();
         int size = Greenfoot.getRandomNumber(10)+5;
         for (int i = 0; i < size; i++)
         {
            Color col = Color.red;
            int rot = getRotation()+180;
            double spd = (double)Greenfoot.getRandomNumber(4)-
.1*Greenfoot.getRandomNumber(5);
            w.addObject(new Particle(rot,spd,col),getX(),getY());
         }
         return;
```

```
    }
    if (thealth <= 20)
        doStuff();
    if (thealth <= 20)
        removeTouching(Bullet.class);
    if (thealth <= 0)
        getWorld().removeObject(this);
    if (Greenfoot.getRandomNumber(1000) > 998)
    {
        Greenfoot.playSound("zombiesound.mp3");
    }
}

public boolean atWorldEdge()
{
    if(getX() < 20 || getX() > getWorld().getWidth() - 20)
        return true;
    if(getY() < 20 || getY() > getWorld().getHeight() - 20)
        return true;
    else
        return false;
}

    public void thurt(int amount)
{
    thealth -= amount;
    int size = Greenfoot.getRandomNumber(5)+1;
}

public void doStuff()
{
    if (atWorldEdge())
    {
        turnTowards(450,300);
    }
    move(1);
    move();
}

private void move()
{
    List p=getObjectsInRange(450,Player.class);
    int Distx, Disty;
    double angle;
    if(p.size()>0)
    {
        Player P=(Player)p.get(0);
        Distx=getX()-P.getX();
```

```
            Disty=getY()-P.getY();
            angle=Math.toDegrees(Math.atan2(Disty,Distx))+180;
            setRotation((int)angle);
            angle = Math.toRadians( getRotation() );
            int x = (int) Math.round(getX() + Math.cos(angle) * speed);
            int y = (int) Math.round(getY() + Math.sin(angle) * speed);
            setLocation(x, y);
        }
    }

    public int getthealth()
    {
        return thealth;
    }
}
import greenfoot.*;

/**
 *
 */
public class Menu extends World
{
    private GreenfootSound sonFondo;
    private Play buttonPlay;
    private Scorebt buttonScore;



    public Menu()
    {
        super(650, 750, 1);

        //inicializacion del escenario
        sonFondo = new GreenfootSound("Music.mp3");
        buttonPlay = new Play();
        buttonScore = new Scorebt();
        preparation();
    }


    public void act()
    {
        loopSom();
        isClicked();

    }
```

```
    private void preparation()
    {
        addObject(buttonPlay, -300, 250);
        addObject(new Helpbt(), -300, 350);
        addObject(buttonScore, -300, 450);
        addObject(new Creditsbt(), -300, 550);
        addObject(new ScoreLoad(),0, 0);
    }


    private void loopSom()
    {
        if(!sonFondo.isPlaying())
            sonFondo.play();
    }


    private void isClicked()
    {

        if(Greenfoot.mouseClicked(buttonPlay) && sonFondo.isPlaying())
            sonFondo.stop();
        if(Greenfoot.mouseClicked(buttonScore) && sonFondo.isPlaying())
            sonFondo.stop();
    }


}
import greenfoot.Actor;
import greenfoot.World;
import greenfoot.Greenfoot;
import greenfoot.GreenfootImage;
import java.util.List;

import java.awt.Color;

public class Necrofago extends Actor
{
    public int speed;
    private int counter = 5;
    private int zhealth = 10;
    private boolean zhurt = true;
    private boolean fromShotgun = false;

    public Necrofago()
    {
     setImage("Necrofago.gif");
    }
```

```
  public Necrofago(int spd)
  {
    speed = spd;
  }

  public void act()
  {
    Actor a = getOneIntersectingObject(Bullet.class);
    Actor b = getOneIntersectingObject(ImpBullet.class);
    if (a != null)
    {
      zhurt(Greenfoot.getRandomNumber(2)+1);
      zhurt = true;
      World w = getWorld();
      int size = Greenfoot.getRandomNumber(10)+5;
      for (int i = 0; i < size; i++)
      {
        Color col = Color.red;
        int rot = getRotation()+180;
        double spd = (double)Greenfoot.getRandomNumber(4)-
.1*Greenfoot.getRandomNumber(5);
        w.addObject(new Particle(rot,spd,col),getX(),getY());
      }
      return;
    }
    if (b != null)
    {
      zhurt(Greenfoot.getRandomNumber(2)+3);
      zhurt = true;
      World w = getWorld();
      int size = Greenfoot.getRandomNumber(10)+5;
      for (int i = 0; i < size; i++)
      {
        Color col = Color.red;
        int rot = getRotation()+180;
        double spd = (double)Greenfoot.getRandomNumber(4)-
.1*Greenfoot.getRandomNumber(5);
        w.addObject(new Particle(rot,spd,col),getX(),getY());
      }
      return;
    }
    if (zhealth <= 10)
      doStuff();
    if (zhealth <= 10)
      removeTouching(Bullet.class);
    if (zhealth <= 0)
      getWorld().removeObject(this);
```

```
    if (Greenfoot.getRandomNumber(1000) > 998)
    {
        Greenfoot.playSound("zombiesound.mp3");
    }
}

public boolean atWorldEdge()
{
    if(getX() < 20 || getX() > getWorld().getWidth() - 20)
        return true;
    if(getY() < 20 || getY() > getWorld().getHeight() - 20)
        return true;
    else
        return false;
}

public void zhurt(int amount)
{
    zhealth -= amount;
    int size = Greenfoot.getRandomNumber(5)+1;
}

public void doStuff()
{
    if (atWorldEdge())
    {
        turnTowards(450,300);
    }
    move(2);
    move();
}

private void move()
{
    List p=getObjectsInRange(450,Player.class);
    int Distx, Disty;
    double angle;
    if(p.size()>0)
    {
        Player P=(Player)p.get(0);
        Distx=getX()-P.getX();
        Disty=getY()-P.getY();
        angle=Math.toDegrees(Math.atan2(Disty,Distx))+180;
        setRotation((int)angle);
        angle = Math.toRadians( getRotation() );
        int x = (int) Math.round(getX() + Math.cos(angle) * speed);
        int y = (int) Math.round(getY() + Math.sin(angle) * speed);
        setLocation(x, y);
```

```
        }
    }

    public int getzhealth()
    {
        return zhealth;
    }
}
import greenfoot.Actor;
import greenfoot.Greenfoot;
import greenfoot.GreenfootImage;

import java.awt.Color;

public class Particle extends Actor
{
    private double speed = 0;

    protected int life = 100;

    protected boolean red = true;

    protected int size = Greenfoot.getRandomNumber(2)+5;

    private int lastX = 0;
    private int lastY = 0;

    public Particle(int rot, double startingSpeed, Color col)
    {
        setRotation(rot);
        speed = startingSpeed;
        makeImage(col);
    }

    private void makeImage(Color col)
    {
        size = Greenfoot.getRandomNumber(2)+5;
        GreenfootImage img = new GreenfootImage(size,size);
        img.setColor(col);
        red = col.equals(Color.red);
        img.fill();
        setImage(img);
    }

    public void act()
    {
        if (lastX == 0 && lastY == 0 && red)
        {
```

```
        lastX = getX();
        lastY = getY();
      }
      Actor a = getOneIntersectingObject(Player.class);
      if (a == null)
        a = getOneIntersectingObject(Zombie.class);
      if (a == null)
        a = getOneIntersectingObject(Jefe.class);
      if (a == null)
        a = getOneIntersectingObject(ZRadiactivo.class);
      if (a != null)
      {
        if ((int)speed == 0)
          speed = 2;
        else
          speed *= 2;
        setRotation(a.getRotation());
      }
      move((int)speed);
      speed -= speed/10.0;
      if (a == null && (lastX != getX() || lastY != getY()) && red)
      {
        lastX = getX();
        lastY = getY();
      }
      else
        red = true;
      if (life <= 0)
        fade();
      else
        dist();
      life--;
      try
      {
        if (getWorld() != null && getX() > getWorld().getWidth() || getY() >
getWorld().getWidth() || getY() < 0 || getX() < 0)
          getWorld().removeObject(this);
          life -= 10;
      }
      catch (IllegalStateException e){}
    }

    private void dist()
    {
      Actor p = (Actor)getWorld().getObjects(Player.class).get(0);
      if (p == null)
      {
        getImage().setTransparency(0);
```

```
        }
        int dist = (int)Distance.distanceBetween(this,p);
        if (dist > 255)
            dist = 255;
        getImage().setTransparency(255-dist);
        if (dist >= 200)
            life--;
    }

    private void fade()
    {
        if (life <= 0 & getImage().getTransparency()-4 >= 0)
            getImage().setTransparency(getImage().getTransparency()-4);
        else if (life <= 0)
            getWorld().removeObject(this);
    }
}
import greenfoot.*;
import java.awt.Color;
import java.awt.Font;

/**
 * Write a description of class Play here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Play extends Buttons
{
    private GreenfootImage imgPlay;
    private Color amarelo;
    private Color laranja;
    private Color laranjaEscuro;

    /**
     * Construtor da classe Play
     */
    public Play()
    {
        drawBox();
        printText();
        imgPlay.setTransparency(0);
    }

    public void act()
    {
        super.act();
        imgPlay.setTransparency(255);
```

```
        click();
    }

    private void drawBox()
    {
        amarelo = new Color(255,188,0);
        laranja = new Color(255,133,0);
        imgPlay = getImage();
        imgPlay.setColor(laranja);
        imgPlay.fill();
        imgPlay.scale(200,50);
        imgPlay.setColor(amarelo);
        int margem = 5;
        int largura = imgPlay.getWidth()-2*margem;
        int altura = imgPlay.getHeight()-2*margem;
        imgPlay.fillRect(margem,margem,largura,altura);
    }

    private void printText()
    {
        laranjaEscuro = new Color(214,95,0);
        imgPlay.setColor(laranjaEscuro);
        Font play = new Font("sanserif",Font.BOLD,30);
        imgPlay.setFont(play);
        int x = imgPlay.getWidth()-130;
        int y = imgPlay.getHeight()-16;
        imgPlay.drawString("Play",x,y);
    }

    /**
     *
     */
    private void click()
    {
        if(Greenfoot.mouseClicked(this))
        {
         getClickSound().play();
         Greenfoot.setWorld(new Game());
        }
    }
}
import greenfoot.Actor;
import greenfoot.Greenfoot;
import greenfoot.GreenfootImage;
import java.awt.Color;
import java.awt.Font;
import java.util.List;
import java.util.ArrayList;
```

```
public class Player extends Actor
{
   GreenfootImage scoreBoard = new GreenfootImage(150,100);
   private int speed = 3;
   private int counter = 5;
   private int health = 100;
   private boolean hurt = true;
   private int radAway;


   private static final GreenfootImage norm = new GreenfootImage("PLAYER.gif");
   private static final GreenfootImage high = new GreenfootImage("PLAYER3.gif");
   private static final GreenfootImage med = new GreenfootImage("PLAYER5.gif");
   private static final GreenfootImage low = new GreenfootImage("PLAYER4.gif");;

   public Player()
   {
    setImage(norm);
    radAway=0;
   }

   public void addedToWorld(greenfoot.World wrld)
   {
    move();
   }
   public void act()
   {
      move();
      shoot();

      if (Greenfoot.getMouseInfo() == null)
         return;
      int x = Greenfoot.getMouseInfo().getX();
      int y = Greenfoot.getMouseInfo().getY();
      turnTowards(x,y);

      Actor a = getOneIntersectingObject(Necrofago.class);
      Actor b = getOneIntersectingObject(Zombie.class);
      Actor c = getOneIntersectingObject(ZRadiactivo.class);
      Actor d = getOneIntersectingObject(Jefe.class);
      Actor e = getOneIntersectingObject(Superviviente.class);

      if (a != null)
      {
         hurt(2);
         a.move(-5);
         hurt = true;
```

```
    }

    if (b != null)
    {
        hurt(4);
        b.move(-10);
        hurt = true;
    }

    if (c != null)
    {
        hurt(1);
        c.move(-20);
        hurt = true;
    }

     if (d != null)
    {
        hurt(1);
        c.move(-50);
        hurt = true;
    }
     if (e != null)
    {
     health += 50;
    }

    if (health <= 95)
        setImage(high);
    if (health <= 65)
        setImage(med);
    if (health <= 20)
        setImage(low);
    if (health <= 0)
        ((Game)getWorld()).gameOver();

   if(foundRad())
     eatRad();

     //else
       //  move();

}

public void move()
{
    if (Greenfoot.isKeyDown("w"))
        move(speed);
```

```
    if (Greenfoot.isKeyDown("s"))
        move(-speed);
}

public void move(int dist)
{
    int x = getX();
    int y = getY();
    super.move(dist);
    if (getX() > getWorld().getWidth() || getX() < 0)
        setLocation(x, getY());
    if (getY() > getWorld().getHeight() || getY() < 0)
        setLocation(getX(), y);
}

private void shoot()
{
    if (counter < 40)
        counter++;
    if (counter < 10)
        return;
    if (Greenfoot.isKeyDown("space"))
    {
        shootAssault();
        Greenfoot.playSound("bullet.wav");
    }
    if (counter < 40)
        return;
    if (Greenfoot.isKeyDown("shift"))
    {
        shootShotgun();
        Greenfoot.playSound("shotBullet.wav");
    }
}

public void shootAssault()
{
    counter = 0;
    Bullet b = new Bullet(false,20);
    getWorld().addObject(b,getX(),getY());
    b.setRotation(getRotation());
    b.move(10);
    b.turn(90);
    b.move(2);
    b.turn(-90);
    b.turn(Greenfoot.getRandomNumber(8)-4);
}
```

```
public void shootShotgun()
{
   counter = 4;
   Bullet[] bees = new Bullet[8];
   for (int i = 0; i < bees.length; i++)
      bees[i] = new Bullet(true,30);
   for (Bullet b : bees)
   {
      getWorld().addObject(b,getX(),getY());
      b.setRotation(getRotation());
      b.move(10);
      b.turn(90);
      b.move(2);
      b.turn(-90);
      b.turn(Greenfoot.getRandomNumber(12)-6);
   }
}

public void hurt(int amount)
{
   health -= amount;
   int size = Greenfoot.getRandomNumber(5)+1;
   for (int i = 0; i < size; i++)
   {
      int rot = Greenfoot.getRandomNumber(360);
      int spd = 2;
   }
}

public int getHealth()
{
 return health;
}

public void eatRad()
{
   Actor RadAway = getOneObjectAtOffset(0, 0, RadAway.class);
   if(RadAway != null) {
      // eat the leaf...
      getWorld().removeObject(RadAway);
      radAway = health + 10;
   }
}

public void foundRadAway()
{
   Actor RadAway = getOneObjectAtOffset(0, 0, RadAway.class);
   if(RadAway != null)
```

```
        {
         // eat the leaf...
         getWorld().removeObject(RadAway);
         radAway = health + 1;
        }
    }

    public boolean foundRad()
    {
       Actor RadAway = getOneObjectAtOffset(0,0,RadAway.class);
       if(RadAway != null)
        {
         return true;
        }
        else
           {
            return false;
           }
    }

     public int getRadEaten()
    {
     return radAway;
    }

}
import greenfoot.*;

/**
 * Escribe una descrición de la clase RADAWAY aquí.
 *
 * @autor (tu nombre)
 * @versión (Un número de versión o una fecha)
 */
public class RadAway extends Actor
{
    /**
     * Act - hace lo que RADAWAY quiere hacer. Este método se llama "cuando quiera" o
whenever
     * los botones 'Actuar or 'Ejecutar' son presionados en el entorno.
     */
    public void act()
    {
       // Agrega tus códigos de acción aquí.
    }
}
import greenfoot.*;
```

```
/**
 * Escribe una descrición de la clase Radio aquí.
 *
 * @autor (tu nombre)
 * @versión (Un número de versión o una fecha)
 */
public class Radio extends Actor
{
   /**
    * Act - hace lo que Radio quiere hacer. Este método se llama "cuando quiera" o whenever
    * los botones 'Actuar or 'Ejecutar' son presionados en el entorno.
    */
   public void act()
   {
      // Agrega tus códigos de acción aquí.
   }
}
import greenfoot.*;

/**
 * Escribe una descrición de la clase Rads aquí.
 *
 * @autor (tu nombre)
 * @versión (Un número de versión o una fecha)
 */
public class Rads extends Actor
{
   /**
    * Act - hace lo que Rads quiere hacer. Este método se llama "cuando quiera" o whenever
    * los botones 'Actuar or 'Ejecutar' son presionados en el entorno.
    */
   public void act()
   {
      // Agrega tus códigos de acción aquí.
   }
}
import greenfoot.*;  // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)
import java.awt.Color;
import java.awt.Font;
import java.util.*;
/**
 * Write a description of class Score here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Score extends Actor
{
```

```
    public void act()
    {

    }



}
import greenfoot.*;

/**
 *
 */
public class Scoreboard extends World
{
    /**
     * Constructor for objects of class Scoreboard.
     *
     */
    public Scoreboard()
    {
        super(650, 750, 1);

        adicionaObjetos();
    }

    /**
     * adicionaObjetos- adiciona o titulo da tabela de pontuações e as respetivas pontuações a
este cenário
     */
    private void adicionaObjetos()
    {
        addObject(new DesignScoreboard(),320,250);
        addObject(new FirstPlace(),100,350);
        addObject(new SecondPlace(),100,450);
        addObject(new ThirdPlace(),100,550);
        addObject(new FourthPlace(),100,650);
        addObject(new GoMenu(),620,720);
    }
}
import greenfoot.*;  // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)
import java.awt.Color;
import java.awt.Font;

/**
 * Write a description of class Score here.
```

```
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Scorebt extends Buttons
{
   private GreenfootImage imgScore;
   private Color amarelo;
   private Color laranja;
   private Color laranjaEscuro;

   /**
    * Construtor da classe Scorebt
    */
   public Scorebt()
   {
      drawBox();
      printText();
      imgScore.setTransparency(0);
   }

   /**
    * Act - do whatever the Score wants to do. This method is called whenever
    * the 'Act' or 'Run' button gets pressed in the environment.
    */
   public void act()
   {
      super.act();
      imgScore.setTransparency(255);
      click();
   }

   /**
    * printText - Escreve texto na caixa.
    */
   private void printText()
   {
      laranjaEscuro = new Color(214,95,0);
      imgScore.setColor(laranjaEscuro);
      Font play = new Font("sanserif",Font.BOLD,30);
      imgScore.setFont(play);
      int x = imgScore.getWidth()-140;
      int y = imgScore.getHeight()-16;
      imgScore.drawString("Score",x,y);
   }

   /**
```

```
    * click - Método que adiciona um evento quando carregamos no botão, neste caso dá um
som e muda para o cenário que apresenta os melhores scores.
    */
   private void click()
   {
      if(Greenfoot.mouseClicked(this)){
         getClickSound().play();
         Greenfoot.setWorld(new Scoreboard());
      }
   }

    /**
    * drawBox - Desenha a caixa e suas margens para o botão.
    */
   private void drawBox()
   {
      amarelo = new Color(255,188,0);
      laranja = new Color(255,133,0);
      imgScore = getImage();
      imgScore.setColor(laranja);
      imgScore.fill();
      imgScore.scale(200,50);
      imgScore.setColor(amarelo);
      int margem = 5;
      int largura = imgScore.getWidth()-2*margem;
      int altura = imgScore.getHeight()-2*margem;
      imgScore.fillRect(margem,margem,largura,altura);
   }
}
import greenfoot.*;  // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)
import java.io.*;
import java.util.*;

/**
 * Write a description of class ScoreLoad here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class ScoreLoad extends Actor
{
   private ArrayList<String> listaMelhores = new ArrayList<String>(); //lista que guardara as
pontuações por ordem decrescente de pontuação
   /**
    * Act - do whatever the ScoreLoad wants to do. This method is called whenever
    * the 'Act' or 'Run' button gets pressed in the environment.
    */
   public void act()
```

```
  {
    // Add your action code here.
  }

  /**
   *
   */
  public ScoreLoad()
  {
    GreenfootImage imagem = getImage();
    imagem.clear();
    try
    {
      BufferedReader reader = new BufferedReader(new FileReader("records.txt")); //criação do objeto reader que permitirá aceder aos dados presentes no ficheiro "records.txt"

      String linha = null; //onde vamos guardar o que é retirado do ficheiro onde estão guardados os recordes
      while((linha=reader.readLine())!=null)  //enquanto for diferente de false quer dizer que ha ainda dados para ler e guarda-o na variavel string e também já processa o que foi recebido
        processaDados(linha);
    }catch(IOException ex)
    {
      System.out.println("No records yet!"); //se nao existir o ficheiro esta exception é "catched"
    }
  }

  /**
   * processaDados(String)- processa os dados obtidos pela leitura do ficheiro "records.txt". Nomeadamente adiciona à listaMelhores a pontuação,
   * mas nao adiciona de qualquer maneira, adiciona de forma decrescente, ou seja a maior pontuação fica na posição 0 da lista.
   */
  private void processaDados(String recordes)
  {
    if(listaMelhores.isEmpty()) //se a lista estiver vazia adicionamos simplesmente na lista sem ser preciso fazer qualquer comparação
      listaMelhores.add(recordes);
    else
    {
      for(int i = 0, n = listaMelhores.size(); i < n ; i++) //vamos incrementando i até encontrarmos uma posicao em que o valor do array seja maior ou igual ao da lista ou até chegarmos à ulima posicao da lista onde fazemos a comparação final
      {
        if(Integer.parseInt(listaMelhores.get(i))<=Integer.parseInt(recordes)) //se o que está na lista for menor que proximo elemento "recordes", entao guardamos nessa posicao da lista
        {
```

```
                listaMelhores.add(i,recordes);
                break;
            }
            else if(i==n-1) //se ja tivermos no ultimo caso e o que estiver na lista for maior que o elemento "recordes", entao metemos na ultima posicao
                listaMelhores.add(recordes);
        }
    }
  }

  /**
   * getList() - retorna a lista que contém as pontuações ordenadas por ordem decrescente
   */
  protected List getList()
  {
    return listaMelhores;
  }
}
import greenfoot.*;  // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)
import java.awt.Color;
import java.awt.Font;
/**
 * Write a description of class FirstPlace here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class SecondPlace extends ScoreLoad
{
  GreenfootImage segundoLugar = new GreenfootImage(400,50);
  private Color amarelo;

  /**
   * Act - do whatever the DesignScoreboard wants to do. This method is called whenever
   * the 'Act' or 'Run' button gets pressed in the environment.
   */
  public void act()
  {
    // Add your action code here.
  }

  /**
   * Construtor da clase SecondPlace, simplesmente faz clear na imagem predefinida e define uma nova imagem
   */
  public SecondPlace()
  {
    display();
```

```
    }

    /**
     * display() - faz display da segunda melhor pontuação alguma vez obtida
     */
    private void display()
    {
        amarelo = new Color(255,188,0);
        segundoLugar.setColor(amarelo);
        segundoLugar.setFont(new Font("sanserif",Font.BOLD,25));
        String score;
        if(getList().size()<2)//se o tamanho da lista for menor que 2 entao significa que ainda o
utilizador ainda nao jogou quatro vezes portanto nao existe quarta melhor pontuação
            score = "0";
        else
            score = (String)getList().get(1); //caso contrario, vai buscar o 2 elemento da lista dando
obviamente a segunda melhor pontuacao de sempre porque a lista esta por ordem descrente

        segundoLugar.drawString("Second best: " + score,150,segundoLugar.getHeight()/2);

        setImage(segundoLugar);
    }
}
import greenfoot.Actor;
import greenfoot.World;
import greenfoot.Greenfoot;
import greenfoot.GreenfootImage;
import java.util.List;

import java.awt.Color;

/**
 * Escribe una descrición de la clase Superviviente aquí.
 *
 * @autor (tu nombre)
 * @versión (Un número de versión o una fecha)
 */
public class Superviviente extends Actor
{
    /**
     * Act - hace lo que Superviviente quiere hacer. Este método se llama "cuando quiera" o
whenever
     * los botones 'Actuar or 'Ejecutar' son presionados en el entorno.
     */
    public int speed;
    private int counter = 5;
    private boolean zhurt = false;
```

```
    public Superviviente()
    {
     setImage("Superviviente.gif");
    }
    public Superviviente(int spd)
    {
     speed = spd;
    }
    public void act()
    {
        // Agrega tus códigos de acción aquí.
        move();
    }
        private void move()
    {
        List p=getObjectsInRange(450,Player.class);
        int Distx, Disty;
        double angle;
        if(p.size()>0)
        {
            Player P=(Player)p.get(0);
            Distx=getX()-P.getX();
            Disty=getY()-P.getY();
            angle=Math.toDegrees(Math.atan2(Disty,Distx))+180;
            setRotation((int)angle);
            angle = Math.toRadians( getRotation() );
            int x = (int) Math.round(getX() + Math.cos(angle) * speed);
            int y = (int) Math.round(getY() + Math.sin(angle) * speed);
            setLocation(x, y);
        }
    }
}
import greenfoot.*;  // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)
import java.awt.Color;
import java.awt.Font;
/**
 * Write a description of class FirstPlace here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class ThirdPlace extends ScoreLoad
{
    GreenfootImage terceiroLugar = new GreenfootImage(400,50);
    private Color laranja;

    /**
     * Act - do whatever the DesignScoreboard wants to do. This method is called whenever
```

```
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public void act()
    {
        // Add your action code here.
    }

    /**
     * Construtor da clase ThirdPlace, simplesmente faz clear na imagem predefinida e define
uma nova imagem
     */
    public ThirdPlace()
    {
        display();
    }

     /**
     * display() - faz display da terceira melhor pontuação alguma vez obtida
     */
    private void display()
    {
        laranja = new Color(255,133,0);
        terceiroLugar.setColor(laranja);
        terceiroLugar.setFont(new Font("sanserif",Font.BOLD,25));
        String score;
        if(getList().size()<3)//se o tamanho da lista for menor que 3 entao significa que ainda o
utilizador ainda nao jogou quatro vezes portanto nao existe quarta melhor pontuação
            score = "0";
        else
            score = (String)getList().get(2); //caso contrario, vai buscar o 3 elemento da lista dando
obviamente a terceira melhor pontuacao de sempre porque a lista esta por ordem descrente

        terceiroLugar.drawString("Third best: " + score,150,terceiroLugar.getHeight()/2);

        setImage(terceiroLugar);
    }
}
import greenfoot.*;

/**
 * Escribe una descrición de la clase Torreta aquí.
 *
 * @autor (tu nombre)
 * @versión (Un número de versión o una fecha)
 */
public class Torreta extends Actor
{
    /**
```

```
    * Act - hace lo que Torreta quiere hacer. Este método se llama "cuando quiera" o whenever
    * los botones 'Actuar or 'Ejecutar' son presionados en el entorno.
    */
   public void act()
   {
      // Agrega tus códigos de acción aquí.
   }
}
import greenfoot.Greenfoot;

public final class Vector
{
   double dx;
   double dy;
   int direction;
   double length;

   /**
    * Create a new, neutral vector.
    */
   public Vector()
   {
   }

   /**
    * Create a vector with given direction and length. The direction should be in
    * the range [0..359], where 0 is EAST, and degrees increase clockwise.
    */
   public Vector(int direction, double length)
   {
      this.length = length;
      this.direction = direction;
      updateCartesian();
   }

   /**
    * Create a vector by specifying the x and y offsets from start to end points.
    */
   public Vector(double dx, double dy)
   {
      this.dx = dx;
      this.dy = dy;
      updatePolar();
   }

   /**
    * Set the direction of this vector, leaving the length intact.
    */
```

```
public void setDirection(int direction)
{
   this.direction = direction;
   updateCartesian();
}

/**
 * Add another vector to this vector.
 */
public void add(Vector other)
{
   dx += other.dx;
   dy += other.dy;
   updatePolar();
}

/**
 * Reduce the length of this vector, leaving the direction intact.
 */
public void reduceLength(double d)
{
   length = length - d;
   updateCartesian();
}

/**
 * Scale this vector up (factor > 1) or down (factor < 1). The direction
 * remains unchanged.
 */
public void scale(double factor)
{
   length = length * factor;
   updateCartesian();
}

/**
 * Set this vector to the neutral vector (length 0).
 */
public void setNeutral() {
   dx = 0.0;
   dy = 0.0;
   length = 0.0;
   direction = 0;
}

/**
 * Revert to horizontal component of this movement vector.
 */
```

```
public void revertHorizontal() {
   dx = -dx;
   updatePolar();
}

/**
 * Revert to vertical component of this movement vector.
 */
public void revertVertical() {
   dy = -dy;
   updatePolar();
}

/**
 * Return the x offset of this vector (start to end point).
 */
public double getX() {
   return dx;
}

/**
 * Return the y offset of this vector (start to end point).
 */
public double getY() {
   return  dy;
}

/**
 * Return the direction of this vector (in degrees). 0 is EAST.
 */
public int getDirection() {
   return direction;
}

/**
 * Return the length of this vector.
 */
public double getLength() {
   return length;
}

/**
 * Update the direction and length fom the current dx, dy.
 */
private void updatePolar()
{
   this.direction = (int) Math.toDegrees(Math.atan2(dy, dx));
   this.length = Math.sqrt(dx*dx+dy*dy);
```

```
    }

    /**
     * Update dx and dy from the current direction and length.
     */
    private void updateCartesian()
    {
        dx = length * Math.cos(Math.toRadians(direction));
        dy = length * Math.sin(Math.toRadians(direction));
    }
}
import greenfoot.Actor;
import greenfoot.World;
import greenfoot.Greenfoot;
import greenfoot.GreenfootImage;
import java.util.List;

import java.awt.Color;

public class Zombie extends Actor
{
    public int speed;

    private int counter = 5;

    private int zhealth = 10;

    private boolean zhurt = true;

    private boolean fromShotgun = false;

    public Zombie()
    {
     setImage("Zombie.gif");
    }

    public Zombie(int spd)
    {
        speed = spd;
    }

    public void act()
    {
        Actor a = getOneIntersectingObject(Bullet.class);
        Actor b = getOneIntersectingObject(ImpBullet.class);//Impacto de bala
        if (a != null)
        {
            zhurt(Greenfoot.getRandomNumber(2)+1);
```

```
            zhurt = true;
            World w = getWorld();
            int size = Greenfoot.getRandomNumber(10)+5;
            for (int i = 0; i < size; i++)
            {
               Color col = Color.red;
               int rot = getRotation()+180;
               double spd = (double)Greenfoot.getRandomNumber(4)-
.1*Greenfoot.getRandomNumber(5);
               w.addObject(new Particle(rot,spd,col),getX(),getY());
            }
            return;
        }
        if (b != null)
        {
            zhurt(Greenfoot.getRandomNumber(2)+3);
            zhurt = true;
            World w = getWorld();
            int size = Greenfoot.getRandomNumber(10)+5;
            for (int i = 0; i < size; i++)
            {
               Color col = Color.red;
               int rot = getRotation()+180;
               double spd = (double)Greenfoot.getRandomNumber(4)-
.1*Greenfoot.getRandomNumber(5);
               w.addObject(new Particle(rot,spd,col),getX(),getY());
            }
            return;
        }
        if (zhealth <= 10)
            doStuff();
        if (zhealth <= 10)
            removeTouching(Bullet.class);
        if (zhealth <= 0)
            getWorld().removeObject(this);
        if (Greenfoot.getRandomNumber(1000) > 998)
        {
            Greenfoot.playSound("zombiesound.mp3");
        }
    }

    public boolean atWorldEdge()
    {
        if(getX() < 20 || getX() > getWorld().getWidth() - 20)
            return true;
        if(getY() < 20 || getY() > getWorld().getHeight() - 20)
            return true;
        else
```

```
        return false;
    }

    public void zhurt(int amount)
    {
        zhealth -= amount;
        int size = Greenfoot.getRandomNumber(5)+1;
    }

    public void doStuff()
    {
        if (atWorldEdge())
        {
            turnTowards(450,300);
        }
        move(2);
        move();
    }

    private void move()
    {
        List p=getObjectsInRange(450,Player.class);
        int Distx, Disty;
        double angle;
        if(p.size()>0)
        {
            Player P=(Player)p.get(0);
            Distx=getX()-P.getX();
            Disty=getY()-P.getY();
            angle=Math.toDegrees(Math.atan2(Disty,Distx))+180;
            setRotation((int)angle);
            angle = Math.toRadians( getRotation() );
            int x = (int) Math.round(getX() + Math.cos(angle) * speed);
            int y = (int) Math.round(getY() + Math.sin(angle) * speed);
            setLocation(x, y);
        }
    }

    public int getzhealth()
    {
        return zhealth;
    }
}
import greenfoot.Actor;
import greenfoot.World;
import greenfoot.Greenfoot;
import greenfoot.GreenfootImage;
import java.util.List;
```

```
import java.awt.Color;

public class ZRadiactivo extends Actor
{
   public int speed;

   private int counter = 5;

   private int zhealth = 10;

   private boolean zhurt = true;

   private boolean fromShotgun = false;

   public ZRadiactivo()
   {
    setImage("Radiactivo.gif");
   }

   public ZRadiactivo(int spd)
   {
    speed = spd;
   }

   public void act()
   {
      Actor a = getOneIntersectingObject(Bullet.class);
      Actor b = getOneIntersectingObject(ImpBullet.class);//Impacto de bala
      if (a != null)
      {
         zhurt(Greenfoot.getRandomNumber(2)+1);
         zhurt = true;
         World w = getWorld();
         int size = Greenfoot.getRandomNumber(10)+5;
         for (int i = 0; i < size; i++)
         {
            Color col = Color.red;
            int rot = getRotation()+180;
            double spd = (double)Greenfoot.getRandomNumber(4)-
.1*Greenfoot.getRandomNumber(5);
            w.addObject(new Particle(rot,spd,col),getX(),getY());
         }
         return;
      }
      if (b != null)
      {
         zhurt(Greenfoot.getRandomNumber(2)+3);
```

```
            zhurt = true;
            World w = getWorld();
            int size = Greenfoot.getRandomNumber(10)+5;
            for (int i = 0; i < size; i++)
            {
               Color col = Color.red;
               int rot = getRotation()+180;
               double spd = (double)Greenfoot.getRandomNumber(4)-
.1*Greenfoot.getRandomNumber(5);
               w.addObject(new Particle(rot,spd,col),getX(),getY());
            }
            return;
        }
        if (zhealth <= 10)
            doStuff();
        if (zhealth <= 10)
            removeTouching(Bullet.class);
        if (zhealth <= 0)
            getWorld().removeObject(this);
        if (Greenfoot.getRandomNumber(1000) > 998)
        {
            Greenfoot.playSound("zombiesound.mp3");
        }
    }

    public boolean atWorldEdge()
    {
        if(getX() < 20 || getX() > getWorld().getWidth() - 20)
            return true;
        if(getY() < 20 || getY() > getWorld().getHeight() - 20)
            return true;
        else
            return false;
    }

    public void zhurt(int amount)
    {
        zhealth -= amount;
        int size = Greenfoot.getRandomNumber(5)+1;
    }

    public void doStuff()
    {
        if (atWorldEdge())
        {
            turnTowards(450,300);
        }
        move(2);
```

```
        move();
    }

    private void move()
    {
        List p=getObjectsInRange(450,Player.class);
        int Distx, Disty;
        double angle;
        if(p.size()>0)
        {
            Player P=(Player)p.get(0);
            Distx=getX()-P.getX();
            Disty=getY()-P.getY();
            angle=Math.toDegrees(Math.atan2(Disty,Distx))+180;
            setRotation((int)angle);
            angle = Math.toRadians( getRotation() );
            int x = (int) Math.round(getX() + Math.cos(angle) * speed);
            int y = (int) Math.round(getY() + Math.sin(angle) * speed);
            setLocation(x, y);
        }
    }

    public int getzhealth()
    {
        return zhealth;
    }
}
```