Universidad Autónoma de San Luis Potosí

314 – Gravity Race

Ulises Yamil Castorena Caldera
1.0

Índice de clases

Jerarquía de la clase

Esta lista de herencias esta ordenada aproximadamente por orden alfabético:

Índice de clases

Lista de clases

Lista de las clases, estructuras, uniones e interfaces con una breve descripción:

Indice de archivos

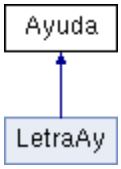
Lista de archivos

Lista de todos los archivos con descripciones breves:

Documentación de las clases

Referencia de la Clase Ayuda

Diagrama de herencias de Ayuda



Métodos públicos

- Ayuda ()
- void act ()

Atributos protegidos

- GreenfootImage img

Descripción detallada

Clase en la cual se crea la ayuda y se muestra.

Autor:

Ulises Yamil Castorena Caldera

Versión:

1.0

Definición en la línea 9 del archivo Ayuda.java.

Documentación del constructor y destructor

Ayuda.Ayuda ()

Constructor para la clase ayuda, unicamnete se crea el objeto y se dibuja la imagen.

Definición en la línea 16 del archivo Ayuda.java.

```
    {
        img = new GreenfootImage("ayuda.png");
        setImage(img);
    }
```

Documentación de las funciones miembro

void Ayuda.act ()

Método Act de la clase que muestra la imagen.

Reimplementado en LetraAy.

Definición en la línea 25 del archivo Ayuda.java.

```
    {
        setImage(img);
```

```
    }
```

Documentación de los datos miembro

GreenfootImage Ayuda.img `[protected]`

Definición en la línea 11 del archivo Ayuda.java.

La documentación para esta clase fue generada a partir del siguiente fichero:

- Ayuda.java

Referencia de la Clase Barra

Métodos públicos

- Barra ()
- void act ()

Descripción detallada

En esta clase se crean los objetos Barra que son los limitantes del vehiculo del personaje.

Autor:

Ulises Yamil Castorena Caldera

Versión:

1.0

Definición en la línea 9 del archivo Barra.java.

Documentación del constructor y destructor

Barra.Barra ()

Constructor de la clase Barra, se crea el objeto para despues mostrar la imagen.

Definición en la línea 16 del archivo Barra.java.

```
    {
        img = new GreenfootImage("Wall.jpg");
        img.scale(800,40);
        setImage(img);
    }
```

Documentación de las funciones miembro

void Barra.act ()

Método act de la clase, unicamente muestra la imagen.

Definición en la línea 26 del archivo Barra.java.

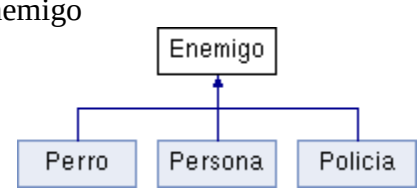
```
    {
        setImage(img);
    }
```

La documentación para esta clase fue generada a partir del siguiente fichero:

- Barra.java

Referencia de la Clase Enemigo

Diagrama de herencias de Enemigo



Métodos públicos

- void Enemigo ()
- void mueve ()

Atributos protegidos

- int ubic
- GreenfootImage img
- int band

Descripción detallada

Super clase de los enemigos en la cual se colocan las variables para que sus sub - clases las hereden.

Autor:

Ulises Yamil Castorena Caldera

Versión:

1.0

Definición en la línea 9 del archivo Enemigo.java.

Documentación del constructor y destructor

void Enemigo.Enemigo ()

Constructor de la clase Enemigo, uniccamente se iguala band a cero.

Definición en la línea 18 del archivo Enemigo.java.

```
{
    band = 0;
}
```

Documentación de las funciones miembro

void Enemigo.mueve ()

Método que mueve la imagen cuatro espacios hacia la izquierda.

Definición en la línea 26 del archivo Enemigo.java.

```
{
    move(-4);
}
```

Documentación de los datos miembro

int Enemigo.band [protected]

Definición en la línea 13 del archivo Enemigo.java.

GreenfootImage Enemigo.img [protected]

Definición en la línea 12 del archivo Enemigo.java.

int Enemigo.ubic [protected]

Definición en la línea 11 del archivo Enemigo.java.

La documentación para esta clase fue generada a partir del siguiente fichero:

- Enemigo.java

Referencia de la Clase Letra

Métodos públicos

- Letra (String valor)
- Letra (int valor)
- void act ()

Descripción detallada

La clase Letra son todos los objetos con texto que aparecen al momento que se estan jugando.

Autor:

Ulises Yamil Castorena Caldera

Versión:

1.0

Definición en la línea 10 del archivo Letra.java.

Documentación del constructor y destructor

Letra.Letra (String *valor*)

Constructor para la clase cadena, unicamente se crea el objeto y lo muestra.

Parámetros:

valor	Es la cadena que se mostrara.

Definición en la línea 18 del archivo Letra.java.

```
    {
        img = new GreenfootImage(valor,20,Color.BLACK,Color.WHITE);
        setImage(img);
    }
```

Letra.Letra (int *valor*)

Constructor para la clase cadena, pero a diferencia del otro constructor en este, se convierte un entero a cadena y lo muestra en el objeto.

Parámetros:

valor	Es el entero que se va a convertir en cadena.

Definición en la línea 28 del archivo Letra.java.

```
    {
        img = new GreenfootImage("NIVEL " +
Integer.toString(valor),30,Color.WHITE,Color.BLACK);
        setImage(img);
    }
```

Documentación de las funciones miembro

void Letra.act ()

Método Act de la clase Letra que solamente muetra el objeto.

Definición en la línea 37 del archivo Letra.java.

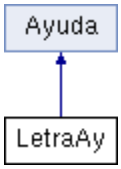
```
    {
        setImage(img);
    }
```

La documentación para esta clase fue generada a partir del siguiente fichero:

- Letra.java

Referencia de la Clase LetraAy

Diagrama de herencias de LetraAy



Métodos públicos

- LetraAy (int valor)
- void act ()
- void verificaMouse ()

Otros miembros heredados

Descripción detallada

Clase donde se crean los objetos relacionados con la ayuda, el boton de acceso y el de regreso.

Autor:

Ulises Yamil Castorena Caldera

Versión:

1.0

Definición en la línea 9 del archivo LetraAy.java.

Documentación del constructor y destructor

LetraAy.LetraAy (int *valor*)

Constructor de la clase, crea un objeto dependiendo del valor que recibe.

Parámetros:

valor	Segun sea su valor va a hacer el objeto a crear si es cero crea un objeto para acceder a la ayuda, sino crea un objeto para regresar de la ayuda.

Definición en la línea 17 del archivo LetraAy.java.

```
    {
        tipo = valor;
        if(tipo == 0)
           img = new GreenfootImage("ay.png");
        else
        {
           img = new GreenfootImage("return.png");
           img.scale(50,50);
        }
        setImage(img);
    }
```

Documentación de las funciones miembro

void LetraAy.act ()

Método Act el cual muestra la imagen y llama al método que verifica si se dio clic en el objeto.

Reimplementado de Ayuda.

Definición en la línea 33 del archivo LetraAy.java.

```
{
    setImage(img);
    verificaMouse();
}
```

void LetraAy.verificaMouse ()

Método que verifica en que clase de objeto se dio clic y dependiendo del objeto es su comportamiento.

Definición en la línea 42 del archivo LetraAy.java.

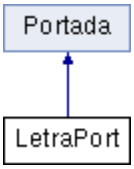
```
{
    if(Greenfoot.mouseClicked(this))
    {
        if(tipo == 0)
        {
           getWorld().removeObjects(getWorld().getObjects(Portada.class));
           getWorld().addObject(new Ayuda(),400,200);
           getWorld().addObject(new LetraAy(1),640,320);
           getWorld().removeObject(this);
        }
        else
        {
           ((Mundo)getWorld()).menu();
           getWorld().removeObject(this);
        }
    }
}
```

La documentación para esta clase fue generada a partir del siguiente fichero:

- LetraAy.java

Referencia de la Clase LetraPort

Diagrama de herencias de LetraPort



Métodos públicos

- LetraPort ()
- void act ()
- void verificaMouse ()

Otros miembros heredados

Descripción detallada

Clase que crea los objetos para poder acceder al juego desde la portada.

Autor:

Ulises Yamil Castorena Caldera

Versión:

1.0

Definición en la línea 9 del archivo LetraPort.java.

Documentación del constructor y destructor

LetraPort.LetraPort ()

Constructor de la clase, que inicializa las variables, aparte de crear el objeto imagen y el objeto sonido.

Definición en la línea 18 del archivo LetraPort.java.

```
    {
         band = 0;
         play = new GreenfootImage("play.png");
         son = new GreenfootSound("Ferrari.mp3");
         setImage(play);
    }
```

Documentación de las funciones miembro

void LetraPort.act ()

Método Act que muestra la imagen y llama a los Métodos que verifican si se dio clic en el objeto, si es asi, se manda un mensaje al mundo que indica que debe de comenzar el juego.

Reimplementado de Portada.

Definición en la línea 30 del archivo LetraPort.java.

```
    {
```

```
        setImage(play);
        verificaMouse();
        if(!son.isPlaying() && band != 0)
           ((Mundo)getWorld()).comiezaJuego();
    }
```

void LetraPort.verificaMouse ()

Verifica si se dio clic en este objeto y reproduce el objeto sonido y se incrementa band.

Definición en la línea 41 del archivo LetraPort.java.

```
    {
        if(Greenfoot.mouseClicked(this))
        {
            son.play();
            band ++;
        }
    }
```

La documentación para esta clase fue generada a partir del siguiente fichero:

- LetraPort.java

Referencia de la Clase Mundo

Métodos públicos

- Mundo ()
- void act ()
- void menu ()
- void nivel1 ()
- void nivel2 ()
- void nivel3 ()
- void incrementaNivel ()
- void inicializa ()
- void pierdeVida (Personaje obj)
- void pierdeVida2 (Personaje obj)
- void reset ()
- void mandaTiempo ()
- void enviaPrimero ()
- void incrementa ()
- void mandaPolicia (int valor)
- void comiezaJuego ()
- void pierdes ()
- void enviaPerro ()

Descripción detallada

En esta clase esta todo lo relacionado al juego, desde su implementación básica de la creación de los objetos base hasta el incemento o decremento de vidas, niveles etc.

Autor:

Ulises Yamil Castorena Caldera

Versión:

1.0

Definición en la línea 10 del archivo Mundo.java.

Documentación del constructor y destructor

Mundo.Mundo ()

Constructor de la clase Mundo, inicializa los valores de las variables.

Definición en la línea 24 del archivo Mundo.java.

```
    {
        super(800, 400, 1);
        nivel = 1;
        pier = 0;
        band = 0;
        band2 = 0;
        primer = 0;
    }
```

Documentación de las funciones miembro

void Mundo.act ()

Método que realiza el actua del mundo, en el se inicializan los mundos y se crean dependiendo del nivel en el que esta.

Definición en la línea 37 del archivo Mundo.java.

```
{
    if(prim == 0)
       menu();
    else if(prim == 2)
    {
       if(band == 0)
          inicializa();
        else if(nivel == 1)
        {
            nivel1();
            enviaPrimero();
        }
        else if(nivel == 2)
        {
            nivel2();
            enviaPrimero();
        }
        else if(nivel == 3)
        {
            nivel3();
            enviaPrimero();
        }
    }
}
```

void Mundo.comiezaJuego ()

Método que indica que el juego ya esta inicializado y es posible comenzar.

Definición en la línea 280 del archivo Mundo.java.

```
{
    removeObjects(getObjects(null));
    prim = 2;
}
```

void Mundo.enviaPerro ()

Método que crea un objeto de tipo Perro y lo añade al mundo.

Definición en la línea 300 del archivo Mundo.java.

```
{
    if(Greenfoot.getRandomNumber(2) == 0)
       addObject(new Perro(0),799,336);
    else
       addObject(new Perro(1),799,64);
}
```

void Mundo.enviaPrimero ()

Envia por primera vez un objeto de la clase reloj, para asi poder iniciar.

Definición en la línea 247 del archivo Mundo.java.

```
{
    if(primer == 0)
    {
       addObject(new Reloj(),799,Greenfoot.getRandomNumber(260) + 64);
       addObject(new Perro(0),799,336);
       primer = 1;
    }
}
```

void Mundo.incrementa ()

Método que manda un mensaje al reloj para incrementarlo.

Definición en la línea 260 del archivo Mundo.java.

```
{
    tiem.incrementaTiempo();
}
```

void Mundo.incrementaNivel ()

Método que incrementa el nivel.

Definición en la línea 148 del archivo Mundo.java.

```
{
    nivel ++;
    if(nivel == 4)
    {
        removeObjects(getObjects(null));
        addObject(new Textos(2),400,200);
        Greenfoot.delay(100);
        removeObjects(getObjects(Textos.class));
        Greenfoot.stop();
    }
    addObject(new Textos(1),400,200);
    Greenfoot.delay(100);
    removeObjects(getObjects(null));
    band = 0;
}
```

void Mundo.inicializa ()

Método que inicializa y añade todos lo objetos necesarios para el juego.

Definición en la línea 168 del archivo Mundo.java.

```
{
    addObject(new Barra(),super.getHeight(),20);
    addObject(new Barra(),super.getHeight(),380);
    addObject(new Personaje(),98,332);
    addObject(new Letra("Tiempo"),30,70);
    addObject(new Letra("Metros"),30,110);
    addObject(new Letra("Vidas"),30,150);
    addObject(new Letra(nivel),super.getHeight(),20);
    addObject(new Numero(),35,130);
    vid = new Vidas();
    addObject(vid,35,170);
    tiem = new Tiempo();
    addObject(tiem,35,90);
    band = 1;
}
```

void Mundo.mandaPolicia (int *valor*)

Método que crea un objeto de tipo Policia y lo añade al mundo.

Parámetros:

valor	Indica si el objeto se añade en la parte superior o inferior.

Definición en la línea 269 del archivo Mundo.java.

```
{
    if(valor == 0)
       addObject(new Policia(valor),799,301);
    else
       addObject(new Policia(valor),799,100);
}
```

void Mundo.mandaTiempo ()

El método es llamado para saber si el reloj a llegado al limite de la pantalla ya así poder enviar otro.

Definición en la línea 239 del archivo Mundo.java.

```
{
    band2 = 1;
}
```

void Mundo.menu ()

Método que es utilizado para el menu, es el primer método en llamar.

Definición en la línea 66 del archivo Mundo.java.

```
{
    addObject(new Portada(),400,200);
    addObject(new LetraPort(),566,225);
    addObject(new LetraAy(0),624,293);
    prim = 1;
}
```

void Mundo.nivel1 ()

Método que añade los objetos enemigos, pero solo del nivel uno.

Definición en la línea 77 del archivo Mundo.java.

```
{
    if(band2 == 1)
    {
       if(Greenfoot.getRandomNumber(2) == 1)
       {
          addObject(new Reloj(),799,Greenfoot.getRandomNumber(260) + 64);
          band2 = 0;
       }
    }
}
```

void Mundo.nivel2 ()

Método que añade los objetos enemigos, pero solo del nivel dos.

Definición en la línea 92 del archivo Mundo.java.

```
{
    if(Greenfoot.getRandomNumber(100) == 0)
    {
        if(Greenfoot.getRandomNumber(2) == 0)
        {
           if(Greenfoot.getRandomNumber(2) == 0)
              addObject(new Perro(0),799,336);
           else
              addObject(new Perro(1),799,64);
        }
        else
        {
           if(Greenfoot.getRandomNumber(2) == 0)
              addObject(new Persona(0),799,317);
           else
              addObject(new Persona(1),799,83);
        }
    }
    if(band2 == 1)
    {
       if(Greenfoot.getRandomNumber(100) == 1)
       {
          addObject(new Reloj(),799,Greenfoot.getRandomNumber(260) + 64);
          band2 = 0;
       }
    }
```

```
    }
```

void Mundo.nivel3 ()

Método que añade los objetos enemigos, pero solo del nivel dos.

Definición en la línea 124 del archivo Mundo.java.

```
    {
        if(Greenfoot.getRandomNumber(100) == 0)
        {
            if(Greenfoot.getRandomNumber(2) == 0)
            {
               if(Greenfoot.getRandomNumber(2) == 0)
                  addObject(new Perro(0),799,336);
               else
                  addObject(new Perro(1),799,64);
            }
            else
            {
               if(Greenfoot.getRandomNumber(2) == 0)
                  addObject(new Persona(0),799,317);
               else
                  addObject(new Persona(1),799,83);
            }
        }
    }
```

void Mundo.pierdes ()

Método que es llamado cuando alguna condicion para perder se cumple.

Definición en la línea 289 del archivo Mundo.java.

```
    {
        removeObjects(getObjects(null));
        addObject(new Textos(3),400,200);
        Greenfoot.delay(100);
        pier = 1;
    }
```

void Mundo.pierdeVida (Personaje *obj*)

Remueve los objetos inicesarios para despues volverlos a intoduccir.

Parámetros:

obj	Un objeto que se debe eleminiar.

Definición en la línea 189 del archivo Mundo.java.

```
    {
        removeObject(obj);
        addObject(new Textos(0),400,200);
        Greenfoot.delay(100);
        removeObjects(getObjects(Textos.class));
        removeObjects(getObjects(Reloj.class));
        removeObjects(getObjects(Enemigo.class));
        removeObjects(getObjects(Numero.class));
        vid.disminuyeVida();
        if(pier == 0)
           reset();
        else
           Greenfoot.stop();
    }
```

void Mundo.pierdeVida2 (Personaje *obj*)

Remueve los objetos inicesarios para despues volverlos a intoduccir.

Parámetros:

obj	Un objeto que se debe eleminiar.

Definición en la línea 209 del archivo Mundo.java.

```
    {
        removeObject(obj);
        addObject(new Textos(0),400,200);
        Greenfoot.delay(100);
        removeObjects(getObjects(Textos.class));
        removeObjects(getObjects(Enemigo.class));
        removeObjects(getObjects(Reloj.class));
        removeObjects(getObjects(Numero.class));
        vid.disminuyeVida();
        tiem.pierdeTiempo();
        if(pier == 0)
           reset();
        else
           Greenfoot.stop();
    }
```

void Mundo.reset ()

Reinicia la ubicación de el usuario y del numero de metros faltantes.

Definición en la línea 229 del archivo Mundo.java.

```
    {
        addObject(new Personaje(),98,332);
        addObject(new Numero(),35,130);
        primer = 0;
    }
```

La documentación para esta clase fue generada a partir del siguiente fichero:

- Mundo.java

Referencia de la Clase Numero

Métodos públicos

- Numero ()
- void act ()
- void decrementaDist ()
- void limite ()

Descripción detallada

Clase donde se crea la distancia que tendra que recorrer el usuario.

Autor:

Ulises Yamil Castorena Caldera

Versión:

1.0

Definición en la línea 10 del archivo Numero.java.

Documentación del constructor y destructor

Numero.Numero ()

Constructor para la clase numero, iguala la cantidad de metros totales que debera recorrer el usuario, asi como un objeto que lo muestre.

Definición en la línea 18 del archivo Numero.java.

```
    {
        num = 5000;
        img = new GreenfootImage(Integer.toString(num),20,Color.BLACK,Color.WHITE);
        setImage(img);
    }
```

Documentación de las funciones miembro

void Numero.act ()

Método Act el cual coloca la imagen y decrementa las distancia.

Definición en la línea 28 del archivo Numero.java.

```
    {
        setImage(new GreenfootImage(Integer.toString(num),20,Color.BLACK,Color.WHITE));
        decrementaDist();
        limite();
    }
```

void Numero.decrementaDist ()

Método que va decrementando la distancia total.

Definición en la línea 38 del archivo Numero.java.

```
    {
        num --;
    }
```

void Numero.limite ()

Si la variable num llega a cero manda un mesnaje al mundo indicando que aumente el nivel.

Definición en la línea 46 del archivo Numero.java.

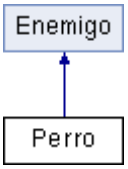
```
{
    if(num == 0)
    {
        ((Mundo)getWorld()).incrementaNivel();
    }
}
```

La documentación para esta clase fue generada a partir del siguiente fichero:

- Numero.java

Referencia de la Clase Perro

Diagrama de herencias de Perro



Métodos públicos

- Perro (int valor)
- void act ()
- void intersecta ()
- void remueveimagen ()

Otros miembros heredados

Descripción detallada

En esta clase se crean las opciones para la validación de los movimientos.

Autor:

Ulises Yamil Castorena Caldera

Versión:

1.0

Definición en la línea 9 del archivo Perro.java.

Documentación del constructor y destructor

Perro.Perro (int *valor*)

Contructor de la clase Perro, se igualan los valores y se crea el objeto para mostrar la imagen.

Parámetros:

valor	Dependiendo del valor que tenga va a ser la ubicación del objeto.

Definición en la línea 15 del archivo Perro.java.

```
    {
       ubic = valor;
       img = new GreenfootImage("perro1.png");
       img.scale(80,50);
       if(ubic == 0)
       {
           img.mirrorHorizontally();
           setImage(img);
       }
       else if(ubic == 1)
       {
           img.mirrorHorizontally();
           img.mirrorVertically();
           setImage(img);
       }
    }
```

Documentación de las funciones miembro

void Perro.act ()

Método Act de la clase Perro, en el se muestra la imagen, se mueve y se valida la opcion para remover la imagen.

Definición en la línea 36 del archivo Perro.java.

```
{
    setImage(img);
    super.mueve();
    remueveimagen();
}
```

void Perro.intersecta ()

Método que valida si un objeto de esta clase se topa con otro objeto.

Definición en la línea 46 del archivo Perro.java.

```
{
    Actor otro;
    otro = getOneIntersectingObject(Personaje.class);
    if(otro != null)
    {
        ((Mundo)getWorld()).mandaPolicia(ubic);
    }
}
```

void Perro.remueveimagen ()

Método que checa si el objeto ha llegado al borde y removerlo del mundo.

Definición en la línea 59 del archivo Perro.java.

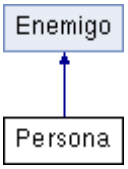
```
{
    if(this.getX() == 0)
    {
       ((Mundo)getWorld()).enviaPerro();
       getWorld().removeObject(this);
    }
}
```

La documentación para esta clase fue generada a partir del siguiente fichero:

- Perro.java

Referencia de la Clase Persona

Diagrama de herencias de Persona



Métodos públicos

- Persona (int valor)
- void act ()
- void remueveimagen ()
- void intersecta ()

Otros miembros heredados

Descripción detallada

En esta clase se crean las opciones para la validación de los movimientos.

Autor:

Ulises Yamil Castorena Caldera.

Versión:

1.0

Definición en la línea 9 del archivo Persona.java.

Documentación del constructor y destructor

Persona.Persona (int *valor*)

Contructor de la clase Persona, se igualan los valores y se crea el objeto para mostrar la imagen.

Parámetros:

valor	Dependiendo del valor que tenga va a ser la ubicación del objeto.

Definición en la línea 15 del archivo Persona.java.

```
    {
        ubic = valor;
        img = new GreenfootImage("peaton.png");
        if(ubic == 0)
        {
            setImage(img);
        }
        else if(ubic == 1)
        {
            img.mirrorVertically();
            setImage(img);
        }
    }
```

Documentación de las funciones miembro

void Persona.act ()

Método Act de la clase Persona, en el se muestra la imagen, se mueve y se valida la opcion para remover la imagen.

Definición en la línea 33 del archivo Persona.java.

```
{
    setImage(img);
    super.mueve();
    intersecta();
    remueveimagen();
}
```

void Persona.intersecta ()

Método que valida si un objeto de esta clase se topa con otro de la clase Personaje.

Definición en la línea 53 del archivo Persona.java.

```
{
    Actor otro;
    otro = getOneIntersectingObject(Personaje.class);
    if(otro != null)
    {
        band = 1;
        ((Mundo)getWorld()).mandaPolicia(ubic);
    }
}
```

void Persona.remueveimagen ()

Método que checa si el objeto ha llegado al borde.

Definición en la línea 44 del archivo Persona.java.

```
{
    if(this.getX() == 0 || band == 1)
       getWorld().removeObject(this);
}
```

La documentación para esta clase fue generada a partir del siguiente fichero:

- Persona.java

Referencia de la Clase Personaje

Métodos públicos

- Personaje ()
- void act ()
- void movimiento ()
- void interseccionPerro ()
- void interseccionPoli ()
- void cambia ()

Descripción detallada

Esta clase es la clase del usuario, todos los movimientos por parte del usuario son validados aqui.

Autor:

Ulises Yamil Castorena Caldera

Versión:

1.0

Definición en la línea 9 del archivo Personaje.java.

Documentación del constructor y destructor

Personaje.Personaje ()

Constructor para la clase Personaje, inicializa el valor de ubic el cual es la ubicación del vehiculo ademas de crear los objetos imagen y sonido, finalmente muestra la imagen del carro.

Definición en la línea 24 del archivo Personaje.java.

```
    {
        ubic = 0;
        band = 0;
        band2 = 0;
        img = new GreenfootImage("carro.png");
        img1 = new GreenfootImage("pv1.png");
        sd = new GreenfootSound("Lacuca.mp3");
        sd1 = new GreenfootSound("Haz.mp3");
        sd2 = new GreenfootSound("Teletrans.mp3");
        img.mirrorHorizontally();
        img.scale(90,60);
        setImage(img);
    }
```

Documentación de las funciones miembro

void Personaje.act ()

Método Act de la clase Personaje, en el se muestra la imagen del vehiculo ademas se encuentran los métodos para sus validaciones.

Definición en la línea 42 del archivo Personaje.java.

```
    {
        setImage(img);
        movimiento();
```

```
        if(band != 0)
        {
            cambia();
        }
        interseccionPerro();
        interseccionPoli();
        if(band2 != 0)
        {
            if(band2 == 1)
            {
               ((Mundo)getWorld()).pierdeVida(this);
            }
            else
            {
               ((Mundo)getWorld()).pierdeVida2(this);
            }
        }
    }
```

void Personaje.cambia ()

Método que cambia la ubcación del usuario.

Definición en la línea 117 del archivo Personaje.java.

```
    {
        if(ubic == 0)
        {
           if(getY() >= 200)
               setLocation(getX(),getY() - 10);
           if(getY() == 192)
              img.mirrorVertically();
           if(getY() <= 192 && getY() > 68)
            setLocation(getX(),getY() - 10);
           if(getY() == 62)
           {
              band = 0;
              ubic = 1;
           }
        }
        else if(ubic == 1)
        {
           if(getY() <= 200)
               setLocation(getX(),getY() + 10);
           if(getY() == 192)
              img.mirrorVertically();
           if(getY() >= 192 && getY() < 333)
             setLocation(getX(),getY() + 10);
           if(getY() == 332)
           {
              band = 0;
              ubic = 0;
           }
        }
    }
```

void Personaje.interseccionPerro ()

Método que verifica si el usuario ha chocado con algun enemigo de la clase perro.

Definición en la línea 95 del archivo Personaje.java.

```
    {
        Actor otro;
        otro = getOneIntersectingObject(Perro.class);
        if(otro != null)
           band2 = 1;
    }
```

void Personaje.interseccionPoli ()

Método que verifica si el usuario ha chocado con algun enemigo de la clase policia.

Definición en la línea 106 del archivo Personaje.java.

```
{
    Actor otro;
    otro = getOneIntersectingObject(Policia.class);
    if(otro != null)
       band2 = 2;
}
```

void Personaje.movimiento ()

Método donde es modificada la ubicación del personaje, dependiendo de la tecla que se presione.

Definición en la línea 68 del archivo Personaje.java.

```
{
    if(Greenfoot.isKeyDown("up") && ubic == 0)
    {
        sd2.stop();
        sd2.play();
        band = 1;
    }
    else if(Greenfoot.isKeyDown("down") && ubic == 1)
    {
        sd2.stop();
        sd2.play();
        band = 1;
    }
    else if(Greenfoot.isKeyDown("right"))
       move(4);
    else if(Greenfoot.isKeyDown("left"))
       move(-4);
    else if(Greenfoot.isKeyDown("w"))
       sd.play();
    else if(Greenfoot.isKeyDown("q"))
       sd1.play();
}
```

La documentación para esta clase fue generada a partir del siguiente fichero:

- Personaje.java

Referencia de la Clase Policia

Diagrama de herencias de Policia

Métodos públicos

- Policia (int valor)
- void act ()
- void intersecta ()
- void remueveimagen ()

Otros miembros heredados

Descripción detallada

En esta clase se crean las opciones para la validación de los movimientos.

Autor:

Ulises Yamil Castorena Caldera

Versión:

1.0

Definición en la línea 9 del archivo Policia.java.

Documentación del constructor y destructor

Policia.Policia (int *valor*)

Contructor de la clase Policia, se igualan los valores y se crea el objeto para mostrar la imagen.

Parámetros:

valor	Dependiendo del valor que tenga va a ser la ubicación del objeto.

Definición en la línea 15 del archivo Policia.java.

```
    {
        ubic = valor;
        img = new GreenfootImage("poli.png");
        img.scale(120,120);
        if(ubic == 0)
           setImage(img);
        else
        {
            img.mirrorVertically();
            setImage(img);
        }
    }
```

Documentación de las funciones miembro

void Policia.act ()

Método Act de la clase Policia, en el se muestra la imagen, se mueve y se valida la opcion para remover la imagen.

Definición en la línea 32 del archivo Policia.java.

```
{
    setImage(img);
    super.mueve();
    remueveimagen();
}
```

void Policia.intersecta ()

Método que valida si un objeto de esta clase se topa con otro objeto y si se topa con uno manda a un policía en su defensa.

Definición en la línea 42 del archivo Policia.java.

```
{
    Actor otro;
    otro = getOneIntersectingObject(Personaje.class);
    if(otro != null)
    {
        ((Mundo)getWorld()).mandaPolicia(ubic);
    }
}
```

void Policia.remueveimagen ()

Método que checa si el objeto ha llegado al borde y removerlo del mundo.

Definición en la línea 55 del archivo Policia.java.

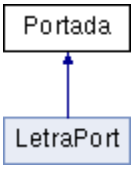
```
 {
     if(this.getX() == 0)
       getWorld().removeObject(this);
 }
```

La documentación para esta clase fue generada a partir del siguiente fichero:

- Policia.java

Referencia de la Clase Portada

Diagrama de herencias de Portada



Métodos públicos

- Portada ()
- void act ()

Atributos protegidos

- GreenfootImage img

Descripción detallada

Clase en la cual se crea la portada y se muestra.

Autor:

Ulises Yamil Castorena Caldera

Versión:

1.0

Definición en la línea 9 del archivo Portada.java.

Documentación del constructor y destructor

Portada.Portada ()

Constructor para la clase portada, unicamnete se crea el objeto y se dibuja la imagen.

Definición en la línea 16 del archivo Portada.java.

```
{
    img = new GreenfootImage("Portada.png");
    setImage(img);
}
```

Documentación de las funciones miembro

void Portada.act ()

Método Act de la clase que muestra la imagen.

Reimplementado en LetraPort.

Definición en la línea 25 del archivo Portada.java.

```
{
    setImage(img);
}
```

Documentación de los datos miembro

GreenfootImage Portada.img `[protected]`

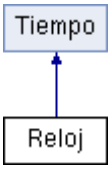
Definición en la línea 11 del archivo Portada.java.

La documentación para esta clase fue generada a partir del siguiente fichero:

- Portada.java

Referencia de la Clase Reloj

Diagrama de herencias de Reloj



Métodos públicos

- Reloj ()
- void act ()
- void chocaPersonaje ()
- void limite ()

Otros miembros heredados

Descripción detallada

Clase que muestra el objeto para poder incrementar el tiempo.

Autor:

Ulises Yamil Castorena Caldera

Versión:

1.0

Definición en la línea 9 del archivo Reloj.java.

Documentación del constructor y destructor

Reloj.Reloj ()

Constructor para la clase Reloj, crea el objeto para mostrarlo, tambien lo escala porque el tamaño original es mas grande.

Definición en la línea 17 del archivo Reloj.java.

```
    {
        img = new GreenfootImage("reloj.png");
        img.scale(50,50);
        band = 0;
        setImage(img);
    }
```

Documentación de las funciones miembro

void Reloj.act ()

Método Act el cual coloca la imagen, mueve el objeto y ademas llama los métodos que validan si ha chocado con el jugador o si llego al límite de la pantalla.

Reimplementado de Tiempo.

Definición en la línea 28 del archivo Reloj.java.

```
    {
        setImage(img);
        move(-4);
        chocaPersonaje();
        limite();
    }
```

void Reloj.chocaPersonaje ()

Verifica si el usuario choca con este objeto.

Definición en la línea 39 del archivo Reloj.java.

```
    {
        Actor act = getOneIntersectingObject(Personaje.class);
        if(act != null)
        {
           ((Mundo)getWorld()).incrementa();
           band = 1;
        }
    }
```

void Reloj.limite ()

Checa si este objeto llega al limite de la pantalla.

Definición en la línea 52 del archivo Reloj.java.

```
    {
        if(getX() == 0 || band == 1)
        {
           ((Mundo)getWorld()).mandaTiempo();
           getWorld().removeObject(this);
        }
    }
```

La documentación para esta clase fue generada a partir del siguiente fichero:

- Reloj.java

Referencia de la Clase Textos

Métodos públicos

- Textos (int valor)
- void act ()

Descripción detallada

Clase que es la encargada de mostrar los textos del juego.

Autor:

Ulises Yamil Castorena Caldera

Versión:

1.0

Definición en la línea 9 del archivo Textos.java.

Documentación del constructor y destructor

Textos.Textos (int *valor*)

Constructor para la clase Textos.

Parámetros:

valor	Es un indicador para saber que clase de texto debe mostrar.

Definición en la línea 17 del archivo Textos.java.

```
    {
        if(valor == 0)
           img = new GreenfootImage("pv1.png");
        else if(valor == 1)
           img = new GreenfootImage("sdn.png");
        else if(valor == 2)
           img = new GreenfootImage("ganaste.png");
        else if(valor == 3)
           img = new GreenfootImage("go.png");
        setImage(img);
    }
```

Documentación de las funciones miembro

void Textos.act ()

Método act de la clase, unicamente muestra la imagen.

Definición en la línea 33 del archivo Textos.java.

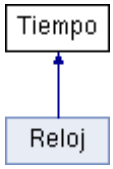
```
    {
        setImage(img);
    }
```

La documentación para esta clase fue generada a partir del siguiente fichero:

- Textos.java

Referencia de la Clase Tiempo

Diagrama de herencias de Tiempo



Métodos públicos

- Tiempo ()
- void act ()
- void incrementaTiempo ()
- void decrementaTiempo ()
- void pierdeTiempo ()
- void pierdePorTiempo ()

Atributos protegidos

- int tiem

Descripción detallada

Clase en la cual se crea el objeto que muestra la cantidad de tiempo restante.

Autor:

Ulises Yamil Castorena Caldera

Versión:

1.0

Definición en la línea 10 del archivo Tiempo.java.

Documentación del constructor y destructor

Tiempo.Tiempo ()

Constructor para la clase tiempo, se iguala el tiempo al valor que se necesite al inicio y se crea la imagen para mostrarlo.

Definición en la línea 20 del archivo Tiempo.java.

```
    {
        tiem = 5;
        band = 0;
        band2 = 0;
        img = new GreenfootImage(Integer.toString(tiem)+"s",20,Color.BLACK,Color.WHITE);
        setImage(img);
    }
```

Documentación de las funciones miembro

void Tiempo.act ()

Método Act que muestra la imagen ademas de llamar al método para poder validar si el tiempo se debe de reducir.

Reimplementado en Reloj.

Definición en la línea 32 del archivo Tiempo.java.

```
    {
        setImage(new
GreenfootImage(Integer.toString(tiem)+"s",20,Color.BLACK,Color.WHITE));
        decrementaTiempo();
        pierdePorTiempo();
    }
```

void Tiempo.decrementaTiempo ()

Método que valida en que momento el tiempo se debe de reducir.

Definición en la línea 51 del archivo Tiempo.java.

```
    {
        if(band == 0)
        {
            tiem --;
            band = 1;
        }
        else
        {
            if(band2 == 70)
            {
               band = 0;
               band2 = 0;
            }
            else
               band2 ++;
        }
    }
```

void Tiempo.incrementaTiempo ()

Método que incrementa el tiempo limite.

Definición en la línea 42 del archivo Tiempo.java.

```
    {
        if(tiem < 500)
           tiem = tiem + 15;
    }
```

void Tiempo.pierdePorTiempo ()

Método que indica si el tiempo llega a cero, manda un mensaje al mundo que muestre que perdio.

Definición en la línea 81 del archivo Tiempo.java.

```
    {
        if(tiem <= 0)
           ((Mundo)getWorld()).pierdes();
    }
```

void Tiempo.pierdeTiempo ()

Cuando este método es llamado, el tiempo se decrementa en 30.

Definición en la línea 73 del archivo Tiempo.java.

```
    {
```

```
        tiem = tiem - 30;
    }
```

Documentación de los datos miembro

int Tiempo.tiem `[protected]`

Definición en la línea 12 del archivo Tiempo.java.

La documentación para esta clase fue generada a partir del siguiente fichero:

- Tiempo.java

Referencia de la Clase Vidas

Métodos públicos

- Vidas ()
- void act ()
- void disminuyeVida ()

Descripción detallada

Clase que muestra el valor de las vidas actuales.

Autor:

Ulises Yamil Castorena Caldera

Versión:

1.0

Definición en la línea 10 del archivo Vidas.java.

Documentación del constructor y destructor

Vidas.Vidas ()

Constructor para la clase vidas, en el se inicializa la cantidad total de vidas, ademas se convierte las vidas a cadena para poder mostrarlas.

Definición en la línea 20 del archivo Vidas.java.

```
{
    vida = 5;
    img = new GreenfootImage(Integer.toString(vida),20,Color.WHITE,Color.BLACK);
    setImage(img);
}
```

Documentación de las funciones miembro

void Vidas.act ()

Método Act de la clase, solamente muestra la imagen de la vida.

Definición en la línea 30 del archivo Vidas.java.

```
{
    setImage(new GreenfootImage(Integer.toString(vida),20,Color.WHITE,Color.BLACK));
}
```

void Vidas.disminuyeVida ()

Método que disminuye una vida y realiza la validación si el usuario llega a cero vidas.

Definición en la línea 38 del archivo Vidas.java.

```
{
    vida --;
    if(vida <= 0)
        ((Mundo)getWorld()).pierdes();
}
```

La documentación para esta clase fue generada a partir del siguiente fichero:

- Vidas.java