MANUAL DEL PROGRAMADOR

Juego: Circus Circus II

Materia: Programación orientada a objetos

Alumnos:

Cisneros Torres Luis

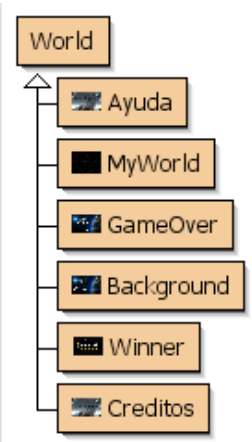
López Ortiz Z. Ivonne

Profesor:

Ing. Cesar Augusto Ramírez Gámez

Mayo de 2017

El proyecto fue realizado en Greenfoot con este tenemos la ventaja de que se tienen clases ya definidas y con métodos igualmente definidos en cada una de ellas.



De la clase World creamos diferentes sub-clases, que nos ayuda a la movilidad entre diferentes escenarios tenemos las clases Ayuda, GameOver, Winner, Creditos y Background, de estas solo se muestran imágenes y botones necesarios para inicializar el juego y llevarnos al escenario donde se desarrolla el juego, la clase MyWorld.

Ejemplo clase Background:

```
import greenfoot.*;  // (World, Actor, GreenfootIma

/**
 * Fondo de la imagen al inicio del juego
 * con sus respectivas opciones
 */
public class Background extends World
{
    public Background()
    {
        super(800, 500, 1);
        addObject(new Start(),570,90);
        addObject(new Help(),510,200);
        addObject(new Credits(),480,330);
    }
}
```

Clase MyWorld donde se crean los objetos con los cuales son para jugar:

```
import greenfoot.*;  // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)
/**
 * Mundo donde se encuentra el juego
 * Se crea un contador de puntos
 * y los objetos que estaran en el juego
 */
public class MyWorld extends World
{
    Counter counter = new Counter();
    Return ret = new Return(0);

    /**
     * Constructor for objects of class MyWorld.
     *
     */
    public MyWorld()
    {
        // Create a new world with 800x500 cells with a cell size of 1x1 pixels.
        super(800, 500, 1);
        addObject(new Sentence(ret),800,500);
        addObject(counter, 450,25);
        addObject(new Personaje(ret),200,200);
        addObject(ret,750,470);
    }

    public Counter getCounter(){
        return counter;
    }
}
```

La siguiente clase que hereda a otras clases es Actor. En esta encontramos diferentes clases con métodos igualmente diferentes, utilizando el polimorfismo de métodos gracias a las clases abstractas.

De las principales tenemos a Monito que hereda a Personaje y a Enemigo que esta última a su vez hereda a Nave y Chango, en las cuales encontramos sus métodos heredados.

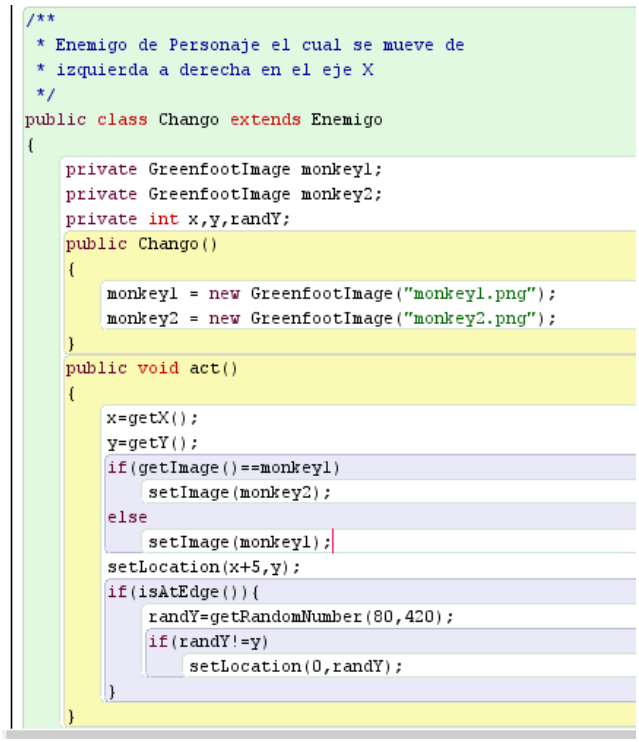
Personaje es en la cual tenemos a los movimientos del payasito haciendo el debido cambio de sprites:

```
}
if(Greenfoot.isKeyDown("right"))   //Move in right side
    {
     if(getImage()==sideR1)
        setImage(sideR2);
     else
        setImage(sideR1);
     if(x<780)
        setLocation(x+3,y);
    }
else
    if(Greenfoot.isKeyDown("left"))
    {
        if(getImage()==sideL1)
            setImage(sideL2);
        else
            setImage(sideL1);
        if(x>28)
            setLocation(x-3,y);
     }
     else
        if(Greenfoot.isKeyDown("up"))
        {
            if(getImage()==back1)
                setImage(back2);
            else
                setImage(back1);
            if(y>80)
                setLocation(x,y-3);
             }
```

Al igual si el personaje “ve” (método heredado de Monito) a un enemigo este le restara puntos y si se queda sin vidas termina juego.

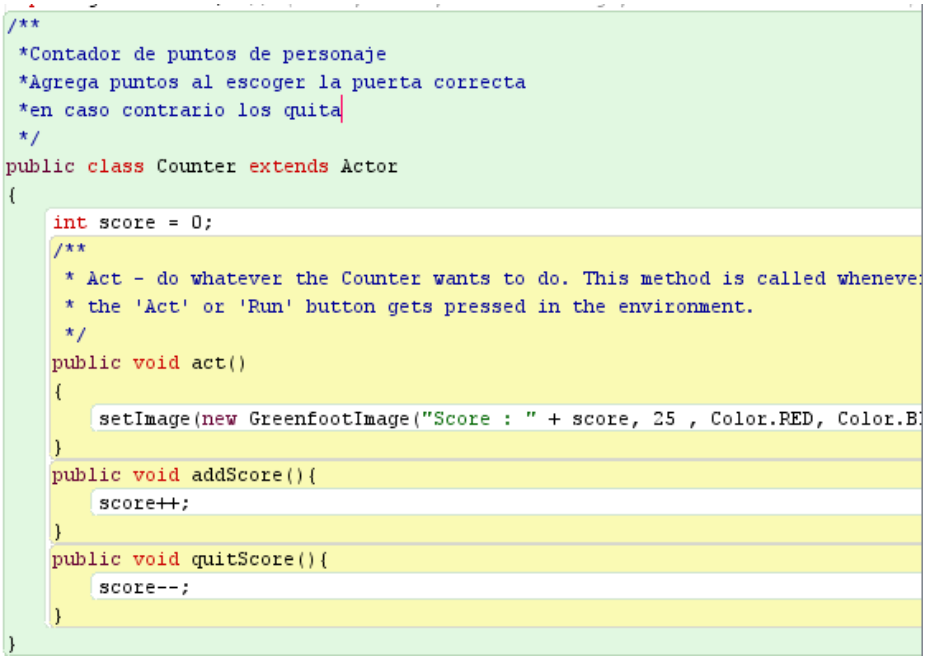
```
            }
if(canSee(Chango.class)){
    aux = getOneIntersectingObject(Chango.class);
    aux2=aux;
    getWorld().removeObject(aux);
    getWorld().addObject(aux2,0,300);
    vidas--;
    getWorld().removeObject(cor[vidas]);
}
if(canSee(Nave.class)){
    aux3 = getOneIntersectingObject(Nave.class);
    aux4=aux3;
    getWorld().removeObject(aux3);
    getWorld().addObject(aux4,300,0);
    vidas--;
    getWorld().removeObject(cor[vidas]);
}
if(vidas == 0){
    Greenfoot.setWorld(new GameOver());
}
```

De los enemigos tenemos a Chango, por ejemplo, con sus propios métodos e igualmente parecidos a los de Nave, solo que tienen de diferencia en el eje que se encuentran cada uno.

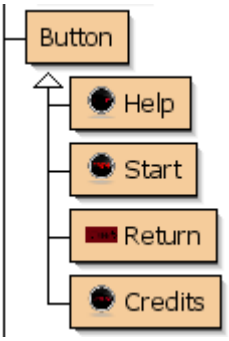


```
/**
 * Enemigo de Personaje el cual se mueve de
 * izquierda a derecha en el eje X
 */
public class Chango extends Enemigo
{
    private GreenfootImage monkey1;
    private GreenfootImage monkey2;
    private int x,y,randY;
    public Chango()
    {
        monkey1 = new GreenfootImage("monkey1.png");
        monkey2 = new GreenfootImage("monkey2.png");
    }
    public void act()
    {
        x=getX();
        y=getY();
        if(getImage()==monkey1)
            setImage(monkey2);
        else
            setImage(monkey1);
        setLocation(x+5,y);
        if(isAtEdge()){
            randY=getRandomNumber(80,420);
            if(randY!=y)
                setLocation(0,randY);
        }
    }
```

Tenemos a la Clase Counter que hace el conteo de puntos y que se utiliza en la clase Personaje



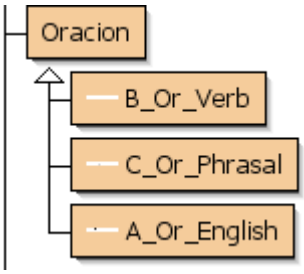
```
/**
 *Contador de puntos de personaje
 *Agrega puntos al escoger la puerta correcta
 *en caso contrario los quita
 */
public class Counter extends Actor
{
    int score = 0;
    /**
     * Act - do whatever the Counter wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public void act()
    {
        setImage(new GreenfootImage("Score : " + score, 25 , Color.RED, Color.BL
    }
    public void addScore(){
        score++;
    }
    public void quitScore(){
        score--;
    }
}
```



Clases las cuales ayudan a navegar dentro de los diferentes mundos ya mencionados en la parte de arriba.

```
/**
 * Boton que manda al mundo de Ayuda
 */
public class Help extends Button
{
    public void act()
    {
        if (Greenfoot.mouseClicked(this)){
            Greenfoot.setWorld(new Ayuda());
        } // Add your action code here.
    }
}
```

En la clase "Return" se implementa el sonido del juego, ya que este botón es agregado en casi todos los mundos del juego y es más fácil el control del sonido desde esta clase. una vez que éste botón es presionado, o que acaba el mundo, se detiene la canción de fondo.



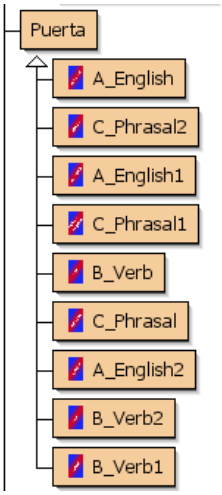
Clases que son utilizadas para que se muestre un arreglo de imágenes en la parte superior que serán las que tendrá que traducir el Personaje.

A_Or_English es de nivel 1, B_Or_Verb son las oraciones de nivel 2 y por ultimo C_Or_Phrasal del nivel 3.

```
/**
 * Clase que muestra las palabras a traducir
 * del nivel 1
 */
public class A_Or_English extends Oracion
{
    private String []name={"appliance.png","disrepute.png","largesse.png","

    /**
     * Act - do whatever the A_Or_English wants to do. This method is calle
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public A_Or_English(){}

    public void returnOrc(int pos){
        setImage(name[pos]);
        setLocation(150,25);
    }
}
```



Al igual que las oraciones que acabamos de ver, tienen la misma función, mostrar en pantalla al usuario diferentes imágenes, pero en cambio de Oración estas clases que heredan de puerta y son las respuestas a las

imágenes de las clases de Oracion, son con las cuales el Personaje interactúa y se sabrá si se le suma puntos o se le restan.

De estas clases las respuestas correctas se centran en 3 clases: A_English2, B_Verb2 y C_Phrasal2.

```
/**
 * Arreglo imagenes de respuestas de nivel 1
 * Respuestas 3 (correctas)
 */
public class A_English2 extends Puerta
{
    private String []Op3={"Appliance3rR.png","Disrepute3rR.png","Largesse3rR.
    public A_English2()
    {
    }
     public void returnImg2(int pos, int x, int y)
    {
        setImage(Op3[pos]);
        setLocation(x,y);
    }
}
```

Sentence: Clase que ayuda a que las clases de Puerta (la forma aleatoria en que se muestran) y Oración se unan para crear la parte principal del juego que es la traducción. Así que en esta clase se crean los objetos de las clases ya mencionadas con anterioridad.

Y donde se crean los métodos divididos en cada nivel.

```
public Sentence(){
    band=cont=enable=0;
    aux=1;

    varE1=new A_English();
    varE2=new A_English1();
    varE3=new A_English2();
    oracionE=new A_Or_English();

    varV1= new B_Verb();
    varV2= new B_Verb1();
    varV3= new B_Verb2();
    oracionV=new B_Or_Verb();

    varP1= new C_Phrasal();
    varP2= new C_Phrasal1();
    varP3= new C_Phrasal2();
    oracionP=new C_Or_Phrasal();
}
public void act(){
    switch(aux){
        case 1: aux = nivel1(); break;

        case 2: aux = nivel2(); break;

        case 3: aux = nivel3(); break;

        case 4: Greenfoot.setWorld(new Winner()); break;
    }
}
```

```
public int nivel1(){
    World myWorld = getWorld();
    MyWorld MyWorld = (MyWorld)myWorld;
    Counter counter = MyWorld.getCounter();
    int au=1;
    if(enable==0){
        getWorld().addObject(new Nivel(1),50,475);
        getWorld().addObject(oracionE,150,25);
        getWorld().addObject(varE1,getRandomNumber(50,750),getRandomNumber(150,350));
        getWorld().addObject(varE2,getRandomNumber(50,750),getRandomNumber(150,350));
        getWorld().addObject(varE3,getRandomNumber(50,750),getRandomNumber(150,350));
        enable=1;
    }
    if(Obj1==0){
        if(varE1.canSee(Personaje.class)){
            getWorld().removeObject(varE1);
            Obj1=1;
            counter.quitScore();
        }
    }
    if(Obj2==0){
        if(varE2.canSee(Personaje.class)){
            getWorld().removeObject(varE2);
            Obj2=1;
            counter.quitScore();
        }
    }
    if(cont<4){
        if(varE3.canSee(Personaje.class)){
            cont++;
            oracionE.returnOrc(cont);
            counter.addScore();
```

La variable “enable” sirve para que el objeto solo se cree una sola vez dentro del nivel, los métodos canSee son por si el personaje ve la puerta incorrecta, la quite e igualmente reste un punto.

```
            if(Obj1==1){
                getWorld().addObject(varE1,getRandomNumber(50,750),getRandomNumber(150,350));
                Obj1=0;
            }
            if(Obj2==1){
                getWorld().addObject(varE2,getRandomNumber(50,750),getRandomNumber(150,350));
                Obj2=0;
            }
            varE1.returnImg(cont,getRandomNumber(50,750),getRandomNumber(150,350));
            varE2.returnImg1(cont,getRandomNumber(50,750),getRandomNumber(150,350));
            varE3.returnImg2(cont,getRandomNumber(50,750),getRandomNumber(150,350));
        }
    }
    else{
        if(varE3.canSee(Personaje.class)){
            if(cont>=4){
                cont++;
                if(cont==10){
                    counter.addScore();
                    getWorld().removeObject(varE1);
                    getWorld().removeObject(varE2);
                    getWorld().removeObject(varE3);
                    au = 2;
                    cont=Obj1=Obj2=enable=0;
                }
            }
        }
    }
    return(au);
```

Al igual si ve la puerta 3 (que son las correctas) elimina las otras puertas y el contador se incrementa para que siga con la siguiente palabra a traducir.

La variable au sirve para hacer el cambio al siguiente nivel, como se ve en el switch del método act. Este último método se repite para nivel 2 y el 3, cambiando a las variables correspondientes.

Están las Clases Nivel y Corazón que son efectos visuales de las variables mencionadas como sería la variable de los niveles y la de las vidas.

Por último, en la clase de ayuda se instancia un objeto de la clase ArchivoMuestra, el cuál

```
public String leerTextoArchivo() {
  String texto = "";
  FileReader archivo = null;
  String linea = "";
  try {
    archivo = new FileReader("Ayuda.txt");
    BufferedReader lector = new BufferedReader(archivo);
    while ((linea = lector.readLine()) != null) {
      texto += linea + "\n";
    }
  } catch (FileNotFoundException e) {
    throw new RuntimeException("Archivo no encontrado");
  } catch (IOException e) {
    throw new RuntimeException("Ocurrio un error de entrada/salida");
  } finally {
    if (archivo != null) {
      try {
        archivo.close();
      } catch (IOException e) {
        e.printStackTrace();
      }
    }
  }
  return texto;
}
```

abre el archivo "Ayuda.txt" y va leyendo línea por línea el contenido del archivo concatenándolo en una variable de tipo String.

Una vez que haya terminado este proceso lo regresa. Se hizo uso del try, catch y finally, para que intente hacer la instrucción (try), si el archivo no se encuentra u ocurre otro error, manda un mensaje de "archivo no encontrado" (catch(FileNotFound)) u "ocurrió un error de entrada o salida" (catch(IOException)) respectivamente. Finalmente, se cierra el archivo (finally).