 Universidad Autónoma de San Luis Potosí

Facultad de Ingeniería

Área de Computación e Informática

Programación Orientada a Objetos



***Manual del programador***

**Profesor:**

***Cesar Augusto Puente Montejano***

**Integrantes:**

***González Reyes Felipe de Jesús***

***Hernández Cleto José Ernesto***

***Vázquez Ávila Abimael***

**\* Importación de clases**: es la sección inicial, en fondo blanco. Mediante la cláusula ‘import‘, podemos incluir las clases que necesitemos en nuestro programa  para que todo funcione correctamente. Las clases suelen encontrarse agrupadas de forma lógica en librerías o packages(paquetes). Un paquete es por tanto un conjunto de clases ya construidas que podemos utilizar en nuestra aplicación, lo que nos ahorra programar todas las funcionalidades desde cero.

**import greenfoot.\*;  // (World, Actor, GreenfootImage, and Greenfoot)**

**import java.util.List;**

**import java.util.ArrayList;**

**\* Declaración de la clase**: Comienza con unos **comentarios** (qué hace la clase, su autor, versión…), cuyo inicio y final se marcan con los caracteres ‘/\* ‘ y ‘\*/’ (es muy recomendable incluir comentarios a lo largo del programa para hacer más entendible nuestro código) y continúa con:

**public class MyWorld extends World**

**{**

**/\*\***

**\* Constructor for objects of class MyWorld.**

**\***

**\*/**

**\* Declaración de atributos:** Los atributos son las características individuales que diferencian un objeto de otro y determinan su apariencia, estado u otras cualidades. Los atributos se guardan en variables denominadas de instancia, y cada objeto particular puede tener valores distintos para estas variables.

Las variables de instancia también denominados miembros dato, son declaradas en la clase, pero sus valores son fijados y cambiados en el objeto.

Además de las variables de instancia hay variables de clase, las cuales se aplican a la clase y a todas sus instancias.

**private int x=0;// Controlando posición x de botones en conjunto**

**private int y=0;// Controlando posición y de botones en conjunto**

**private Hombre hombre = new Hombre();**

**private Escenario1 Tgamer;**

**private Escenario2 Tabb;**

**private Escenario3 Tbou;**

**\* Constructor**: es el proceso que crea un objeto a partir de una clase y que se ejecuta automáticamente cada vez que instanciamos un objeto. Tiene la misma estructura que un método aunque con algunas particularidades: debe tener siempre el mismo nombre que la clase y no tener ningún *return type*.

No es necesario declarar siempre un constructor, ya que toda clase tiene uno que inicializa los campos del objeto con valores por defecto. Si queremos personalizar ese proceso de creación, entonces podemos escribir uno o varios constructores, a los que a su vez podemos pasar o no parámetros.

**public Menu()**

**{**

**// Create a new world with 600x400 cells with a cell size of 1x1 pixels.**

**// Agregando y posicionando botones**

**super(900, 600, 1);**

**/\*addObject(new BotonOpciones(),x+201,y+59);**

**addObject(new BotonSalir(),x+60,y+195);**

**addObject(new BotonAyuda(),x+60,y+333);**

**addObject(new BotonJugar(),x+270,y+267);\*/**

**CreaBotones();**

**}**

\* **Métodos**: todo el código restante de esta clase, son las definiciones de sus métodos. Cada uno aparece en un recuadro amarillo, con una estructura muy parecida:

* Comentario indicando que realiza ese método
* Declaración del método: ‘public void act()‘, ‘public boolean foundLeaf()‘, ‘public void eatLeaf()‘,…
* encerrados entre llaves {}, las instrucciones que han de ejecutarse

Aunque no podemos entrar en detalle de todo el código, vamos a revisar por ejemplo el primer método que aparece: act() :

**public void act()**

**{**

**if(Greenfoot.getMouseInfo()!=null)//instruccion que checa la informacion del mouse**

**{**

**if(Greenfoot.mousePressed(BotonJugar))//Checa si se sio click en el boton jugar**

**{**

**Greenfoot.setWorld(new MyWorld());**

**}**

**if(Greenfoot.mousePressed(BotonAtras))**

**{**

**Greenfoot.setWorld(new Menu());**

**}**

**if(Greenfoot.mousePressed(BotonCreditos))//Checa si se dio click en el boton creditos**

**{**

**RemoveObj();**

**showText("Programacion Orientada a Objetos: ", x+201,y+59);**

**showText("Tarjeta Bancaria ", x+201,y+105);**

**showText("Cleto ", x+201,y+151);**

**showText("Felipe ", x+201,y+201);**

**showText("Abimael ", x+201,y+250);**

**addObject(BotonAtras,x+500,y);**

**}**

**if(Greenfoot.mousePressed(BotonInstr))**

**{**

**RemoveObj();**

**showText("Estas son las instrucciones ", x+201,y+59);**

**addObject(BotonAtras,x+500,y);**

**}**

**}**

**}**

**Herencia:** Esta nueva clase tendrá (heredará) los atributos y métodos de la anterior, pudiendo mantener o modificar dichos elementos o añadir unos nuevos si así lo necesita, para que se adapten a sus características y comportamiento particulares (recordemos la expresión ‘extiende a…’). Un aspecto que debemos tener en cuenta es que estas relaciones de jerarquía deben tener sentido, pero no tienen que ser idénticas y recrear todas las que existen en el mundo real.

**public class Ab extends World**

**{**

**public Ab()**

**{**

**// Create a new world with 600x400 cells with a cell size of 1x1 pixels.**

**super(600, 473, 1);**

**CreaF();**

**}**

**public class Bout extends World**

**{**

**public Bout()**

**{**

**// Create a new world with 600x400 cells with a cell size of 1x1 pixels.**

**super(600, 400, 1);**

**CreaBoutique();**

**}**

**public class Game extends World**

**{**

**public Game()**

**{**

**// Create a new world with 600x400 cells with a cell size of 1x1 pixels.**

**super(900, 600, 1);**

**CreaGame();**

**}**