Práctica 8: Juego del comecocos

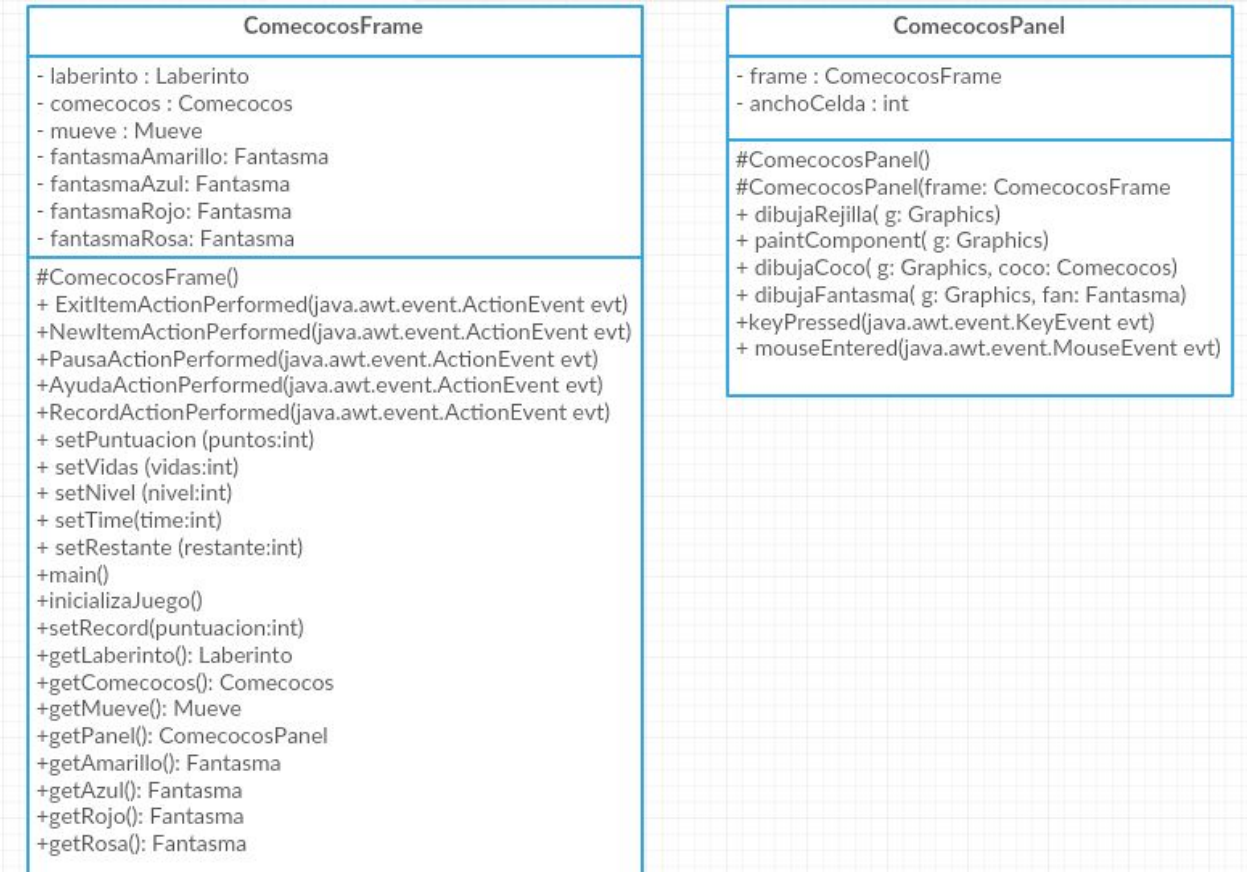
UNIVERSIDAD DE GRANADA

Autor: David Algarra Medina
Correo: amdavid96@correo.ugr.es
Asignatura: Complementos de Programación

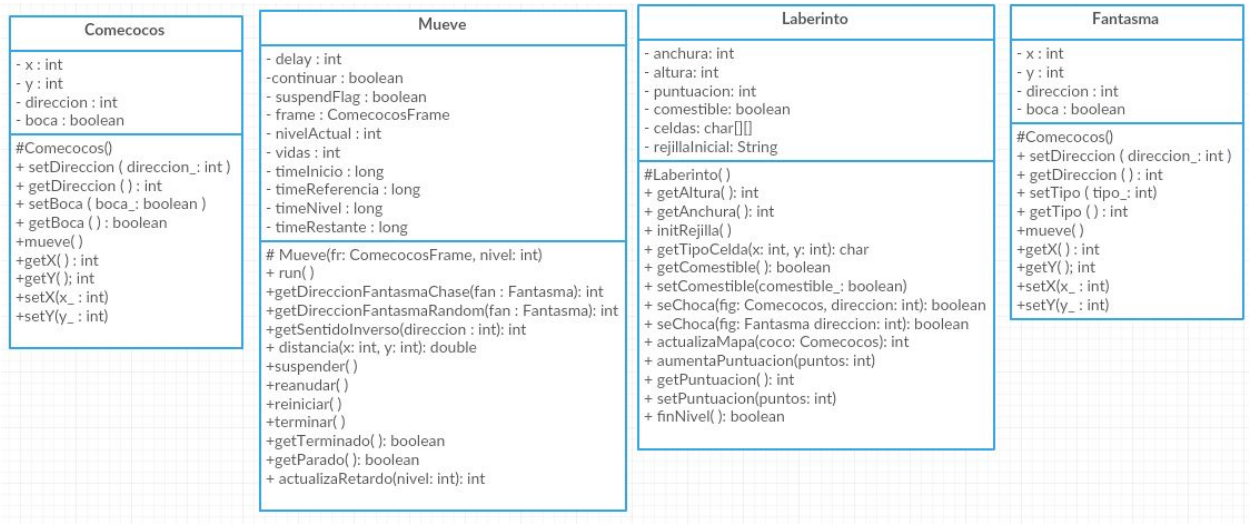
1. Diagrama UML

La clase ComecocosFrame tiene una relación de asociación con todas las demás clases ya que es la principal.

Paquete guicomecocos:



Paquete data:



Justificación de la solución

En este apartado justificaremos por qué hemos escogido cada una de las clases y analizaremos su funcionalidad en el programa. Analizaremos primero las clases en principio más sencillas, y finalizaremos con las clases más importantes que hacen que ejecuten el programa o que usan otras clases para funcionar. Empecemos con las clases del paquete ‘data’ y por último las dos del paquete ‘guicocos’.

Comecocos

Clase usada para representar al comecocos de nuestro programa, contiene lo necesario para poder controlarlo.
Como datos miembro necesitamos saber las coordenadas dentro del mapa, qué dirección lleva y una variable booleana boca, necesaria para saber si está abierta o cerrada.
En esta clase hay 4 variables estáticas finales que van a ser útiles para determinar la manera de indicar la dirección en nuestro programa. Estas variables son enteros, y cada una representa una dirección, al ser públicas todas las clases pueden usar esta referencia para especificar una dirección.
Los métodos que implementa son en su mayoría métodos para obtener o modificar los datos miembro, es decir métodos get() y set().
El único método que se sale de la norma es el método mueve, que hace que se mueva el comecocos dependiendo de la dirección que indique su dato miembro ‘direccion’.

Fantasma

La clase fantasma es igual que la clase Comecocos, tan solo cambia que debemos especificarle el tipo de fantasma ya que hay puede ser de 4 tipos(amarillo, azul, rojo y rosa) y no necesitamos la variable booleana boca porque los fantasmas no implementan ninguna animación al moverse por el mapa.
Para especificar el tipo de fantasma, tenemos 4 variables estáticas finales y públicas para que cualquier clase pueda acceder, entonces al llamar al constructor de Fantasma hay que especificarle uno de los cuatro tipos que hay en las variables finales.

Laberinto

Clase que contiene todo lo necesario relativo al mapa en el que se desarrolla el juego. Es la encargada de establecer la situación del mapa(donde se encuentra cada elemento) y se encarga de contener la puntuación.
Como datos miembro tiene una matriz de altura y anchura especificada, esta matriz es la base del laberinto, ya que en cada posición se indica qué elemento hay, puede ser un muro(hay muchos tipos de muros), estar vacía, contener un punto pequeño, un punto grande o que sea un puerta.
Un dato miembro rejillaInicial que es un String de longitud altura*anchura de la matriz, donde se describe el contenido inicial del mapa a elaborar.
En esta clase, implementamos una variable booleana 'comestible' que especifica si estamos en un estado de comestible, donde PacMan persigue a los fantasmas o al revés.
Además de los métodos get() y set() para acceder a los datos miembro, esta clase implementa los siguientes métodos:

- assignTipoCelda() para establecer que elemento hay en una celda.
- initRejilla() para iniciar el mapa según lo que marca el dato miembro rejillaInicial.
- seChoca(Comecocos, direccion). Dado un objeto y comecocos y una dirección, determinar si se mueve en esa posición se choca o no.

Aquí hay una relación de dependencia entre la clase Comecocos y la clase Laberinto.

- actualizaMapa(Comecocos). De nuevo se da un objeto Comecocos. Este método se usa para saber si el comecocos pasa sobre un punto grande o pequeño, y si es así que ese elemento pase a estar vacío.
- finNivel(). Para comprobar si se ha acabado de obtener todos los puntos del mapa.

Mueve

Clase troncal en nuestro programa, es una clase que implementa Runnable, por tanto será la que implemente la hebra principal de nuestro programa para ejecutar el movimiento del

Comecocos y los fantasmas, es la encargada de controlar el tiempo que pasa y el nivel en el que se encuentra

Tiene una relación de asociación con la clase ComecocosFrame, veremos más adelante esta clase, ya que un dato miembro es un objeto de esta clase.
Y tiene una relación de dependencia con Laberinto, Comecocos y Fantasmas ya que usa sus métodos.

Como datos miembro tiene el objeto ComecocosFrame ya comentado antes, variables necesarias para el control del tiempo, variables para controlar los distintos flags si se suspende o se termina la hebra, el numero de vidas, el nivel, una variable para determinar el retardo que está relacionada con el nivel.

El método principal es el método sobrescrito run() donde aquí se comprende todo el movimiento de los fantasmas y todas las llamadas a métodos de la clase y de otras clases, con el fin de establecer la puntuación, el nivel, saber si se termina el juego, establecer el tiempo.

En esta clase usamos métodos auxiliares para el método run() y hacer el código más funcional. Como por ejemplo tenemos funciones para saber cuál debe ser la dirección que toman los fantasmas.
Además, se implementan funciones como reanudar(), suspender(), reiniciar() y terminar() que tienen control sobre la ejecución de la hebra.

ComecocosPanel

Esta es una de las dos clases que pertenece al paquete de 'guicocos'. Es una clase que extiende JPanel. Su función principal es la de dibujar todo lo relativo al juego del comecocos, esto es el laberinto, el comecocos y los fantasmas.
Tiene una relación de asociación con ComecocosFrame y relación de dependencia con Laberinto, Comecocos y Fantasma.
Como dato miembro tiene un objeto de tipo ComecocosFrame.
Sus métodos son el método sobrescrito paintComponent() donde se encarga de dibujar lo referente al panel de juego. Se pinta el Laberinto, el Comecocos y los fantasmas, para ello se llama a funciones que lo pintan, y así el código es más funcional.
Las funciones a las que llama son: dibujaRejilla, dibujaCoco y dibujaFantasma.
Además, en esta clase se registran los eventos de teclado y de ratón, para establecer el movimiento de PacMan.

ComecocosFrame

Es la clase más importante de nuestro programa, extiende a JFrame y contiene el main que ejecuta todo lo necesario para que se ejecute el programa y procede a crear el Frame donde se sitúan todos los elementos, el panel del comecocos, los menús del programa, los botones y los cuadros de texto.
Tiene una relación de asociación con todas las clases anteriormente descritas.
Los datos miembro son un objeto Laberinto, un objeto Comecocos y 4 objetos Fantasma.
Tiene métodos propios de la clase JFrame, para iniciar los componentes. Tiene métodos para inicializar el juego, controlar los menús del programa, y métodos para definir los distintos componentes que aparecen por pantalla.

Mejoras Implementadas

Incluir puntos en el tablero

Hemos incluido en el tablero que se vean las puntos pequeños y grandes. Para pintarlos, hemos usado la misma técnica para las demás opciones del laberinto, en el método paintRejilla de ComecocosPanel. Para sumar puntos cada vez que se pasa por uno de los puntos, a cada movimiento del Comecocos se llama a actualizaMapa() para actualizar el mapa y sumar los puntos.
Los puntos son visibles en el Frame, en el apartado que pone puntos.
Cada vez que se detecta que el Comecocos ha pasado por un ficha, se comprueba con el método de Laberinto ‘finNivel()’, si es afirmativo, pasaría al siguiente nivel.

Laberinto de forma detallada

Hemos dibujado el laberinto tal y como se describe en el guión. Usando numerosos tipos de muros, para que se vea el mapa con más detalle.

Pausar el juego

En la interfaz hay un botón de pausa, que si se pulsa se para o reanuda el juego dependiendo del estado previo. Además, también se puede pulsar la barra espaciadora para pausar o reanudar el juego.
Cada vez que se pulsa el botón de pausa o espacio, se llama al método reanudar o suspender de la hebra que se está ejecutando.

Diferentes aspectos para el comecocos

Dependiendo de la dirección del Comecocos tendrá un aspecto u otro. La boca se orienta hacia su dirección. Para hacerlo, hay 4 casos distintos a al hora de dibujar a Pacman cuando se llama al método dibujaCoco().

Fantasmas

Hemos incluido 4 fantasmas en el juego como se nos indica, al principio están encerrados y van saliendo poco a poco conforme avanza la partida. Si un fantasma alcanza al comecocos éste muere y se consume una vida, con un máximo de tres. Cada vez que se consume una vida se comprueba si quedan, en caso negativo se acaba el juego.
Cada vez que se come un punto grande, se pasa al estado comestible durante un periodo de tiempo donde Pacman puede comer a los fantasmas, cuantos más fantasmas come mayor es la recompensa: 200(1 fantasma)->400(2 fantasmas) -> 800(3 fantasmas) -> 1600(4 fantasmas). Para conseguir el estado comestible, hay una variable booleana que pasa a true cuando se come un punto grande, y tras un tiempo, que se establece con un contador que disminuye una vez cada instante, vuelve al estado normal de false.
Para el movimiento de los fantasmas:

- Método getDireccionFantasmaChase: donde en cada instante se calcula cual es la ruta más corta hacia Pacman. En el caso en el que pasan a estado comestible, se escoge la ruta más lejana.
- Método getDireccionFantasmaRandom: donde el movimiento es aleatorio, se utiliza un objeto Random para generar las direcciones.

Pasar de nivel

Cada vez que se comen todos los puntos de un mapa, se aumenta nivel, ésto afecta a la función actualizaRetardo(nivel), donde se establece el tiempo que la hebra se queda dormida. Cuanto menor sea el tiempo que se queda dormida, los personajes se moverán más rápido, siendo la dificultad cada vez mayor.

Abrir partida

En el menú, si se le da a la opción New, se crea una nueva partida, para crearla nueva, se crea una nueva hebra, se restablece el Laberinto a su posición original, y los fantasmas y Comecocos vuelven a la situación desde donde se inician.

Menú de ayuda

Hemos incluido un menú de ayuda, en él salen las instrucciones de juego. Estas instrucciones están en el fichero ayuda.txt en la carpeta src. Para consultarla desde dentro del juego, se puede ir al Menú Game -> Ayuda, y sale una ventana desplegable mostrando el contenido del archivo ayuda.txt.

Records

Cada vez que se juega una partida sale un mensaje por la salida estándar donde se comunica la puntuación obtenida, además, se registra la puntuación en el archivo record.txt, donde están todos los récords hechos hasta el momento. Para consultarlos, podemos darle en el menú Game -> Record, donde sale una ventana desplegable con todas las partidas hasta ahora.

Mostrar el tiempo

En la interfaz nos aparece el tiempo transcurrido desde que se inició el juego, da igual si se pausó el juego o no. Para hacerlo, la clase Mueve se encarga de llevar la cuenta del tiempo, se estableció una variable que marca el momento de inicio, y en cada momento se consulta la diferencia de tiempo con el momento actual, dando así el tiempo de juego.

Tiempo para un nivel

En la interfaz podemos observar que se nos indica el tiempo restante para poder pasar el nivel, si ese tiempo llega a 0, se acaba la ronda y se consume una vida. Para hacerlo, nuevamente se encarga la clase Mueve de llevar la cuenta del tiempo restante. Primero se establece un tiempo por nivel. Luego en cada momento, se guarda en una variable el momento en el que tiempo restante varió, en cada ejecución se tiene en cuenta si ha cambiado el momento, si es afirmativo el tiempo restante decrementa en uno.

Sumar puntos si sobra tiempo

Cada vez que se supera un nivel, se comprueba cual es el tiempo restante y se le añade este tiempo multiplicado por 10 a la puntuación total. De nuevo, es la clase Mueve() quien se encarga de realizar esta operación.