Módulo Programación.

1º DAW.

PRÁCTICA 2: Introducción POO.
UNIDAD DE TRABAJO 4.
Profesor: Pedro Antonio Santiago Santiago.

1. Introducción.

El paradigma OO se basa en una serie de principios como son la encapsulación, la ocultación de la información o la abstracción entre otros. En esta práctica se trabaja principalmente con los principios anteriores desarrollando clases e instanciando objetos para la creación de un juego clásico Arkanoid.

2. Materiales.

Ordenador con JDK 8 instalado.

Netbeans 12.

Proyecto plantilla con Maven en GitHub.

3. Desarrollo de la práctica.

A partir del código que se facilita y en base a la práctica propuesta del juego Pong crear el juego Arkanoid.

Las características del juego son:

- La pelota ha de rebotar contra las paredes superior, derecha e izquierda cambiando el ángulo al golpear.
- De igual forma al golpear la barra ha de rebotar, el ángulo de salida depende del lugar en de la barra que golpee la pelota.

- Los ladrillos poseen una dureza, al golpear la pelota un ladrillo concreto la dureza ha de bajar un punto, en el momento que sea 0, el ladrillo ha de desaparecer. Existen ladrillos con dureza infinita, no se pueden romper.
- Existe un contador de “vidas”, inicialmente son 3, cuando llega a 0 el juego ha terminado.
- Cuando se inicia el juego la pelota ha de estar fijada a la barra en su parte superior hasta que se pulsa la barra espaciadora.
- Cada pérdida de vida, la barra y la pelota han de estar centradas en la parte inferior de la pantalla.
- Se tiene que poder definir niveles, en el nivel se tiene la velocidad inicial de la pelota, cada cuanto la pelota se acelera, el incremento de velocidad y un mapa que indique la posición de los ladrillos en el nivel.
- Al golpear con la barra, dependiendo del lugar el ángulo ha de ser diferente.
- Cuando no quedan más ladrillos que se puedan romper en un nivel se pasa al siguiente nivel.
- En caso de no quedar niveles, se ha ganado el juego.
- Se ha de conocer en todo momento el número de ladrillos pendiente de quitar, utilizándose para saber si se pasa de nivel.
- Se ha de producir un sonido cuando la pelota rebote.
- Se ha de tener sonido de fondo.
- (OPCIONAL)En ciertos ladrillos, al romperse ha de aparecer un bonus, este bonus cae por la pantalla y si colisiona con la barra modifica algunas condiciones del juego.
    - Bonus ampliar tamaño barra. Tiene un tiempo máximo
    - Bonus pelota de hierro. La pelota atraviesa los ladrillos sin rebotar y los elimina, independientemente de su dureza. Tiene un tiempo máximo
    - Bonus multiplicación de pelota. La pelota se multiplica por 3. No tiene tiempo.

3.1. Código de ejemplo.

En el código de ejemplo se divide en 2 capas el juego, la primera capa posee la lógica y el modelo y una segunda capa llamada interfaz que se encarga de pintar los diferentes elementos.

Capa de lógica y modelo, se facilitan las clases:

- Barra: Representa la barra, posee un alto, un ancho y las coordenadas con un objeto de tipo Point2D.
- Pelota. Posee un radio (aunque es un alto y un ancho), un ángulo (0 a 360), una posición (Point y una velocidad (por defecto 1). Posee el método mover que actualiza la posición en función del ángulo y la velocidad de eje x y el eje y.
- Campo: Posee las dimensiones del juego, la barra, la pelota, en el constructor se inicializa la barra y la pelota,
- Juego: Posee la lógica del juego, tiene un atributo campo y dos estáticos que indican el ancho y el alto. El método public EstadoCambiosJuego ciclo(boolean pulsados[]) recibe de la parte gráfica si las teclas para mover la barra se encuentran pulsadas para mover la barra, mueve la pelota y ha de llamara a otros métodos para implementar la lógica del juego, devuelve un enumerado con las acciones más destacadas (para poder pintar convenientemente la parte gráfica).

Se han de crear otras clases como Ladrillo y Nivel así como completar las clases anteriores.

La capa gráfica encargada de “pintar” los diferentes elementos del modelo, las clases son:

- InterfazGrafica. Posee un atributo de tipo Juego y otro de JuegoUI que se encargará de pintar la pantalla, además controla las teclas pulsadas. El método GameLoop se encarga de cada 5 mililsegundos pintar realizar un ciclo del juego y pintar la pantalla, pasándole el resultado del ciclo (para optimizar el juego). Para parar o reiniciar el juego se hace click con el ratón para pode depurar.
- CanvasLayer. Lienzo, encargado de pintar lo que se le indique, posee 2 canvas para que funcione de forma más optima, llamados fondo y principal.

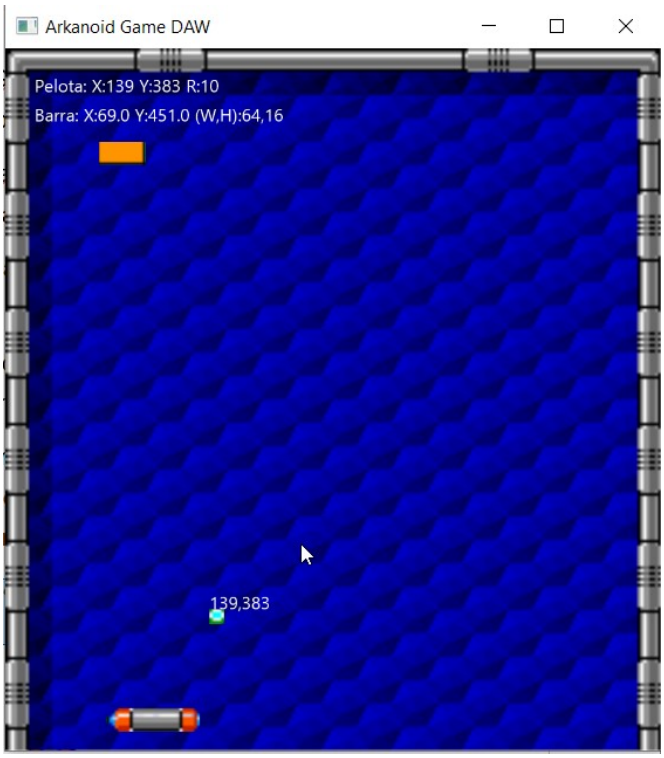
Ahora para cada clase del modelo se crea una clase en la parte gráfica encargado de pintar la parte del modelo, cada elemento tiene los ficheros multimedia.

- JuegoUI , posee un atributo de tipo Juego y las imágenes necesarias para pintar, también un atributo de BarraUI (posee una animación) y los métodos:
  - public void paint(GraphicsContext principal, GraphicsContext fondo, EstadoCambiosJuego estado): Pinta el principal y dependiendo del estado el fondo para optimizar.
  - private void paintPrincipal(GraphicsContext gc): Borra el canvas principal y pinta la pelota directamente y la barra llamando al método pintar de BarraUI.
  - public void paintDebug(GraphicsContext gc) : En el caso de estar en modo depuración (atributo debug a true) muestra información en la pantalla.
- BarraUI. Encargada de pintar la barra, con una animación, se le pasa la imagen y la barra del modelo.

Se recomienda para cada clase creada en el modelo definir otra en la parte gráfica, por ejemplo el nivel para tener el código más compacto.

Para pintar un ladrillo o cualquier otra imagen es necesario desplazarse por la imagen cargada indicando la posicion x e y origen, el ancho y alto destino, la posición x e y a pintar en el canvas y el ancho y alto en el canvas, por ejemplo para pintar un ladrillo:

```
gc.drawImage(this.imagen_bloques,  this.colorladrillos[1][0][0], this.colorladrillos[1][0][1], 16, 8, 64,64, 32, 16);
```



3.2. Recomendaciones.

1. Empezar colocando límites a la barra y hacer que la pelota rebote.
2. Definir las clases que representen los ladrillos y los niveles.
3. Pintar los ladrillos.
4. Hacer que la pelota rebote contra los ladrillos.
5. Cambiar el valor de dureza y color de los ladrillos al golpear.
6. Gestionar pasar de nivel.
7. Añadir la gestión de vidas.

3.3. Temas transversales.

Durante el desarrollo de las clases, en oficinas y en casa muchos de los equipos se colocan en standby por comodidad, ya que se facilita su puesta en marcha. Esta comodidad tiene su coste, y no es menor, tanto económico como especialmente ambiental.

A partir de la calculadora proporcionada por la asociación de consumidores, https://www.ocu.org/vivienda-y-energia/nc/calculadora/consumo-en-stand-by:

Aparatos de ofimática

		Potencia en Stand-by	Consumo anual	Gasto anual	CO2 producido
☑	Ordenador ⓘ	5	44	7,45	28,5
☑	Portátil ⓘ	4	35	5,96	22,8
☑	Monitor CRT ⓘ	3	26	4,47	17,1
☑	Monitor LCD ⓘ	1	9	1,49	5,7
☑	Router	8	70	11,91	45,6
☑	Impresora	8	70	11,91	45,6
☑	Altavoces PC	3	26	4,47	17,1

Aparatos de cocina

1. Calcular el coste para el centro de las aulas del ciclo formativo y lo que es más importante el CO2 producido.
2. Calcular el consumo y CO2 producido por los aparatos que se encuentran en "standby" en casa.
3. Comentar con los compañeros alguna posible solución para este problema.

4. Entrega.

La práctica se entrega en formato ZIP con 2 ficheros (código y presentación en formato PDF, en el campus virtual ww.aules.edu.gva.es (pendiente de matricula e inicio de curso). Se fijará la fecha en AULES.

El fichero 1 será el código generado comprimido a su vez también en zip, **importante comentar el código. Cada una de las partes en un proyecto Maven.**

El fichero 2 contiene la presentación de la práctica, que tendrá al menos los siguientes apartados. .

1. Índice.
2. Introducción.
3. Justificación de clases y paquetes creados.
4. Detalles más importantes de implementación de cada clase, por ejemplo.
    - Inicialización del juego.
    - Algoritmo de colisión.
    - Gestión de ángulos.
    - Implementación de ladrillos y niveles.
    - Gestión de imágenes.
    - Declaración de atributos y métodos estáticos.
    - Estructura del proyecto, paquetes….
5. Conclusiones.

5. Evaluación.

Unos días después de la entrega se realizá se realizará una presentación de 10 minutos por los alumnos en que se explicará apoyándose en la presentación presentada la realización de la práctica. Además se realizará una demostración de la funcionalidad para el resto de la clase.

Finalizada la presentación los componentes del grupo tendrán que responder a preguntas del profesor y/o resto de compañeros

- CE2a. Se han identificado los fundamentos de la programación orientada a objetos.
- CE2b. Se han escrito programas simples.
- CE2c. Se han instanciado objetos a partir de clases predefinidas 1
- CE2d. Se han utilizado métodos y propiedades de los objetos.
- CE2e. Se han escrito llamadas a métodos estáticos.

- CE2f. Se han utilizado parámetros en la llamada a métodos.
- CE2g. Se han incorporado y utilizado librerías de objetos.
- CE2h. Se han utilizado constructores.
- CE2i. Se ha utilizado el entorno integrado de desarrollo en la creación y compilación de programas simples.
- CE4a. Se ha reconocido la sintaxis, estructura y componentes típicos de una clase.
- CE4b. Se han definido clases.
- CE4c. Se han definido propiedades y métodos.
- CE4d. Se han creado constructores.
- CE4e.Se han desarrollado programas que instancien y utilicen objetos delas clases creadas anteriormente.
- CE4f. Se han utilizado mecanismos para controlar la visibilidad de las clases y de sus miembros.
- CE4h. Se han creado y utilizado métodos estáticos.
- CE4j. Se han creado y utilizado conjuntos y librerías de clases.