Primera convocatoria
Programación Orientada a Objetos
- Mayo 2022 -

2h30'

JManzanas

NORMAS DEL EXAMEN

1. Descargar el enunciado y los códigos suministrados.
2. Preparar el entorno del examen:
    a. Consola.
    b. Explorador de archivos.
    c. Editor de texto.
    d. Ventana del navegador con la doc de la API:
        i. https://docs.oracle.com/en/java/javase/16/docs/api/index.html
3. Comenzar la grabación con OBS: NO empezar el examen hasta no estar seguros de que la grabación se ha iniciado.
4. Abrir el editor de texto.
    a. Escribir la palabra POO.
    b. Copiarla.
    c. Pegarla dos veces.
5. Pulsar la combinación de teclas Windows + Tab.
6. Comenzar el examen.

Desarrollar un juego en el lenguaje de programación Java que siga la lógica que se describe a continuación y cuyo JAR se suministra en Moodle junto a este enunciado para que se pueda comprobar su funcionamiento.

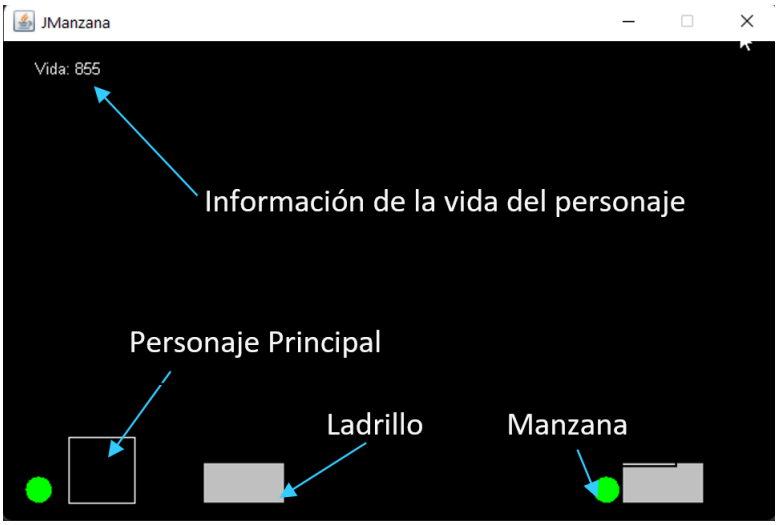

Fig 1. Ventana del juego y sus objetos

Descripción de los objetos del juego

El juego se desarrolla sobre una posición fija en el eje Y, moviéndose los objetos (manzanas y ladrillos) por el eje X sobre el ancho de la ventana, de derecha a izquierda (Fig 1).

En el juego existe un **personaje principal** (cuadrado blanco sin relleno) que solo puede tener dos posiciones:

1. Su posición de inicio: x=50 y sobre la Y base que tome de referencia el alumno.
2. Una posición superior de -100 pixels desde su posición inicial en el eje Y, llevando a cabo un salto. Se llegará a esta posición de salto al pulsar la barra espaciadora (SPACE). Cuando se libere la barra espaciadora, volver a su posición base de Y.

IMPORTANTE: en el salto no existirá transición de subida ni de bajada (no será un movimiento continuo). Pasará de una forma discreta directamente de Y a Y-100 y viceversa.

Como se describió previamente, los **objetos del juego** (ladrillos y manzanas) se moverán horizontalmente solo sobre el eje X, de derecha a izquierda, de forma automática (sin responder a ninguna tecla) a una velocidad dada. Cada objeto se moverá una velocidad y la forma de asignarse será mediante la invocación a la clase que se suministra: **Generador**. Esta clase devolverá velocidades de forma aleatoria válidas (un número entero en el rango 1 a 4). Este número determinará el número de pixels que se desplazará cada objeto cada vez que el juego se lo solicite de una forma automática e iterativa. Una velocidad de 4 significará que tendremos que mover el objeto 4 pixels en el eje X de derecha a izquierda.

La posición X inicial de los objetos (para que todos no estén juntos y no sea determinista) se calculará de forma aleatoria también gracias a la clase Generador que se suministra. Nótese como habrá que pasarle un valor mínimo y máximo de X dentro del cual calculará el valor aleatorio. Recordad que todos estarán fijos en una posición de Y.

Las dimensiones por defecto de todos los objetos para el examen serán:

- Personaje principal: un cuadrado blanco sin relleno de lado 50.
- Ladrillo: un rectángulo de 30x60 LIGHT_GRAY con relleno.
- Manzana: un círculo de 20 de radio GREEN con relleno.

Aunque partamos de estos valores por defecto para esta solución y para todos los objetos de un mismo tipo, se deberá programar la solución para cualquier posibilidad de valores. Los valores del color y relleno serán siempre los mismos para siempre. Por lo tanto, el día de mañana podremos tener manzanas grandes y pequeñas, pero todas verdes y con relleno.

Objetivo del juego

El objeto del juego será mantener la vida inicial proporcionada al personaje (1000) por encima de 0. Cuando dicha vida alcance el valor 0, el juego terminará y mostrará el mensaje:

Fig 2. Estado del juego al finalizar (vida <=0)

La forma de modificar la vida del personaje será la siguiente:

- Cuando el personaje impacte con una manzana se incrementará su vida en la cantidad que le proporcione dicho objeto, por defecto, todas las manzanas tendrán un valor de 5.
- Cuando el personaje impacte con un ladrillo se decrementará su vida en la que le dañe dicho objeto, por defecto, todos los ladrillos, 20.
- Si lo dejáramos en este punto, el personaje podría estar siempre saltando (manteniendo pulsada siempre el SPACE) y no perderíamos nunca. Por ello, para hacer atractivo el juego decidiremos que saltar cansa, es decir, consume vida. La pérdida de vida en el salto será la misma que la que proporciona una manzana, 5.

Se podría pensar que la estrategia para mantener la misma vida inicial siempre (vida inicial = 1000) sería no impactar con algún ladrillo y comer tantas manzanas como saltos se hagan.

Diseño del juego

Además de la clase Generador, también se proporcionan la clase Juego, la cual está completamente desarrollada y no hay que tocar nada y la clase Tablero que representa el soporte (lienzo) donde se creará el juego. Esta clase habrá que completarla para alcanzar la funcionalidad deseada.

La clase Tablero posee un atributo llamado *fps* (frames por segundo) que nos permite calcular el tiempo que dejaremos pasar (delay) hasta volver a actualizar todos los personajes del juego con su nueva posición calculada. La lógica general de actualización de los movimientos de los objetos del juego (solo las manzanas y los ladrillos ya que el personaje se mueve por eventos de teclado) será la siguiente:

```
mientras(personaEsteViva)
{
        esperar(delay)
        moverTodosObjetosASuNuevaPosicion()
        checkPersonajePrincipalImpactaConObjetos()
        repintaTablero()
}
```

Cálculo de impacto

Este pseudocódigo muestra cómo después de mover los objetos se comprobará si los ladrillos o manzanas han impactado con el personaje. La forma de saber si un objeto impacta con el personaje será mediante la intersección de los rectángulos que delimitan el área que ocupan los objetos. Una forma sencilla de resolver esta funcionalidad es gracias a la clase java.awt.Rectangle y su método intersect.

Para simplificar la implementación del juego, los objetos no desaparecerán ni se eliminarán después de impactar con el personaje, como sería de esperar. Esto significa que, al atravesar una manzana al personaje, le aportará 5 de vida mientras dure la intersección. Fijándonos en el bucle del pseudocódigo, si el tiempo que tarda en atravesar la manzana al personaje son 1000ms (1seg), por ejemplo, y hay un delay de 100ms, se repetirá el impacto 10 veces, aportando 50 de vida. Esto nos lleva a pensar que una manzana lenta es más saludable que un rápida. 😊

El desplazamiento de los objetos se produce cada cierto tiempo (delay) desplazándose unos determinados pixels (velocidad).

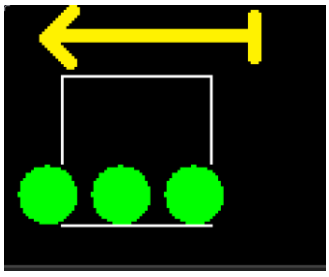

Fig 3. Repetición de impactos de una misma manzana

Inicialización del juego

El programa leerá de un fichero (data/configuración.txt) la información de entrada del juego, que será:

- El número de manzanas en el juego.
- El número de ladrillos.
- El valor de los fps por defecto del juego.

```
manzanas: 3
ladrillos: 2
fps: 60
```

Fig 4. Contenido del fichero

A continuación, se muestra un ejemplo de ejecución al modificar los valores del fichero a 30 manzanas y 1 ladrillo, por ejemplo.

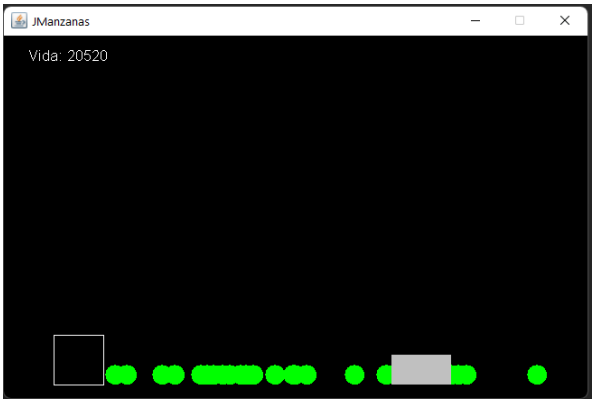

Fig 4. Contenido del fichero

Recomendaciones de implementación

Se recomienda realizar una implementación iterativa del juego, por partes que 100%% funcionales.
Iteraciones a realizar:

1. Estructura de las clases necesarias de la lógica de objeto.
2. Personaje con funcionalidad de salto.
3. Desplazamiento de objetos.
4. Detección de impactos.
5. Cálculo de la vida.
6. Finalizar el juego

Se concederá un punto adicional (pudiendo llegar a una nota de 11) si se implementa la siguiente funcionalidad habiendo realizada toda la funcionalidad anterior: mostrar un listado de los 5 mejores resultados de tiempo durante toda la vida del juego y una opción de volver a jugar.

Herramientas que se podrán utilizar en el examen:

- Editor de texto.
- Compilador.
- JShell: si alguien desea ejecutar código Java de una forma rápida.
- No se podrá consultar ningún sitio web.
- Solo se podrá consultar la API que se encuentra en la siguiente URL: https://docs.oracle.com/en/java/javase/16/docs/api/index.html
- No se podrá utilizar ningún IDE (Visual Studio Code, IntelliJ, Eclipse, etc.)

Se pide:

- Implementar todas las clases para dar respuesta al enunciado cumpliendo con los principios de orientación a objetos vistos en la asignatura.
- Seguir la estructura de paquetes que desee el alumno.
- Se podrá utilizar cualquier clase del JDK y estilo de programación visto en clase: funcional, por ejemplo.
- Se podrán utilizar import con *.
- El examen deberá ser grabado con OBS.