ADSW- Ejercicio 3
Juego de la serpiente

josé a. mañas / juan a. dela puente

20.3.2019

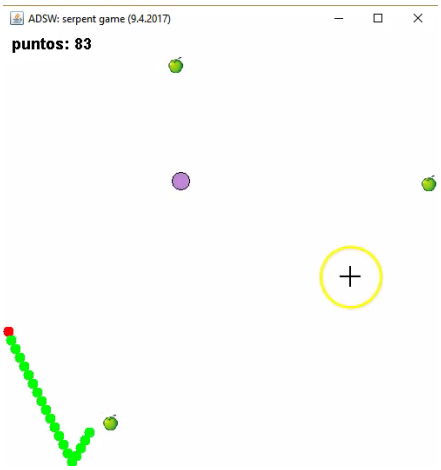
1 Objetivo

Trabajar con threads concurrentes.

Para ello recurrimos a un juego con interfaz gráfica donde los errores “se ven”.

2 El juego

Tenemos una serpiente que maneja el jugador. Tenemos bolas que caen del cielo, a una velocidad fija, rebotando en las paredes laterales. Tenemos manzanas que no se mueven.



Hay una serpiente. En un momento dado puede haber entre 0 y un número indeterminado de bolas cayendo y entre 0 y un número indeterminado de manzanas en el terreno de juego.

La serpiente avanza a base de toques del jugador que le va indicando la dirección de avanza. Cuando llega a las paredes, “rebota”. Mientras avanza va perdiendo puntos.

Cuando la serpiente se come una manzana, la manzana desaparece y la serpiente se alarga. También suma puntos.

Cuando una bola golpea una manzana, desaparecen la manzana y la bola.

Cuando una bola golpea la cabeza de la serpiente, la serpiente queda parada.

Cuando una bola golpea el cuerpo de la serpiente, la rompe y la serpiente sigue viva con lo que queda de cuerpo. También pierde puntos.

3 Descripción de las clases java

Por orden alfabético.

class Apple implements Screen.Thing

> Las manzanitas del juego. Carga una imagen de un fichero y luego la pinta cuando se le llama a paint() desde Screen. Cuando una bola choca con una manzana, desaparecen ambas.
>
> La interface Screen.Thing requiere un método paint().

class AppleGenerator

Una tarea para ir generando manzanitas de vez en cuando en sitios aleatorios.

class AppleListMonitor (a desarrollar por el alumno)

Contiene una estructura de datos a compartir, que es la lista de manzanitas creadas y no destruidas.

class Ball implements Runnable, Screen.Thing

Las bolas que van cayendo. Son threads. Nacen en el borde superior y van bajando hasta desaparecer por abajo o cuando chocan con una manzana. Rebotan en las paredes laterales.

La interface Screen.Thing requiere un método paint().

class BallGenerator

Una tarea para ir generando bolas de vez en cuando en sitios aleatorios. También es aleatoria la velocidad y la inclinación.

class Game

Funciona como un almacén de objetos que son compartidos por los diferentes componentes del juego.

Lleva cuenta de todos los elementos que son únicos en el juego: la pantalla (screen), la puntuación, el monitor de manzanas (AppleListMonitor) y la serpiente.

También lleva cuenta del estado del juego: RUNNING, FINISHED, TESTING.

class Nap

Métodos útiles para dormir las threads olvidándonos de las interrupciones.

class Play

Arranca el juego en modo normal.

class Puntuacion implements Screen.Thing, Runnable (a desarrollar por el alumno)

Lleva cuenta del número de puntos en cada momento. Y se pinta en pantalla. Si la cuenta baja a 0, el juego termina.

class RW_Monitor_0

Versión simplificada de RW_Monitor: tiene los métodos; pero no controla nada.

class RW_Monitor extends RW_Monitor_0 (a desarrollar por el alumno)

Es un monitor que controla lecturas y escrituras de forma exclusiva: pueden darse lecturas simultáneas, pero para escribir solo puede haber 1 thread activa.

class Screen

Ventana para ir pintando lo que ocurre en el juego.

Proporciona una interface Thing que requiere clases que implementan un método paint() que es el que al final dibuja en la pantalla.

Para interactuar con el jugador hay 2 opciones:

- TouchPadListener1: captura toques en el touchpad para ir citando a la serpiente.
- TouchPadListener2: captura el deslizamiento del dedo para ir orientando a la serpiente. Para pantallas táctiles.

class Serpent implements Runnable, Screen.Thing

La serpiente.

La interface Screen.Thing requiere un método paint().

class StressTest1

Provoca situaciones con muchas threads y sirve para validar los controles de concurrencia.

class StressTest2

Provoca situaciones con muchas threads y sirve para validar los controles de concurrencia.

class TouchPadListener1 implements MouseListener

Para touchpad. Captura toques y dirige la serpiente hacia el punto en el que se toca.

class TouchPadListener2 implements MouseMotionListener

Captura el movimiento del dedo por una pantalla táctil. Viene desactivado, debe sacarlo de comentario para que atienda al dedo.

class XY

Puntos en 2 dimensiones. Incluye operaciones aritméticas clásicas como el teorema de Pitágoras, la proyección de un punto sobre una recta y la distancia de un punto a un segmento. Todo para saber si en su movimiento, las bolas o la serpiente pasan cerca de una manzana o de un segmento de la serpiente.

4 Lo que hay que hacer

class AppleListMonitor	el alumno debe escribirla
class Puntuación	se da sin control de concurrencia. el alumno debe completarla
class RW_Monitor	el alumno debe escribirla
class RW_MonitorTest	el alumno debe escribirla
class Screen	se da el alumno puede decidir si la serpiente se controla por clic o por deslizamiento del dedo en una pantalla táctil

class StressTest…	se facilitan algunas pruebas de carga

4.1 class RW_Monitor

Es la clase que implementa el mecanismo de lectores-escritores.

Los nombres de los métodos vienen en el javadoc.

Este monitor hay que usarlo en varios sitios para proteger las variables locales a la vez que se permite la máxima concurrencia (readers) siempre y cuando no haya que modificar los datos críticos (writers).

- Lo usa la clase Serpent, que se da hecha.
- Debe usarlo la clase AppleListMonitor que la escribe el usuario.

4.2 class RW_MonitorTest

Pruebas de que el monitor RW_Monitor está funcionando correctamente.

Por ejemplo

```
assertEquals(0, monitor.getNReadersIn());
assertEquals(0, monitor.getNWritersIn());

monitor.openReading();
assertEquals(1, monitor.getNReadersIn());
assertEquals(0, monitor.getNWritersIn());
```

Otro ejemplo:

```
@Test(expected = Exception.class)
public void ow_cr() {
    monitor.openWriting();
    monitor.closeReading();
}
```

Se espera del alumno que proporcione casos de prueba para cubrir adecuadamente toda la funcionalidad de la clase.

4.3 class AppleListMonitor

Tiene una lista de manzanas y un monitor RW.

Hay que proteger los métodos de acceso según puedan modificar dicha lista o sean simplemente consultivos.

void add(Apple)	modifica la lista
void remove(Apple)	modifica la lista
Apple getCloseApple(XY, XY)	consultivo
Apple hitCloseApple(XY, XY)	modifica la lista

Ambos métodos getCloseApple() y hitCloseApple() localizan una manzana cercana al segmento P1-P2. Pero mientras que getCloseApple() se limita a informar de que la ha encontrado, el método hitCloseApple() ejecuta la eliminación de la manzana de la lista.

La forma de usarlo es en 2 fases; primero se averigua si hay una manzana cerca de forma concurrente, sin consecuencias; solo miramos. Si hay una manzana, entonces hay que bloquear a las demás tareas mientras se elimina.

```
if (appleListMonitor.getCloseApple(prev, xy) != null) {
    if (appleListMonitor.hitCloseApple(prev, xy) != null) {
```

Por favor, tenga en cuenta que estos métodos compiten con otras threads. Es posible que el primero detecte una manzana a tiro y que para cuando ejecutamos el segundo la manzana ya no esté.

4.4 class Puntuacion

Hay que modificarla estableciendo una zona crítica de acceso exclusivo al contador de puntos.

5 Zonas críticas

El juego, con varias threads concurrentes, presenta varias estructuras de datos compartidas que hay proteger frente a intentos concurrentes de acceso por diferentes threads que pudieran corromper los datos.

Lista de manzanas

Es un campo interno, private, de AppleListMonitor. Es objeto de la carga de nuevas manzanas, de consultas de cercanía y de la eliminación de manzanas.

Se califican los métodos como de lectura (solo consultan datos) o de escritura (modifican la lista) y se protegen por medio de un monitor del tipo Readers-Writers.

Puntos

Es un campo interno de Puntuacion. Es objeto de incrementos y decrementos por parte de la propia thread Puntuacion, y por parte de la Serpent.

La protección es una zona crítica de acceso exclusivo durante las modificaciones.

El cuerpo de la serpiente (lista de bolitas)

Es un campo interno de la Serpent. La propia serpiente lo modifica al arrastrarse y cuando come una manzana y crece. Las bolas pueden fragmentarlo.

Se da hecho, con un monitor RW que controla los accesos para consulta (read) y modificación (write).

Lista de cosas a pintar en pantalla.

Es interno de Screen. Sufre accesos concurrentes durante las operaciones de pintado de la pantalla (read) y puesta y retirada de elementos en la pantalla (write).

Se da hecho, con una zona crítica de acceso exclusivo para las operaciones de acceso y modificación de la estructura.

6 Entrega

En el paquete

es.upm.dit.adsw.ej3

las clases

- AppleListMonitor
- Puntuacion
- RW_Monitor
- RW_MonitorTest

Si ha desarrollado más pruebas, adjúntelas.