Práctica 3: Interfaces, Ficheros y Excepciones

Fecha de entrega: 15 de Enero de 2015 a las 23:00

Objetivo: Implementación y uso de la herencia, interfaces, excepciones y ficheros

1. Descripción de la práctica

En la Práctica 3 vamos a integrar los dos juegos de la vida de las prácticas 1 y 2. Es decir, vamos a poder realizar, en la misma práctica, simulaciones del juego de la Práctica 1 (*juego simple*) y del juego de la Práctica 2 (*juego complejo*). Además vamos a permitir guardar la configuración del juego actual en un fichero de texto para poder después cargarla. En particular esto supone la introducción de dos nuevos comandos:

- guardar nomFich: permite almacenar en un fichero de texto, de nombre nomFich, la configuración del juego actual.
- cargar nomFich: carga como juego actual, el almacenado en el fichero de texto nomFich

Todo el tratamiento de errores asociado a los ficheros, así como al resto de métodos susceptibles de fallar, lo haremos utilizando *tratamiento de excepciones*. Finalmente y dado que tenemos los dos juegos integrados en la misma práctica, ahora desaparecen los comandos crearcelulasimple y crearcelulacompleja, y aparece un único comando crearcelula f c, que dependiendo del juego actual que se esté simulando, creará unos tipos de célula u otros. Por ejemplo, si el juego actual es el juego simple, entonces se creará una célula simple en la posición (f,c). Sin embargo si el juego actual fuera el complejo, el comando pedirá al usuario que indique el tipo de célula a crear (simple o compleja), y dependiendo de la opción tecleada por el usuario se creará un tipo de célula u otro. Por otro lado, y al tener dos juegos disponibles, permitiremos cambiar de un juego a otro a través del comando jugar, que admite dos tipos de uso:

- jugar simple N M S: inicia la simulación con una nueva configuración del juego simple, con una superficie de N filas y M columnas, y con S células simples colocadas de forma aleatoria.

- jugar complejo N M S C: Es similar al comando anterior, donde C representa el número de células complejas que debe contener el mundo complejo.

2. Ficheros

El estado de un juego vamos a almacenarlo en un fichero de texto, donde la primera línea indica si el juego es simple o complejo, las dos líneas siguientes contienen el número de filas y columnas respectivamente, y el resto de líneas hacen referencia a la información almacenada en una posición de la superficie. Por ejemplo, el siguiente fichero:

```
simple
4
3
1 1 simple 1 0
2 0 simple 1 0
2 1 simple 1 0
2 2 simple 1 0
3 0 simple 1 1
3 1 simple 1 0
```

guarda la configuración de un juego simple, con una superficie de 3 filas y 4 columnas, que contiene 6 células simples en las posiciones (1,1), (2,0), (2,1), (2,2), (3,0) y (3,1). El resto de la información de la línea almacena, respectivamente, el valor de los atributos que controlan el número de pasos dados, y el número de pasos sin movimiento. Así por ejemplo, la línea:

```
2 1 simple 1 0
```

representa una célula simple en la posición (2,1), que ha dado 1 paso y que lleva 0 pasos sin movimiento. El fichero completo representa la siguiente superficie:

```
- - -
- X -
X X X
X X -
```

Los ficheros que corresponden a configuraciones del juego complejo varían el formato para admitir el nuevo tipo de célula. Por ejemplo, el fichero de texto:

```
complejo
5
7
0 0 compleja 0
0 1 compleja 1
0 2 compleja 1
1 0 simple 1 1
1 1 simple 1 0
2 2 compleja 0
3 0 simple 1 0
```

contiene una superficie de 5 filas y 7 columnas, 4 células complejas y tres células simples. Cada fila puede representar o una célula simple, donde la información se almacena de forma similar a como hemos comentado anteriormente, o una celula compleja de la forma X Y

compleja N, donde (X,Y) es la posición de la célula, y N es el número de veces que la célula compleja ha comido.

3. Tratamiento de excepciones

En esta práctica se controlarán los errores a través de excepciones. Por un lado aparecen excepciones asociadas al tratamiento de los ficheros, como son FileNotFoundException o IOException. Dichas excepciones deben tratarse de forma correcta y en caso de que se produzcan el fichero debe quedarse estable y la aplicación funcionando. Además de estas excepciones, pueden aparecer, entre otras, las siguientes:

- FormatoNumericoIncorrecto: se produce cuando se espera un número pero se teclean caracteres no numéricos. Por ejemplo en el comando CREARCELULA pp pp se lanzaría una excepción de este tipo.
- PalabraIncorrecta: cuando leemos de un fichero y encontramos palabras que no se corresponden con el formato estipulado para el fichero. Por ejemplo si la primera línea del fichero no es simple o complejo, entonces debe saltar una excepción, y dejar el juego estable.
- IndicesFueraDeRango: cuando intentamos acceder a posiciones de la superficie que no existen.
- ErrorDeInicializacion: si se intenta jugar a un juego donde la superficie no tiene capacidad para almacenar el número de células introducidas por el usuario.

Puedes añadir más tipos de excepciones para refinar el tipo de mensajes de error lanzados por la aplicación.

4. Clases que hay que implementar

Para implementar la Práctica 3 necesitaremos todas las clases de la Práctica 2, con algunas modificaciones para soportar simultáneamente ambos juegos.

- Controlador: incorpora un nuevo atributo boolean simulacionTerminada que controla el fin de la simulación. El método realizaSimulacion() es similar al de la Práctica 2, pero incorpora tratamiento de excepciones. Por otro lado ahora, puesto que tenemos dos mundos (el simple y el complejo) y podemos cambiar de uno a otro a través del comando jugar, los comandos se ejecutarán sobre el controlador, y no sobre el mundo como ocurría en la Práctica 2. Por lo tanto las llamadas al método ejecuta de los comandos se realizará pasando como parámetro al controlador, es decir comando.ejecuta(this). Además como ahora los comandos interactuan con el controlador, dentro de esta clase necesitarás métodos para manipularlos. Por ejemplo, el método ejecuta(Controlador controlador) implementado en el comando paso realizará una llamada a controlador.daUnPaso(), y el método daUnPaso en controlador contendrá la instrucción mundo.evoluciona(). El mismo proceso debes realizarlo con todos los comandos.

  Con respecto a los ficheros, el controlador contendrá el método void cargar(String nombreFichero), que utilizará la primera línea del mismo para crear un mundo simple o complejo por defecto, y llamar a continuación al método cargar del mundo.

- Comando: es una interfaz similar a la Práctica 2, salvo por el método ejecuta, que recibe como parámetro un controlador. Esta interfaz es implementada por cada uno de los comandos disponibles. Para el caso del comando Jugar, la clase contendrá un atributo privado Mundo mundo, que se inicializará en el método parsea dependiendo de los parámetros del comando. Por ejemplo si el usuario teclea jugar simple 3 4 2, el método parsea devolverá new MundoSimple(3,4,2). De forma similar, el método ejecuta de la clase Jugar contendrá una llamada a controlador.juega(this.mundo). Los comandos Cargar y Guardar tendrán un atributo privado String nombreFichero para almacenar el nombre del fichero.

- Mundo: es una clase abstracta que contiene, igual que en la Práctica 2, un atributo del tipo Superficie. Ahora además contendrá dos atributos para almacenar el número de filas y columnas del juego. La constructora por defecto de Mundo inicializa el número de filas y columnas a 0, y la superficie a null. Por el contrario, la constructora con argumentos Mundo(int filas, int columnas) inicializará los atributos para filas y columnas de acuerdo a los parámetros de entrada, y creará una superficie de ese tamaño, llamando a continuación a un método abstracto inicializaMundo() que tendrán que implementar las clases que heredan de Mundo. El método evoluciona() de esta clase es igual al de la Práctica 2, salvo el tratamiento de excepciones. Además la clase contendrá métodos para guardar y cargar un mundo en un fichero de texto.

  De la clase Mundo heredan las clases MundoSimple y MundoComplejo, que no son abstractas y por lo tanto implementan, entre otros, el método inicializaMundo(). Observa también que el método para guardar el mundo en un fichero, necesita escribir si se está en presencia de un mundo simple o complejo, y por lo tanto esta información sólo puede escribirse en las clase concretas.

- Superficie: similar a la Práctica 2, salvo por el tratamiento de excepciones y los métodos para guardar y cargar una superficie.

- Celula: ahora pasa a ser una interfaz, que además de los métodos que contenía en la Práctica 2, añade dos nuevos métodos abstractos para guardar y cargar en un fichero. Esta interfaz es implementada por las clases CelulaSimple y CelulaCompleja.

5. Ejemplos de ejecución

A continuación mostramos un ejemplo de ejecución sencillo, donde se muestran los comandos nuevosjugar, guardar y cargar.

```
 -  -  -  -  -  X
 -  -  -  -  -  -
 X  X  X  -  -  X
 -  -  X  -  -  -
 X  -  -  -  -  -

Comando > jugar complejo 5 7 3 5
 -  -  -  -  -  -  -
 -  X  X  -  X  *  -
 -  -  -  -  *  -  -
 *  -  -  -  -  -  -
 -  *  -  -  *  -  -
```

```
Comando > guardar complejo.txt
 -  -  -  -  -  -  -
 -  X  X  -  X  *  -
 -  -  -  -  *  -  -
 *  -  -  -  -  -  -
 -  *  -  -  *  -  -


Comando > jugar simple 3 2 1
 -  X
 -  -
 -  -

Comando > vaciar
 -  -
 -  -
 -  -

Comando > cargar complejo.txt
 -  -  -  -  -  -  -
 -  X  X  -  X  *  -
 -  -  -  -  *  -  -
 *  -  -  -  -  -  -
 -  *  -  -  *  -  -


Comando > crearcelula 3 3
De que tipo: Compleja (1) o Simple (2): 1

Creamos nueva celula compleja en la posición: (3,3)
 -  -  -  -  -  -  -
 -  X  X  -  X  *  -
 -  -  -  -  *  -  -
 *  -  -  *  -  -  -
 -  *  -  -  *  -  -

Comando > cargar simple.txt
EXCEPCION: Fichero no encontrado
 -  -  -  -  -  -  -
 -  X  X  -  X  *  -
 -  -  -  -  *  -  -
 *  -  -  *  -  -  -
 -  *  -  -  *  -  -

Comando > vaciar
 -  -  -  -  -  -  -
 -  -  -  -  -  -  -
 -  -  -  -  -  -  -
```

```
 -  -  -  -  -  -  -
 -  -  -  -  -  -  -

Comando > cargar complejo.txt
 -  -  -  -  -  -  -
 -  X  X  -  X  *  -
 -  -  -  -  *  -  -
 *  -  -  -  -  -  -
 -  *  -  -  *  -  -


Comando > salir
```

6. Entrega de la práctica

La práctica debe entregarse utilizando el mecanismo de entregas del campus virtual, no más tarde de la fecha y hora indicada en la cabecera de la práctica.

El fichero debe tener al menos el siguiente contenido:

- Directorio src con el código de todas las clases de la práctica.
- Fichero alumnos.txt donde se indicará el nombre de los componentes del grupo.
- Directorio doc con la documentación generada automáticamente sobre la práctica.