Práctica 3: Interfaces, Ficheros y Excepciones

Fecha de entrega: 3 de Marzo, 16:00

Material proporcionado:

Fichero	Explicación
`TestPr3.zip`	Proyecto de tests para la práctica.
`validador.zip`	Programa validador de la entrega.
`documentacion.zip`	Documentación de ficheros y clases necesarias para la implementación.

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

1. Nueva funcionalidad

Vamos a extender la práctica anterior añadiendo dos nuevas instrucciones que permiten:

- Comando DROP. Que WALL·E suelte un objeto de los que lleva en su inventario y lo deposite en el lugar en el que se encuentre en ese momento. No se podrá dejar un objeto en un lugar si en él ya existe otro objeto que se llame igual.
- Comando RADAR. Mostrar la información de los objetos que estén disponibles en el lugar donde se encuentra WALL·E. Este comando es distinto de SCAN, que se centraba en objetos ya recogidos por WALL·E.

Por otro lado se permitirá que la información de la ciudad (lugares, calles, items, etc.) se cargue a través de un fichero de texto. El fichero concreto se seleccionará a través de los parámetros de la aplicación, según queda descrito más abajo.

Para la implementación de esta práctica necesitarás realizar una profunda refactorización para mejorar la extensibilidad y claridad de código y para hacer uso de excepciones.

2. Formato del fichero donde se almacena una ciudad

El formato de los ficheros de texto donde se almacena la información de una ciudad es el siguiente:

```
BeginCity
BeginPlaces
place 0 Sol You_are_at_the_center_of_Madrid noSpaceShip
place 1 Callao In_this_square_you_can_find_a_code_card noSpaceShip
place 2 Colon People_concentrates_here_to_watch_football noSpaceShip
place 3 Exit Ok,_finally_you_have_found_your_spaceship... spaceShip
EndPlaces
BeginStreets
street 0 place 0 south place 1 open
street 1 place 1 east place 2 open
street 2 place 2 north place 3 closed onetwothreefourfive
EndStreets
BeginItems
fuel 0 Petrol from_olds_heatings 10 3 place 0
fuel 1 Battery to_get_cracking -50 1 place 0
codecard 2 Card The_key_is_too_easy onetwothreefourfive place 1
garbage 3 Newspapers News_on_sport 30 place 2
EndItems
EndCity
```

Observa que el fichero comienza siempre con una línea que contiene la palabra `BeginCity` y finaliza con `EndCity`. Dentro del fichero se distinguen tres partes: La descripción de los *lugares*, la descripción de las diferentes *calles* y finalmente los *objetos* que forman parte del juego. Para delimitar donde empiezan y terminan estas descripciones se utiliza `BeginX` y `EndX`, donde `X` $\in$ `{Places,Streets,Items}`.

Fuera de estas palabras reservadas, cada línea empieza por una palabra de las siguientes `{place,street,fuel,codecard,garbage}` . Después de cada palabra viene un número. Este número es importante, pues actúa como identificador que puede ser utilizado en otras partes del fichero para referirse a ese `place` o `street`. Para que la carga sea fácil, el formato del fichero *exige* que para cada tipo de elemento vayan consecutivos y el primero sea el `0`, lo que se traduce en que estarán siempre en el rango `{0 .. N-1}`, donde N es el número de elementos de ese tipo que existen en el fichero y no podrá haber dos iguales.

La descripción de un lugar es la siguiente:

1. Comienza con la palabra `place`.

2. A continuación viene el identificador numérico del lugar. Como con el resto de identificadores numéricos del fichero debe comprobarse que el primero es 0 y los siguientes van siendo incrementados en la unidad.

3. La siguiente palabra es el identificador alfanumérico del lugar.

4. A continuación aparece la descripción del lugar. En principio se puede asumir que esta descripción no contiene espacios en blanco sino que éstos han sido sustituidos por el carácter "_". Cuando se cargue la ciudad, estos caracteres se sustituirán por espacios en blanco para que las cadenas se visualizen correctamente.

5. Finalmente aparecen las palabras `spaceShip` o `noSpaceShip` para indicar si el lugar es la nave espacial.

La descripción de una calle comienza con el identificador `street` seguido de:

1. El identificador numérico de calle. Las calles empiezan a enumerarse obligatoriamente desde 0 por las razones antes expuestas.

2. A continuación aparece la palabra `place` seguida de un identificador numérico que indica el lugar de origen de la calle. Este lugar de origen debe coincidir con alguno de los identificadores numéricos listados en la sección `BeginPlaces`.

3. Posteriormente aparece la dirección, que puede ser `north,south,east,west`.

4. Luego viene la palabra `place`, otra vez, seguida de otro dentificador numérico que representa el lugar de destino de la calle. Este lugar de destino debe coincidir con alguno de los identificadores numéricos listados en la sección `BeginPlaces`.

5. Finalmente aparecerá `open` o `closed`. Si aparece `open` la descripción termina. Si aparece `close` a continuación aparecerá la clave necesaria para abrir la calle.

Finalmente aparece la colección de objetos/items que componen el juego. Cada línea debe comenzar indicando el tipo de objeto que se va a describir. Los identificadores posibles serán `fuel`, `garbage` o `codecard`. Después aparece un identificador numérico, que al igual que en los casos anteriores, debe comenzar en 0 e irse incrementando. A continuación aparece el nombre concreto del objeto seguido por su descripción. Más tarde, dependiendo del tipo de objeto, aparecerá la siguiente información:

- En el caso de *fuel*, los dos números siguientes indican, respectivamente, la energía que aporta y el número de veces que se puede utilizar.

- Para el caso de *garbage*, aparece la cantidad de materia reciclada que aporta cuando se usa.

- Para un *codecard* aparece la clave que abre.

Todos los objetos terminan con la palabra `place` seguido de un identificador numérico correspondiente al lugar donde se encuentra este objeto. El identificador debe coincidir con alguno de los proporcionados en la sección `BeginPlaces`.

Ten en cuenta que no es responsabilidad del cargador del fichero comprobar que la ciudad descrita sea correcta desde un punto de vista lógico, es decir si tiene algún lugar marcado como nave espacial, si se puede llegar a ella, si todos los lugares son accesibles, etc.

Sí que debe comprobar que se satisface el formato tal y como se ha presentado en este enunciado. Como parte opcional se valorará que la aplicación permita, además, ficheros donde las descripciones tengan espacios. En este caso, para distinguir cuándo comienza y termina una descripción, utilizaremos comillas dobles. Por ejemplo, la descripción del primer lugar del ejemplo anterior sería:

```
place 0 Sol "You are at the center of Madrid" noSpaceShip
```

3. Ejemplo de Ejecución

Cuando se ejecuta la aplicación sin parámetros, ésta presenta por la salida de error (`System.err`) un mensaje explicando cómo debe utilizarse. Además, la aplicación terminará con el código de error `1` (usando `System.exit(1)`).

```
$> java tp.pr3.Main
Bad params.
Usage: java tp.pr3.Main <mapfile>

<mapfile> : file with the description of the city.
```

Algo parecido ocurre si se ejecuta utilizando un fichero que no existe, o que no puede leerse (por tener un formato incorrecto). En este caso, el código devuelto será `2`:

```
$> java tp.pr3.Main noExiste.txt
Error reading the map file: noExiste.txt (No existe el fichero o
el directorio)
```

Por último, si se especifica un fichero correcto, la aplicación se ejecutará con normalidad (sin escribir nada por `System.err` y saliendo con código `0`[1]). El siguiente ejemplo de ejecución utiliza el fichero de texto descrito en la sección anterior:

```
$> java tp.pr3.Main madrid.txt
Sol
You are at the center of Madrid
The place contains these objects:
   Battery
   Petrol
WALL·E is looking at direction NORTH
      * My power is 100
      * My reclycled material is 0
WALL·E> pick Petrol
WALL·E says: I am happy! Now I have Petrol
WALL·E> pick Battery
WALL·E says: I am happy! Now I have Battery
WALL·E> turn left
WALL·E is looking at direction WEST
      * My power is 95
      * My reclycled material is 0
WALL·E> turn left
WALL·E is looking at direction SOUTH
      * My power is 90
      * My reclycled material is 0
WALL·E> move
WALL·E says: Moving in direction SOUTH
Callao
In this square you can find a code card
The place contains these objects:
   Card
```

[1] Las aplicaciones devuelven 0 por defecto cuando termina la ejecución del `main`, por lo que no hace falta indicarlo explícitamente con `System.exit(0)`.

```
      * My power is 85
      * My reclycled material is 0
WALL·E> pick Card
WALL·E says: I am happy! Now I have Card
WALL·E> scan
WALL·E says: I am carrying the following items
   Battery
   Card
   Petrol
WALL·E> drop Battery
Great! I have dropped Battery
WALL·E> drop Cord
You do not have any Cord.
WALL·E> scan
WALL·E says: I am carrying the following items
   Card
   Petrol
WALL·E> radar
Callao
In this square you can find a code card
The place contains these objects:
   Battery
WALL·E> turn right
WALL·E is looking at direction WEST
      * My power is 80
      * My reclycled material is 0
WALL·E> turn right
WALL·E is looking at direction NORTH
      * My power is 75
      * My reclycled material is 0
WALL·E> turn right
WALL·E is looking at direction EAST
      * My power is 70
      * My reclycled material is 0
WALL·E> move
WALL·E says: Moving in direction EAST
Colon
People concentrates here to watch football
The place contains these objects:
   Newspapers

      * My power is 65
      * My reclycled material is 0
WALL·E> turn left
WALL·E is looking at direction NORTH
      * My power is 60
      * My reclycled material is 0
WALL·E> operate Card
WALL·E> move
WALL·E says: Moving in direction NORTH
Exit
Ok, finally you have found your spaceship...
The place is empty. There are no objects to pick

      * My power is 55
      * My reclycled material is 0
```

```
WALL·E says: I am at my spaceship. Bye bye
```

4. Refactorización

En ingeniería del software se entiende por refactorización la reestructuración del código fuente alterando su disposición sin cambiar su comportamiento externo. Algunos objetivos de la refactorización son la mejora de la consistencia interna del código, claridad y extensibilidad. En esta práctica abordaremos los siguientes puntos:

- Cambio en el nombre del paquete principal, de `tp.pr2` a `tp.pr3`
- Creación de nuevos subpaquetes para una mejor estructuración de las clases. Concretamente aparecen los paquetes `tp.pr3.items` y `tp.pr3.instructions` que contendrán las clases relacionadas con los items y las instrucciones respectivamente.
- Cambio en la localización de la ciudad, el lugar en el que se encuentra WALL·E y la dirección actual a la que está mirando. En la Práctica 2 esta información se encontraba en la clase `RobotEngine`, mientras que en esta nueva versión se encontrará en la clase `NavigationModule`, encargada, como indica su nombre, de controlar la navegación de WALL·E por la ciudad. Este cambio implicará cambiar algunos métodos de sitio y realizar las modificaciones pertinentes.
- Cambios completos en la implementación de las instrucciones. La clase `Instruction` pasa a ser un interfaz. Se pasa a tener una clase por cada tipo de instrucción considerado e implementará la interfaz `Instruction`. Estas nuevas clases deberán ser capaces de ejecutar la instrucción que representan. También deberán ser capaces de construir la instrucción a partir de la cadena de texto que la describe y de indicar el texto de ayuda para esa instrucción; esas dos capacidades serán utilizadas desde la clase `Interpreter`.
- *Gestión de errores* de algunos de los métodos de la práctica anterior utilizando el mecanismo de excepciones en vez de la notificación vía el valor devuelto por la función. Consulta la documentación para ver qué métodos han cambiado.

Con las refactorizaciones anteriores, algunas clases de la práctica anterior dejan de tener sentido y deben borrarse. Comprueba en la documentación proporcionada qué clases han desaparecido y que nuevos paquetes aparecen para la declaración de las excepciones.

5. Recomendaciones generales

Para la implementación de la práctica aconsejamos partir de la práctica 2 e ir construyéndola por el siguiente orden, intentando que tras cada uno de los pasos la práctica siga pudiendo ser compilada y ejecutada:

- Cambiar los nombres de los paquetes y crear los paquetes nuevos.
- Cambiar la información sobre la ciudad, dirección a la que mira el robot, y lugar en el que se encuentra, de la clase `RobotEngine` a la clase `NavigationModule`.

- Refactorizar todo lo relacionado con las instrucciones, convirtiendo la antigua clase `Instruction` en una interfaz, creando una nueva clase por cada instrucción, haciendo que esta nuevas clases implementen la interfaz `Instruction` y modificando las partes afectadas (`Interpreter`, `RobotEngine`, etc.).
- Implementar las nuevas instrucciones `DropInstruction` y `RadarInstruction`.
- Implementar el cargador de mapas (clase `CityLoaderFromTxtFile`).

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.

Es indispensable que el código fuente supere los tests de unidad proporcionados y que el fichero enviado pase el programa validador publicado.

Sólo uno de los dos miembros del grupo debe hacerlo, subiendo al campus un fichero llamado `GrupoNN.zip`, donde `NN` representa el número de grupo con dos dígitos.

El fichero debe tener al menos el siguiente contenido[2]:

- 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.

[2]Puedes incluir también opcionalmente los ficheros de información del proyecto de Eclipse