Práctica 3:
Motor de Aventuras Conversacionales v1.0

Fecha de entrega: 20 de enero de 2011, 23:55 horas

Esta práctica consiste en completar la primera versión funcional del motor de aventuras conversacionales desarrollado en prácticas anteriores. Se realizarán varias mejoras y correcciones tanto a nivel de diseño como de código, además de añadir nuevas funcionalidades, entre ellas la capacidad básica que requiere cualquier motor de este tipo: poder jugar a juegos diferentes cuyas descripciones estén codificadas en ficheros independientes.

Todo lo relativo a los métodos y las herramientas de trabajo, así como al sistema de corrección y pruebas se mantiene similar a prácticas anteriores. El alumno debe crear un proyecto nuevo para cada práctica, siendo recomendable copiar solamente aquel código de prácticas anteriores que sea obviamente reutilizable.

1. Descripción de la práctica

La funcionalidad principal que añadimos al motor en esta práctica es la de cargar desde un fichero la descripción de un juego, creando el objeto correspondiente en memoria y pudiendo, por tanto, ejecutarlo y jugarlo.

Este sería un ejemplo parcial de fichero de texto con la descripción de un juego:

```
# Example Game: The Two Towels
game "THE TWO TOWELS" "John" "Three Rings for the Elven-kings..."

location forest "FOREST OF THONTUNIA" "You are a 'professional ..."
n cave

location cave "ENTRANCE OF THE CAVE" "Aha! The inside of ..."
item "skull" "This skull is really scary." 25
w smallPassage
ne bigPassage
s forest


[...]
```

```
[...]

location heaven "THE HEAVEN" "Now you are happy and ..." 50 "You
have achieved your goals, my friend!"
```

El fichero debe comenzar por una descripción del juego, que comienza por la palabra reservada **game** y continúa por las cadenas de texto del título, el autor y la descripción inicial del mismo.

A continuación se describen las localizaciones del juego una por una, comenzando por la palabra reservada **location** seguida de un identificador de la habitación y dos cadenas de texto: el nombre y la descripción de la misma. Opcionalmente cada localización puede llevar un número y una cadena de texto detrás, significando la puntuación mínima que debe tener el jugador para que el juego acabe al entrar en esta habitación y se muestre un mensaje de finalización del mismo.

Justo después de la descripción de una localización se pueden incluir las descripciones de los objetos que están situados inicialmente en ella. Esto se hace usando la palabra reservada **item**, más las cadenas de texto del nombre y la descripción, más el valor en puntos que tiene el objeto.

Justo después la descripción de los objetos de una localización se pueden incluir también las descripciones de las conexiones que dicha localización tiene con otras localizaciones. Comenzando con una palabra reservada de dirección (abreviaturas en inglés de las ocho direcciones básicas más "arriba" y "abajo": **n**, **ne**, **e**, **se**, **s**, **sw**, **w**, **nw**, **u** y **d**) y siguiendo por el identificador de la localización de destino.

Además de todas estas consideraciones, que introducen sus correspondientes nuevas funcionalidades en el motor, en el ejemplo puede verse como el fichero también pueden incluir comentarios (precedidos de **#**).

Por otro lado, nuestra aplicación también será capaz de usar de forma nativa un fichero de entrada, un fichero de salida o ambos para automatizar la ejecución y prueba del motor del juego. En caso de proporcionarle a la aplicación un fichero de entrada, se ejecutarán una serie de órdenes en secuencia dentro del juego, de manera no-interactiva, en vez de interactuar a través de la entrada estándar con el usuario como ocurre por defecto; y en caso de proporcionarle un fichero de salida, se guardarán todos los mensajes de salida en dicho fichero, en vez de mostrarlos por la salida estándar como ocurre por defecto.

Este sería un ejemplo parcial de fichero de entrada:

```
go north
help
pick skull
look
drop skull
[...]
```

Y este sería un ejemplo parcial de fichero de salida (nótese que las órdenes de entrada introducidas no forman parte de la salida):

```
FOREST OF THONTUNIA
You are a 'professional dungeoneer' of the Kingdom of Thontunia, a
brave guy who explores dangerous places looking for lost treasures.
Now you are wandering in the forest, ready for another exciting day
full of adventures. It seems there is a cave at the north...

>
ENTRANCE OF THE CAVE
Aha! The inside of the cave is really big. You can smell the aroma
of adventure... no, no, wait... what you are smelling is the stench
of trolls! You should be careful exploring the cave, trying not to
find a troll in your way... or you'll be part of his diet :-(
There are two passages in the cave, an small one at the west and a
big one at the northeast. Close to the northeast passage there is a
skull on the floor (ughs!).

>
[...]
```

Es importante recordar que los mensajes de salida, así como el comportamiento del motor y las distintas órdenes, vendrá decidido en función del fichero de configuración, cuyos valores por defecto figuran en la documentación Javadoc.

Finalmente hay un cuarto tipo de fichero con el que podrá trabajar nuestra aplicación. Se trata del fichero de configuración del motor, un fichero de propiedades (*properties*) también textual que permite cambiar la manera en que se comporta el motor definiendo los valores de una serie de parámetros y las cadenas de texto que deben usarse para cada uno de los mensajes que mostrará el juego. Este fichero permitirá introducir una gran flexibilidad a la hora de personalizar la experiencia del jugador sin tener que modificar el código fuente del motor, por ejemplo decidiendo si se debe mostrar el listado de los objetos presentes cada vez que se accede a una localización o su valoración en puntos.

Este sería un ejemplo de fichero de configuración que muestra el uso de algunas de las opciones disponibles:

```
# Engine configuration

flag.showLocationItems = true
flag.showItemValue = false
```

```
message.unknownCommand = Pardon?
message.endGame = GAME OVER

[...]
```

2. Especificación del diseño y la implementación

En esta práctica será necesario hacer cambios en el diseño y en el código, introduciendo nuevas clases y cambiando algunas que ya teníamos.

Como viene siendo habitual, se proporcionará al alumno la documentación Javadoc correspondiente a la realización de esta práctica, que debe estudiar con detenimiento y replicar con precisión a la hora de la implementación.

2.1. Corrección de errores

Se han cambiado los nombres de varias clases y métodos, para usar una nomenclatura más acorde con la genericidad del motor (**`Location`** en vez de **`Room`**, por ejemplo). Incluso los paquetes y subpaquetes han sido renombrados (**`es.ucm.fdi.lps.p3.command`** en vez de **`lps.pr3.commands`**), por lo que se aconseja al alumno que vaya a copiar código de la práctica anterior no tratar de realizar todos estos cambios manualmente sino aprovecharse de las funciones básicas de refactorización que proporciona Eclipse (como el renombrado automático de identificadores mediante *Alt+Shift+R*)

2.2. Nuevas clases

Revisando el diseño se ha redistribuido el código entre las clases existentes, llegando a crearse incluso una nueva, Engine, de manera que tengamos claramente diferenciado la clase que define el comportamiento del motor del juego de la clase que define el comportamiento del juego en sí.

Para automatizar la ejecución con ficheros de entrada y de salida en realidad no debemos hacer ningún cambio en el motor puesto que es algo para lo que ya estaba preparado. Lo que sí haremos es aprovechar para enriquecer la clase Main con el uso de JArgs, de manera que la aplicación acepte parámetros para lanzar la ejecución del motor como el fichero de configuración a utilizar, el fichero del juego, más los de entrada y salida "automatizada" opcionales.

```
Usage: run [{-c,--config} configFilePath] gameFilePath [{-i,--
input} inputFilePath] [{-o,--output} outputFilePath]
      (config, game, input and output file paths should include
their file extensions)
```

3. Pruebas del código

Los alumnos deben desarrollar sus propias pruebas unitarias e integrales del código del proyecto, en una carpeta `test` y con una estructura de paquetes

paralela a la que tiene el código fuente, tal y como se ha explicado en clase. La cantidad y calidad de las pruebas será un factor a valorar de cara a la calificación final.

4. Juegos de prueba y ficheros asociados

Los alumnos deben desarrollar al menos *un* juego de prueba, lo suficientemente complejo como para verificar que todas las características del motor han sido implementadas y funcionan correctamente. Para ello podrán incluir opcionalmente un fichero de configuración específico para el juego y *deberán incluir* al menos *tres* ficheros de entrada, con sus correspondientes ficheros de salida, que sirvan para automatizar tres ejecuciones completas del juego.

5. Instrucciones de entrega y corrección

La práctica deberá ser entregada usando los mecanismos habituales para entregas de prácticas a través del Campus Virtual. La entrega se realizará no más tarde de la fecha y hora de entrega establecida por el profesor que figura en la cabecera de este documento, y antes de la hora establecida por el profesor.

Sólo uno de los miembros del grupo realizará la entrega, subiendo al Campus Virtual un fichero comprimido en formato **ZIP** llamado **`lps`*`X`*`-g`*`NN`*`.zip`** siendo ***`X`*** el número de práctica y ***`NN`*** el número de grupo con dos dígitos.

El contenido del fichero será el siguiente:

- Directorio **`src`** con el código fuente Java de la práctica.
- Directorio **`test`** con el código fuente Java de las pruebas de los alumnos.
- Directorios **`game`*`ZZ`*** con los ficheros de configuración (**`game`*`ZZ`*`.cfg`**), juegos de ejemplo (**`game`*`ZZ`*`.game`**), ficheros de entrada (**`game`*`ZZ`*`_`*`WW`*`.in`**) y ficheros de salida (**`game`*`ZZ`*`_`*`WW`*`.out`**) de cada uno de los juegos desarrollados por los alumnos; siendo *ZZ* la numeración consecutiva de los juegos (01, 02, 03, …) y *WW* la numeración consecutiva de los ficheros de entrada y de salida válidos para un mismo juego (01, 02, 03, …).

La corrección se realizará en la sesión de laboratorio posterior a la fecha de la entrega y deberán asistir todos los integrantes del grupo para proceder a la defensa individual de su trabajo mediante una entrevista con el profesor.