PRÁCTICA DE LA ASIGNATURA
LABORATORIO DE PROGRAMACIÓN II
Curso 2008/09

Ingeniería Informática
Ingeniería Técnica en Informática de Sistemas
Ingeniería Técnica en Informática de Gestión

Roberto Rodríguez Echeverría (rre@unex.es)
Encarna Sosa Sánchez (esosa@unex.es)
José María Conejero (chemacm@unex.es)

PRISONBREAK REDUX

Entrega 3

Objetivos de la entrega

En esta tercera fase del proyecto, el objetivo es construir el sistema final a partir del prototipo, añadiendo todos los tipos de personaje que se usarán en la simulación. Este prototipo debe incluir la siguiente funcionalidad:

1. Creación de presos
2. Creación de guardias
3. Algoritmo de la simulación
    1. Control del movimiento de los personajes
    2. Información final: resultados de la simulación
4. Lectura del fichero de inicio: configuración de los personajes
5. Salida del sistema al fichero de registro según el formato especificado
6. Juegos de pruebas unitarias
    1. Juego de pruebas de cada unidad del sistema siguiendo un esquema de integración incremental

Creación de personajes

Inicialmente, se ha considerado que sólo participarán dos tipos de personajes en la simulación: presos y guardias. Ambos personajes deben ser tratados de manera uniforme por el algoritmo de la simulación.

Presos

Con el fin de identificar a cada preso dentro de la simulación, aparte de su nombre, se le asignará una marca que consistirá en una letra.

Como ya se especificó en el primer enunciado, antes del comienzo de la simulación, cada preso tendrá acceso al mapa de la prisión para poder trazar su ruta de escape. El algoritmo para el cálculo de dicha ruta se especificó en el enunciado de la EC2. Es necesario, sin embargo, realizar dos observaciones a este respecto:

- el preso debe crear la ruta completa desde la entrada de la planta más alta hasta la salida de la más baja,
- y cada preso usará una secuencia distinta de elección para saber qué celda debe seguir en un cruce. Por ejemplo, un preso puede tener la secuencia (N, S, E, O) y otro, la (S, E, N, O).

Una vez haya comenzado la simulación, cada vez que le toque, un preso realizará una serie de acciones en orden. Esta secuencia será:

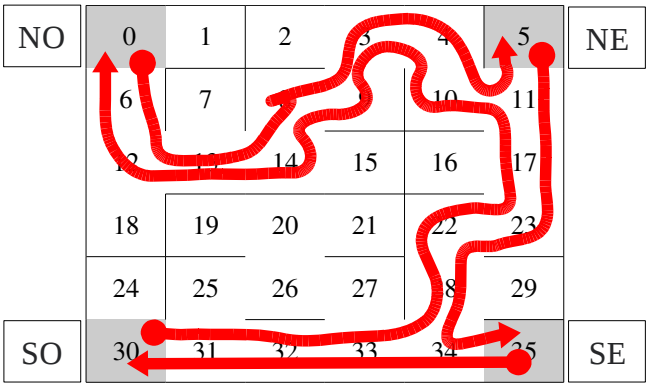
1. Si hay puerta en la sala, intentar abrirla usando las llaves que lleva consigo. Empezará a probar con la última llave que se haya encontrado.
2. Moverse a la siguiente sala según la orientación apuntada en su ruta. Si el movimiento supone un cambio de planta, se debe acceder a la siguiente planta por su sala de entrada.
3. Robarle una llave al primer guardia que se encuentre en la sala a la que llega. Esta acción no debe alterar el orden de movimiento de personajes de la sala.
4. Recoger una llave del suelo de la sala a la que llega, si se encuentra alguna.

Guardias

Para cada guardia es necesario mantener su nombre y la marca con la que se le identificará dentro de la simulación, que consistirá en una letra.

Cada guardia estará asignado a la vigilancia de una planta. Una planta puede tener uno o varios guardias asignados.

Antes del comienzo de la simulación, a cada guardia se le hará entrega de una copia propia de las llaves impares de su planta. Además, cada guardia debe calcular su ruta de vigilancia por la planta. La ruta de vigilancia de un guardia consiste en recorrer las rutas mínimas que unen las salas situadas en las esquinas de la planta en el sentido de las manecillas del reloj. Si suponemos que su ruta comienza en la esquina SE, debería seguir la ruta mínima que le lleve a la esquina SO. Desde allí seguir la ruta mínima a la esquina NO. Desde allí seguir la ruta mínima a la esquina NE. Y, finalmente, desde allí seguir la ruta mínima a la esquina SE, de nuevo. El guardia debe realizar esta ruta cíclica indefinidamente. La figura muestra cuál sería la ruta de un guardia en una prisión con una sola planta de 6x6.



Una vez haya comenzado la simulación, cada vez que le toque, un guardia realizará una serie de acciones en orden. Esta secuencia será:

1. Si hay puerta en la sala, cerrarla.
2. Moverse a la siguiente sala según la orientación apuntada en su ruta.
3. En el caso de que llegue a una sala en la que hay algún preso, debe capturar al primero y encerrarle en la celda de castigo de la prisión. Los presos que se encuentren en la celda de castigo no forman parte de la simulación para el resto de turnos. Esta acción no debe alterar el orden de movimiento de personajes de la sala.
4. Mientras disponga de llaves, debe perder una cada vez que entre en una sala con identificador par.

El guardia se va deshaciendo de las llaves en orden decreciente según sus identificadores tanto en la acción 4 de guardia como en la 3 de preso.

Algoritmo de la simulación

Para comenzar la simulación, todos los presos deben colocarse en la sala de entrada de la planta más alta. Mientras que cada guardia se colocará en la sala de la esquina sureste de la planta a la que está asignado.

Dado que los presos y los guardias van a recorrer las plantas moviéndose por las diferentes salas, la simulación establecerá un orden para que los personajes se muevan por turnos. De este modo, cada personaje sólo podrá moverse una única vez en cada turno. El orden establecido es el siguiente: primero se moverán los personajes de la planta superior, después los personajes de la planta inferior a ésta y así sucesivamente hasta llegar a la planta más baja. En cuanto al orden seguido en una misma planta, los personajes se moverán siguiendo el orden de las salas en las que se encuentren, primero los de la sala con identificador 0, seguidamente los de la sala 1 y así sucesivamente hasta haber movido a todos los personajes de todas las salas. Los personajes (tanto presos como guardias) que se encuentran en una sala se moverán siguiendo un orden histórico, es decir, aquellos que llegaron primero a la sala saldrán también los primeros de la misma.

La simulación puede llegar a su fin por dos situaciones distintas: (1) cuando se produzca la fuga de presos de la prisión, o (2) cuando todos los presos hayan sido capturados y llevados a la celda de castigo. En el primer caso, se debe completar el turno de manera que se escapen todos los presos que están en situación de hacerlo y aún no se han movido en ese turno.

Fichero de inicio

En esta entrega se añade la inicialización de dos nuevas entidades: PRESO y GUARDIA.

Los campos que forman la especificación de un preso son: nombre, marca, secuencia de orientaciones y turno en el que comienza a moverse. Ejemplo:

PRESO#el_culebras#C#NSEO#1#
Preso cuyo nombre es “el_culebras”, cuya marca es 'C', cuya secuencia de orientaciones para la elección de siguiente sala en un cruce es (Norte, Sur, Este, Oeste) y que comienza a moverse en el turno 1 de la simulación.

Los campos que forman la especificación de un guardia son: nombre, marca, número de planta y turno en el que comienza a moverse. Ejemplo:

GUARDIA#guardia001#G#0#3#
Guardia cuyo nombre es “guardia001”, cuya marca es 'G', que vigila la planta 0 y que comienza a moverse en el turno 3 de la simulación.

Un ejemplo de fichero de inicio podría quedar de la siguiente manera:

```
--Plantas de la prisión
PLANTA#0#10#10#0#99#30#5#
PLANTA#1#6#6#0#35#30#5#
--Personajes de la simulación
PRESO#@#@#NESO#1#
PRESO#$#$#ESON#3#
PRESO#X#X#SENO#5#
GUARDIA#G#G#0#1#
GUARDIA#U#U#1#3#
```

Con el objetivo de centrar el trabajo en los puntos más interesantes, se proporcionará el código necesario para el procesamiento básico de este fichero de inicio. Este código deberá ser extendido por cada uno para contemplar la creación de cada una de las posibles entidades de su sistema.

Fichero de registro

En esta fase se debe añadir al registro, por un lado, la ruta de cada personaje y, por otro, la información del estado de la simulación en cada turno.

Con respecto a la ruta de cada personaje, se usará la misma localización y el mismo formato de la EC2. Básicamente, en la información estructural de la prisión, se debe mostrar por planta, sustituyendo la ruta genérica que se mostraba en la EC2 por la ruta de cada personaje presente en cada planta. En el caso de que haya guardias, se debe mostrar su ruta de vigilancia (un ciclo). En el caso de que existan presos, se debe mostrar su ruta de escape completa. El orden en el que se muestran las rutas sigue el mismo que se usa para el turno. El formato para mostrar la ruta de un personaje es el siguiente:

```
(ruta:<marca del personaje>:<secuencia de orientaciones>)
```

Ejemplo para guardia:
(ruta:G: O N O N O N N N O O O S O S O O S S S N N N N N N N N N S S E E E N E E E N E E E O O O S S S O S E S S S E S E S E)

Ejemplo para preso:
(ruta:@: S S E E S E E S O S O S O S S E E E N N E E S E S E S E)

Con respecto al estado del sistema en cada turno, el formato del fichero de registro debe seguir esta especificación:

```
(turno:<turno>)
(celdacastigo:1101)
<para cada preso en la celda de castigo>
(<tipo de personaje>:<marca>:<id celda actual>:<turno_de_captura>: <llaves>)
<para cada planta de la prision>
(planta:<numero de planta>:<id celda entrada>:<id celda salida>:<número llaves>)
(puerta:<estado>:<altura de apertura>: <llaves probadas>)
<pintar mapa 2D de la planta incluyendo marca de presos[1]>
<para cada celda de la planta que contenga llaves>
(celda:<id celda>:<llaves>)
<para cada personaje en la planta (orden de turno)>
(<tipo de personaje>:<marca>:<id celda actual>:<turno>: <llaves>)
```

Al final de la simulación, se incluirá la información de resultados: presos fugados.

```
(presosfugados)
<para cada preso fugado>
(preso:<marca>:1111:<turno>: <llaves>)
```

Extracto de ejemplo del fichero de registro de una prisión con una planta:

```
(turno:11)
(celdacastigo:1101)
(preso:$:1101:11: 27 4 1)
(planta:0:0:99:30)
(puerta:cerrada:5:)
 _ _ _ _ _ _ _ _ _ _
| | |_ _ | |   _ |
|  _ _|    _  |_ _| |
|  _   _|  _     | |
|  _|_ X  |  _| |_| |
|  _ _|  G    | |   |
|  _|  @|_|_| |_|_|_|
|     _|_      |  _ _|
| | | | | |_|_  |  _|
| |    _ _   _|_  | |
|_|_|_|_ _ _ _ _|_ _|
(celda:10: 2 3 )
(celda:20: 5 6 )
(celda:22: 29 29 )
(celda:44: 17 )
(celda:46: 19 )
(celda:56: 21 )
(celda:66: 7 7 8 9 9 23 )
(celda:76: 10 11 11 12 13 25 )
(celda:77: 13 14 15 15 16 )
(celda:87: 17 17 18 19 19 )
(celda:88: 20 21 21 22 23 27 )
(celda:98: 23 24 25 25 26 29 )
(preso:X:33:11: 28 5 1)
(guardia:G:44:11: 15 13 11 9 7 5 3 1)
(preso:@:53:11: 27 3 0)
```

1 Cuando hay más de un preso en una celda, se muestra el número de presos en vez de la marca de los mismos.

Para ver un ejemplo completo de fichero de registro, se recomienda usar los generados por el ejecutable demo proporcionado por los profesores.

Consideraciones

- El nombre del fichero de inicio se especificará como un parámetro del ejecutable, según la sintaxis: <nombre_del_programa> <nombre_fichero>

Ejemplo: **`./prisionbreak inicio.txt`**

- El ejecutable debe admitir **dos modos de ejecución: normal y pruebas**. Por defecto, el ejecutable está en modo normal y realiza la ejecución de la implementación del sistema. El modo de ejecución se puede cambiar a pruebas mediante un parámetro opcional del ejecutable. En modo pruebas, el sistema ejecuta las pruebas unitarias definidas para el sistema en lugar de la funcionalidad del mismo. El parámetro que indica el modo de pruebas aparece al final de los parámetros de ejecución y contiene la cadena "TEST". Se trata de un parámetro opcional, por lo tanto, puede aparecer o no. Si no aparece, la ejecución se realizará en modo normal. La sintaxis de ejecución del sistema quedará: <nombre_del_programa> <nombre_fichero> [TEST]

Ejemplo: **`./prisionbreak inicio.txt TEST`**

- La salida por pantalla no tiene un formato predefinido, cada alumno puede hacerla como quiera. Opcionalmente, se puede proporcionar un interfaz gráfico de usuario según las bases trabajadas en las sesiones prácticas, que podrá servir para incrementar la nota final del proyecto.

- El fichero de registro debe denominarse **registro.log** y debe seguir **estrictamente** el **formato** definido en los enunciados**.**

- El contenido del fichero de **registro** resultante debe ser **contrastado** con el proporcionado por los profesores **de forma automática**, usando diff, kompare o meld.

- El proyecto **no** debe contener **ficheros fuente que no formen parte del ejecutable final.**

- El programa entregado debe ejecutar de principio a fin **sin pedir ninguna tecla.**

- Se debe hacer un uso correcto de los conceptos de **herencia** y **polimorfismo** en el desarrollo de la tercera fase del sistema.

- Se debe usar el mecanismo de **excepciones** para gestionar los errores del sistema.

- Uso común de los **contenedores** y **algoritmos** de la **STL**.

- Uso correcto del patrón de diseño **Template Method**.

- Implementación de las **pruebas unitarias de todas las clases** definidas.

- Generación de la **documentación externa completa** del sistema: ficheros PDF y XMI.

- Adicionalmente, se deben seguir las recomendaciones apuntadas en enunciados anteriores.

TERCERA PRUEBA DE LA EVALUACIÓN CONTINUA (EC3)

Cada alumno deberá realizar la **entrega de esta tercera fase del proyecto antes de presentarse a la modificación**. El código de la entrega contendrá **exclusivamente el código necesario para satisfacer las especificaciones** de la misma y nada más. Para realizar esta entrega se habilitará una **tarea en el aula virtual**. Posteriormente, el alumno **realizará la prueba de modificación sobre el mismo código que haya entregado** en el aula virtual.

Requisitos sobre el código de la EC3:

- El código debe estar perfectamente documentado con la notación doxygen y debe generar la documentación correctamente. No es necesaria la entrega de la documentación generada.
- El código debe estar escrito siguiendo un único estilo de programación.
- En la documentación interna del código debe indicarse:

  Nombre y Apellidos:
  Asignatura:
  Grupo:
  Número de Entrega:
  Curso:

Antes de la prueba se entregará:

a) Una carpeta cuyo nombre sea igual al identificador de correo-e del alumno (sin "@alumnos.unex.es"). Dentro de esta carpeta se incluirá solamente el proyecto desarrollado con los ficheros fuente que formen parte del ejecutable final **(*)**.

Al final de la prueba se entregará:

a) Una carpeta cuyo nombre sea igual al identificador de correo-e del alumno (sin "@alumnos.unex.es"). Dentro de esta carpeta se incluirá solamente el proyecto desarrollado con los ficheros fuente que formen parte del ejecutable final **(*)**.

En ambos caso, la carpeta creada será comprimida en un fichero en formato **ZIP** con el mismo nombre precedido del código de la entrega o modificación. En este caso, el código a utilizar para la entrega es "junio_" y para la modificación "junio_mod_". Ejemplo de nombre de fichero zip de la entrega: **junio_rre.zip**

La entrega de dicho fichero comprimido se realizará a través de una actividad propuesta en Avuex, de forma que cada alumno subirá a la plataforma virtual dicho fichero. Es muy importante que cada alumno suba su archivo con su cuenta en Avuex.

(*) Nota: deben eliminarse de los proyectos entregados aquellos ficheros que no son necesarios para su corrección, ejemplo: ficheros objeto y fichero ejecutable.