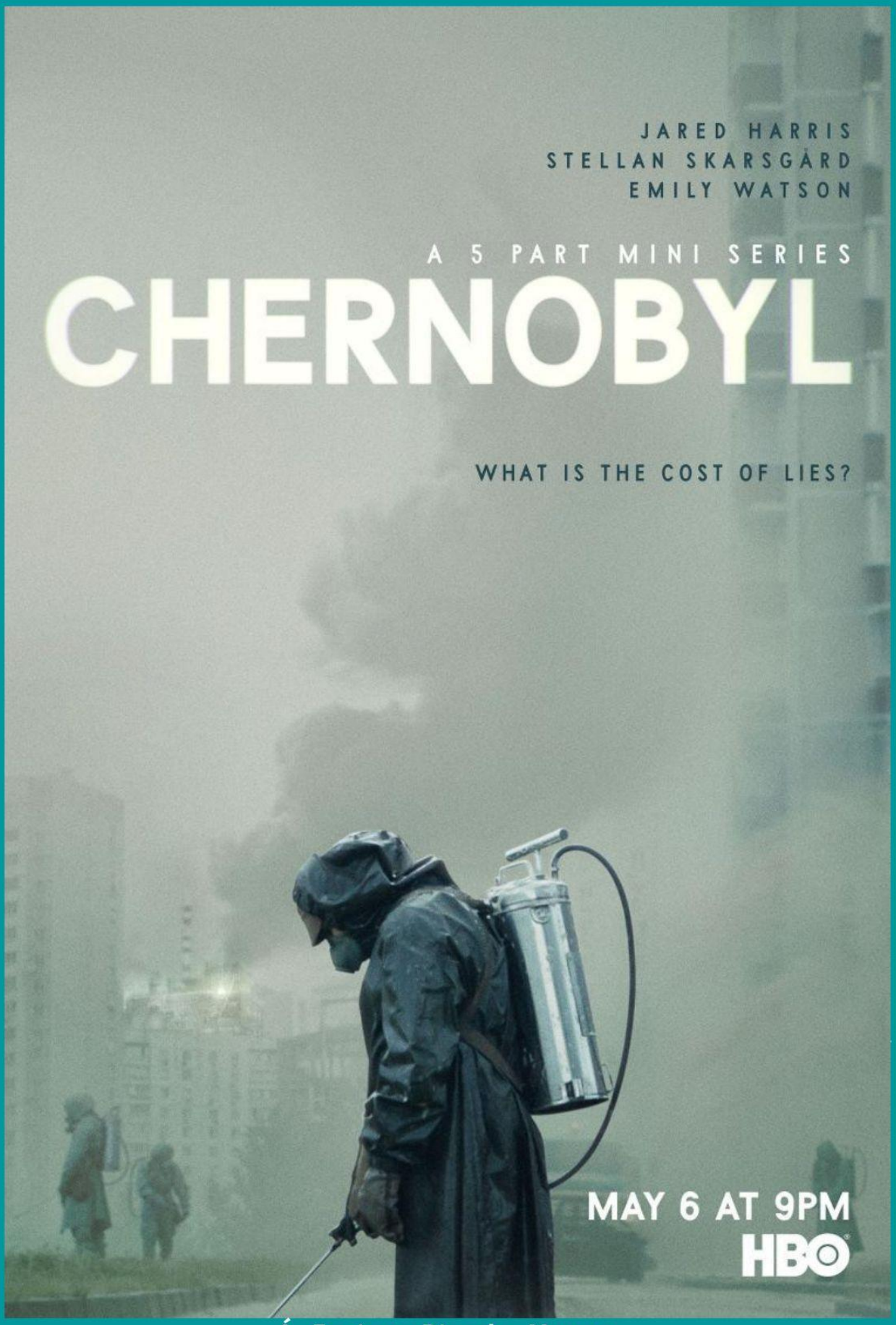
ENTREGA FINAL



Á. Enrique Pineda Navas
(quiquepineda@informatica.iesvalledeljerteplasencia.es)

CURSO 2022/2023

“Se meterán en el agua porque hay que hacerlo. Lo harán porque nadie más puede. Y si no, morirán millones (de personas)”

Boris Shcherbina (Stellan Skarsgård) a los trabajadores de la central nuclear, en la serie “Chernobyl”.

índice

Chernobyl

Nos encontramos ante la última entrega de la práctica *Chernobyl*. Esta entrega consta de dos partes bien diferenciadas:

- Finalización de la simulación.
- Ficheros: carga de datos y registro de la simulación.

En la finalización de la simulación debemos completar las acciones de los personajes de nuestra práctica: implementaremos cómo cogen nuestros científicos las llaves que se vayan encontrando a lo largo de la central nuclear, cómo destruirán esas llaves los miembros de la KGB, conseguiremos abrir las puertas que los personajes puedan escapar, etc. Además, ampliaremos la jerarquía de personajes añadiendo uno nuevo: los robots. Estos robots recorrerán la central nuclear en base a una ruta cifrada, registrando todas las celdas por las que vayan pasando…

En cuanto a la parte de ficheros, lo que conseguiremos será, principalmente, cargar los datos de la simulación desde un fichero de carga y de registrar la salida en los ficheros correspondientes.

Implementaciones

Una vez más, te tocará analizar este apartado metódicamente para saber cómo implementar los métodos de cada clase.

Constantes

Se considera una *buena práctica* que no aparezcan valores concretos a lo largo de la implementación de un programa, entendiendo por “valores” números enteros, cadenas de caracteres, etc. Por ello, crearemos una constante por cada uno de los valores que haya en nuestro código fuente.

Celda

Métodos

En esta última entrega, necesitaremos los siguientes métodos:

- Un método para retornar un personaje según una posición.
- Método que retornará el tamaño de la lista de personajes

CentralNuclear

Atributos

- lista de personajes, que almacenará todos Personajes que haya en la ejecución.

Métodos

- Insertar personaje en el atributo lista de personajes.
- Retornar el tamaño de la lista de personajes.
- Método que retorne el personaje de la lista dada una posición.
- Método para insertar una llave en la celda correspondiente.
- Método para borrar la llave recibida por parámetros.
- Método getAdyacentes, que recibirá una fila y una columna y, a partir de ellas, generará un vector con las celdas adyacentes.

Llave

Ningún cambio para esta entrega.

Puerta

Necesitaremos un atributo más. Este atributo servirá para almacenar el identificador de la llave con la que se puede abrir la puerta. Este atributo, por tanto, será de tipo entero.

Personaje

Ningún cambio reseñable en esta entrega.

Operador

Atributos

Necesitaremos crear un nuevo atributo booleano en esta clase:

- *enPuertaDeSalida*, que nos servirá para saber si el Operador ha llegado a la puerta de salida o no. Debemos tener en cuenta que, una vez nuestros Operadores lleguen a la puerta de salida, no deberán realizar ninguna acción más en ninguno de los turnos siguientes.

Métodos

- Debemos modificar el método que comprueba si un Operador está en la puerta de salida o no, de manera que si el atributo *enPuertaDeSalida* del Operador está a falso, comprobamos si acaba de llegar (cogemos su id de Celda y comprobamos si hay puerta en esa celda). En caso afirmativo, establecemos a true el atributo *enPuertaDeSalida*.

Pista: No podremos crear nunca un Operador en la puerta de salida.

Minero

Implementaremos un método para mostrar los kilos totales de escombros recogidos.

Bombero

Ningún cambio reseñable en esta entrega.

Científico

Atributos

Nuestros científicos tendrán un nuevo atributo: un *HashSet de llaves* donde ir almacenando todas las llaves que se encuentren a lo largo de su recorrido por la central nuclear. Debemos, por tanto, implementar los métodos necesarios para tratar este nuevo atributo.

Debemos recordar que, en esta última entrega, nuestros científicos realizarán dos acciones más: *cogerLlave* y *abrirPuerta*, por lo que el orden de las acciones para este tipo de personajes quedará de la siguiente manera:

Acciones:

- 1ª acción: Comprobar Puerta Salida.
- 2ª acción: Mover.
- 3ª acción: Coger llave.
- 4ª acción: Abrir puerta de salida.

Kgb

Ningún cambio reseñable en esta entrega.

Oficial

Ya sabemos que en esta última entrega los oficiales de la KGB tienen la capacidad de destruir llaves. Es por eso que debemos realizar la implementación del método en esta clase.

Voluntario

Ningún cambio reseñable en esta entrega.

Robot

En esta última entrega, contamos con un nuevo tipo de personaje: los *robots*. Estos robots se moverán a lo largo de la central nuclear en base a una serie de movimientos establecidos en su ruta (se moverán, por tanto, exactamente igual que el resto de personajes de la simulación). La diferencia está al cargar los movimientos en la ruta ya que, por motivos de seguridad, los robots reciben la ruta cifrada.

Aunque debemos tener en cuenta que la función principal de un robot es la de escanear su zona de influencia: esto es, comprobar si en sus celdas adyacentes hay alguna persona.

Ejemplo de ruta cifrada para un robot:

RNZXCSETEPLPOWSAAH

Si miramos más detenidamente la ruta cifrada, descubrimos lo siguiente:

R**N**ZXC**SE**T**E**PLP**O**W**S**AAH

Por tanto, la ruta descifrada sería:

[N] [S] [E] [E] [O] [S]

Atributos

Ya sabemos que los robots contarán con los atributos de los personajes y contarán, además, con un arrayList - puedes utilizar también una *cola* - para poder registrar los identificadores de las celdas por las que vayan pasando.

Métodos

Constará de un método para descifrar la ruta. Este método recibirá como parámetro un String y el método se encargará de añadir todos los movimientos (todas las letras ‘N’, ‘S’, ‘E’ y ‘O’ que haya en el String) a la ruta. Además, por supuesto, del método encargado de registrar los identificadores de celda por los que vayan pasando nuestros robots.

Por último, la clase *Robot* implementará el método escanear, que será la última de las acciones que tendrá que realizar cada robot en su turno.

Acciones:

El orden de las acciones que deben realizar los robots es el siguiente:

- 1ª acción: Mover.
- 2ª acción: Registrar movimiento.
- 3ª acción: Escanear.

Utilidad

Ningún cambio reseñable.

Nota importante: Todas las clases deberán implementar constructores por defecto, parametrizados, *getters* y *setters*. Tal y como se especifica en el apartado calificación, se valorará la eficiencia y la reutilización del código, que no tiene porqué estar especificada en este apartado.

Acciones de personajes

Ahora mismo, los personajes de Chernobyl son capaces de moverse a lo largo de la central nuclear, refrigerar las zonas calientes de la misma (recargando la batería del refrigerador si fuese necesario), recoger todos los escombros o, como en el caso de los miembros de la KGB, catalogar a los operadores nucleares con los que se encuentren.

A continuación, se detallan las nuevas acciones que debemos implementar para los personajes de *Chernobyl*.

Coger Llave

Prueba coger llave

Los científicos implementarán el método de manera que, una vez se hayan movido y, por tanto, se encuentren en la nueva celda, el método *coger llave* deberá obtener el identificador de la nueva celda en la que se encuentra, recoger la llave del "suelo" (en caso de que la haya) y almacenar dicha llave en la estructura de datos que tienen como atributo los científicos.

Cada vez que un Científico coja una llave, deberá mostrar el mensaje "[Nombre]: Llave recogida en celda X".

Ejemplo: "Shcherbina: Llave recogida en celda 57."

Destruir llaves

Prueba destruir llave

Los Oficiales de la KGB están muy bien entrenados y, además de catalogar a todos los operadores nucleares con los que coincidan, también se encargarán de ir destruyendo las llaves que se vayan encontrando a su paso.

Para ello, deberán comprobar si hay una llave en la celda en la que se encuentran y, si la hay, la destruirá y mostrará el mensaje correspondiente:

Ejemplo: "Vladimir: Llave recogida en celda 17."

NOTA: Para destruir una llave, basta con ponerla a *null*.

Abrir puerta

Prueba abrir puerta

Los encargados de abrir las puertas de la central nuclear para que todos los personajes puedan escapar son los Científicos.

Para que puedan abrir una puerta, los científicos deben ir probando todas las llaves que tengan hasta conseguir abrirla. Si alguno de los identificadores de las llaves del científico coincide con la "llave de la cerradura", la puerta se abrirá.

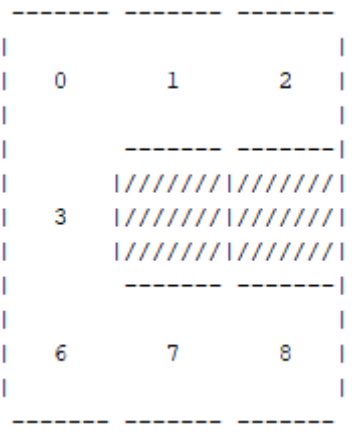
El método retornará un booleano para saber si se ha abierto la puerta o no.

Registrar movimiento

Prueba registrar movimiento

Como ya se ha comentado anteriormente, los robots se moverán a lo largo de la central nuclear registrando las celdas por las que vayan pasando.

Ejemplo:



Vamos a suponer que hay un robot en la celda 2 y que su ruta es:

[O] [O] [N] [S] [N]

Los identificadores que debería tener registrados este robot al final de la simulación serían:

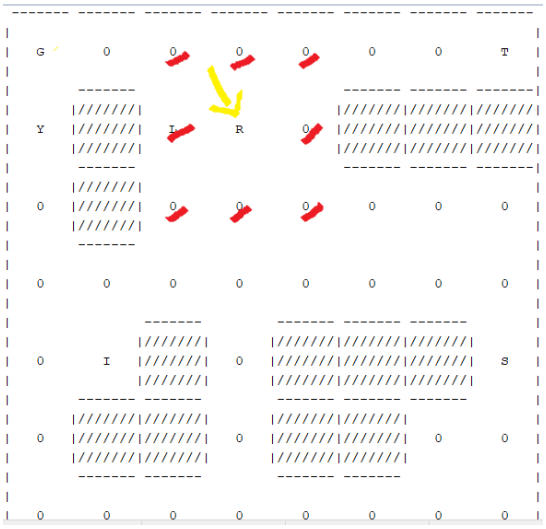
[1][0][-1][3][0]

IMPORTANTE: Ten en cuenta que si el robot no puede hacer un movimiento, registrará un -1.

Escanear

Prueba escáner robot

La función principal de los robots es la de escanear sus zonas de influencia. Este escaneo tiene como objetivo el de mostrar cuántos personajes hay alrededor del robot.



Turno "X" de la Central Nuclear.

Nos encontramos ante el Robot "*R*", situado en la celda 11 de la central nuclear.

La zona de influencia para ese robot, serían las celdas adyacentes a él:

- Norte.
- Noreste.
- Este.
- Sureste.
- Sur.
- Suroeste.
- Oeste.
- Noroeste.

En su zona de influencia se encuentra sólo un personaje: *L*, en la celda 10. El robot debería mostrar, por tanto, el siguiente mensaje:

[nombreRobot]: Detectadas 1 personas en la zona de influencia.

FINALIZADA SIMULACION 3

Simulación entrega final sin ficheros

Recursos facilitados al alumno

En esta última entrega, el profesor facilitará una serie de recursos al alumno para facilitarle a éste la implementación de la práctica *Chernobyl*.

CentralNuclear

Matriz de adyacencia para la central de 8x8.

Constructor clase CentralNuclear. Encargado de inicializar los atributos de la clase.

Método getInstancia. Método para crear la instancia del patrón Singleton.

Método inicializarEscombros. Encargado de añadir una serie de kilos en unas celdas concretas de la central nuclear.

NOTA IMPORTANTE: Recuerda que debes adecuar los métodos que muestran la central nuclear por pantalla para que ahora escriban en el fichero de simulación.

Programa principal

Se adjuntan ficheros para la realización de pruebas parciales del desarrollo de la práctica, así como el fichero de carga *inicio.txt* y los *log* con los que comprobar tu resultado.

Ficheros de carga:

- Simulación de la Central Nuclear de 8x8 (*inicio.txt*)

Ficheros *log* (finales) con los que el alumno debe comparar los suyos:

- Simulación de la Central Nuclear 8x8.
  - informes.log
  - simulacionEntregaFinal.log

Ficheros

En esta última entrega debemos seguir organizando nuestras clases. En esta ocasión debemos crear el paquete *Ficheros*, que contendrá las clases *Lectura* y *Escritura*, que servirán para cargar los datos y para registrar los datos de la simulación.

Lectura: Carga de datos Prueba lectura

La carga inicial del sistema (operadores nucleares, miembros de la KGB, puertas y llaves) se realizará al comienzo de la ejecución de la simulación mediante la lectura de un fichero de texto (*inicio.txt*) que contiene una configuración en un formato concreto. Cada línea del fichero de carga definirá los detalles de un determinado objeto de *Chernobyl*.

Todas las líneas del fichero de carga tienen una primera palabra que establecerá el tipo de objeto (MINERO, OFICIAL, LLAVE, etc) que nuestro sistema debe cargar. Después de ese primer *token*, nos encontraremos con los detalles de configuración del objeto a cargar (cada uno de estos tokens estará separado por el carácter #).

En el fichero de carga también nos encontraremos comentarios; comentarios que, por supuesto, no serán tenidos en cuenta en la carga de datos de la simulación. Dichos comentarios comienzan con --.

A continuación vamos a detallar las distintas líneas que nos podemos encontrar en nuestro fichero de carga:

CENTRAL#turnosNecesarios#

Esta línea nos indica el número de turnos que necesitará la simulación. Para ello, crearemos un nuevo atributo en la clase CentralNuclear que almacenará dicho dato.

CIENTIFICO#Nombre#turno#celdaActual#Marca#Ruta#

Indica que el objeto a cargar es un Científico con su nombre, el turno en el que comienza a realizar sus acciones, la celda desde la que va a comenzar a moverse y, por último, la ruta que debe seguir a lo largo de la central.

Ejemplo: CIENTIFICO#Legásov#1#10#L#ESSON#

Científico de nombre Legásov, que empezará a realizar sus acciones a partir del turno 1 y que comienza situado en la celda 10. El Científico se identificará con la marca “L” y la ruta que debe seguir es Este-Sur-Sur-Oeste-Norte.

MINERO#Nombre#turno#celdaActual#Marca#Ruta#

Igual que los Científicos. La línea detalla el nombre del Minero, el turno en el que comienza a realizar sus acciones, la celda desde la que va a comenzar a moverse y, por último, la ruta que debe seguir a lo largo del mapa.

Ejemplo: MINERO#Glújov#3#0#Z#EESS#

Minero cuyo nombre es Glújov, que empezará a realizar sus acciones a partir del turno 3 y que comienza situado en la celda 0. Se identificará con la marca “Z” y la ruta que debe seguir es Este-Este-Sur-Sur.

BOMBERO#Nombre#turno#celdaActual#Marca#Ruta#

Igual que los otros operadores nucleares.

Ejemplo: BOMBERO#Ignatenko#2#5#B#OOSE#

Bombero de nombre Ignatenko. Realizará sus acciones desde el turno 2 y empezará a moverse desde la celda 5. Su marca es una “B” y la ruta que debe seguir es Oeste-Oeste-Sur-Este.

OFICIAL#Nombre#turno#celdaActual#Marca#Ruta#

Al igual que los Operadores, la línea del fichero de carga que hace referencia a los Oficiales de la KGB consta del nombre del Oficial, el turno en el que comienza a realizar sus acciones, la celda donde se encuentra y la marca que lo distingue. El último token hace referencia a la ruta que debe seguir.

Ejemplo: OFICIAL#Vladimir#1#27#G#OOOEEE#

Oficial de la KGB de nombre Vladimir. Comienza sus acciones en el turno 1 y lo hace desde la celda 27. Su marca es la G y la ruta que debe seguir es Oeste-Oeste-Oeste-Este-Este-Este.

VOLUNTARIO#Nombre#turno#celdaActual#Marca#Ruta#

Misma estructura que los Oficiales, pero con VOLUNTARIO como primer token.

ROBOT##Nombre#turno#celdaActual#Marca#Ruta#

Tal y como sucede con el resto de personajes de *Chernobyl*, podremos cargar también la información de un robot a través de la línea del fichero de carga correspondiente. Ésta consta del nombre del Robot, el turno en el que comienza a realizar sus acciones, la celda donde se encuentra y la marca que lo distingue. El último token hace referencia a la ruta que debe seguir. La cual, no debemos olvidar, que estará cifrada.

Ejemplo: ROBOT#Spirit#20#47#R#RFFNHHJEEKLNLKO#

Robot de nombre Spirit que empezará a realizar sus acciones en el turno 20. Se situará en la celda 47 y tiene como marca la R. La ruta que debe seguir, una vez descifrada, será Norte-Este-Este-Norte-Oeste.

LLAVE#Código#idCelda#

Una línea de este tipo nos indica que el objeto a cargar en la simulación será una Llave y cuyo código (o *idLlave*) asociado será el especificado en el segundo token. La celda donde se situará esta Llave será la especificada en el idCelda.

Ejemplo: LLAVE#1#5#

Llave con código asociado 1 que se encontrará en la celda 5 de la central nuclear.

PUERTA#idCelda#llaveCerradura#

En este caso, el objeto a cargar sería una Puerta. La puerta se situará en la celda indicada por idCelda y se abrirá con la llave indicada en el token correspondiente a *llaveCerradura*.

Ejemplo: PUERTA#19#63#

Puerta situada en la celda 63 de la central nuclear. Esta puerta se abrirá con la llave con identificador 19.

-- Línea de comentario

Toda línea que comience con "--" hace referencia a un comentario del fichero de carga y, por tanto, no deberá ser cargada en nuestra simulación.

Ejemplo: -- Esto es un ejemplo de comentario.

Para cargar toda la información crearemos una clase *Lectura*. En dicha clase tendremos, entre otros, los siguientes métodos:

- Método *cargarFichero*, que retornará un booleano: *true* si todo ha ido bien y *false* en caso contrario.
  Ten en cuenta que cada línea leída se corresponde con un objeto a crear en nuestro programa. Por tanto, para cada línea debemos averiguar de qué tipo de elemento se trata. Una vez sepamos qué tipo de elemento debemos crear, será tan sencillo como crear el objeto con las características indicadas en el fichero e insertarlo en todas las estructuras necesarias.
- Método queElemento. Recibirá la línea leída del fichero y, ayudado de un vector que almacenará los distintos posibles token que nos podremos encontrar, buscará a qué tipo de objeto hace referencia la línea. El método retornará la posición del vector auxiliar en la que se encuentra el token buscado. De esta manera sabremos qué objeto tenemos que crear.

Escritura: Registro de la simulación

En informática, se usa el término *log* para referirse a la grabación secuencial en un archivo (conocido como *fichero log*) de todos los acontecimientos, eventos o acciones que afectan a un programa. De esta forma se constituye una evidencia del comportamiento del sistema.

En nuestra práctica, vamos a crear dos ficheros log (ambos serán tratados como ficheros de texto): uno registrará la simulación y otro registrará los diferentes informes.

SIMULACIÓN

Debemos crear, en la clase Utilidad, un método para generar el nombre del fichero que almacenará la *simulación*. De hecho necesitaremos este método para generar el nombre tanto del fichero que almacenará la simulación como del fichero que almacenará los informes.

simulacionEntregaFinal_DM.log

, siendo

- D, el día de la ejecución.
- M, el mes de la ejecución.

Ejemplo: simulacionEntregaFinal_164.txt, que almacenaría la ejecución realizada el 16 de abril.

Para ello, debemos:

- Crear los métodos necesarios para abrir los flujos (y otro método para cerrarlos) en la clase *Escritura*. Los métodos para abrir los flujos serán dos: uno retornará un *FileWriter* y el otro retornará un *PrintWriter*. El método para cerrar los flujos recibirá por parámetros el FileWriter y el PrintWriter a cerrar y los cerrará.
- Adaptar nuestro método mostrarMatriz para que, en esta última entrega, no muestre por pantalla la simulación sino que la escriba en el fichero log. (Se recomienda modificar el nombre de este método por el de *pintarMatriz*). En realidad, esta adaptación del método se basa en pasarle por parámetros el flujo adecuado para que pueda escribir en el fichero y, en lugar de mostrar por pantalla, especificarle que escriba en el fichero.
- Adaptar todos los métodos que escriban por pantalla en algún momento (mover, cogerLlave, destruirLlave, escanear, …) para que escriban en el *log*.

INFORMES

El nombre del fichero *informes* tendrá el siguiente formato:

informes_DM.log

, siendo

- D, el día de la ejecución.
- M, el mes de la ejecución.

Ejemplo: informes_84.log, que almacenaría la ejecución realizada el 8 de abril.

Para generar todos los informes en el fichero, crearemos los métodos correspondientes en la clase *Escritura* y, además, adaptaremos el método encargado de mostrar las celdas refrigeradas en la Central Nuclear para que, en lugar de mostrarlas, las escriba en el fichero log. Debemos adaptar también, para que escriba en el fichero log, el método que mostraba los kilos de escombros recogidos por un minero. Por último, adaptaremos el método que mostraba las celdas visitadas por los robots.

FINALIZADA ENTREGA 3

Commits

Listado de commits obligatorios para la última entrega de la práctica *Chernobyl*:

nº Commit	Comentario
1	*Clase Celda*
2	*métodos lista de personajes*
3	*métodos llaves*
4	*getAdyacentes finalizado*
5	*Clase Puerta*
6	*Clase Operador*
7	*Clase Minero*
8	*Clase Robot*
9	*acción coger llave*
10	*acción destruir llaves*
11	*acción abrir puerta*
12	*acción registrar movimientos*
13	*acción escanear*
14	*Finalizada simulación 3*
15	*Lectura*
16	*Escritura simulacion*
17	*Escritura informes*
18	*Finalizada Entrega 3*

Ejecución

Se recomienda al alumno/a realizar pruebas parciales de la práctica para ir comprobando si vamos haciendo todo correctamente o, en una de las pruebas parciales, localizamos algún fallo que debemos solucionar.

Para la realización de las pruebas parciales se adjuntan varios archivos que se recomienda utilizar según vayamos avanzando en el desarrollo de la práctica. Estos archivos son:

- *PruebaAbrirPuerta.txt*
- *PruebaCogerLlave.txt*
- *PruebaDestruirLlave.txt*
- *PruebaEscanerRobot.txt*
- *PruebaRegistrarMovimiento.txt*

En estos ficheros tendremos, en la parte superior, la información de los personajes que van a participar en la prueba (información que deberemos copiar en nuestro programa principal) y, justo debajo, el resultado de la ejecución de la misma (el cual debemos contrastar con la ejecución de nuestra prueba).

También se aporta el fichero *SimulacionEntregaFinalSinFicheros.txt* donde se encuentra el programa principal y el resultado de la ejecución de la práctica totalmente terminada, exceptuando la parte de ficheros.

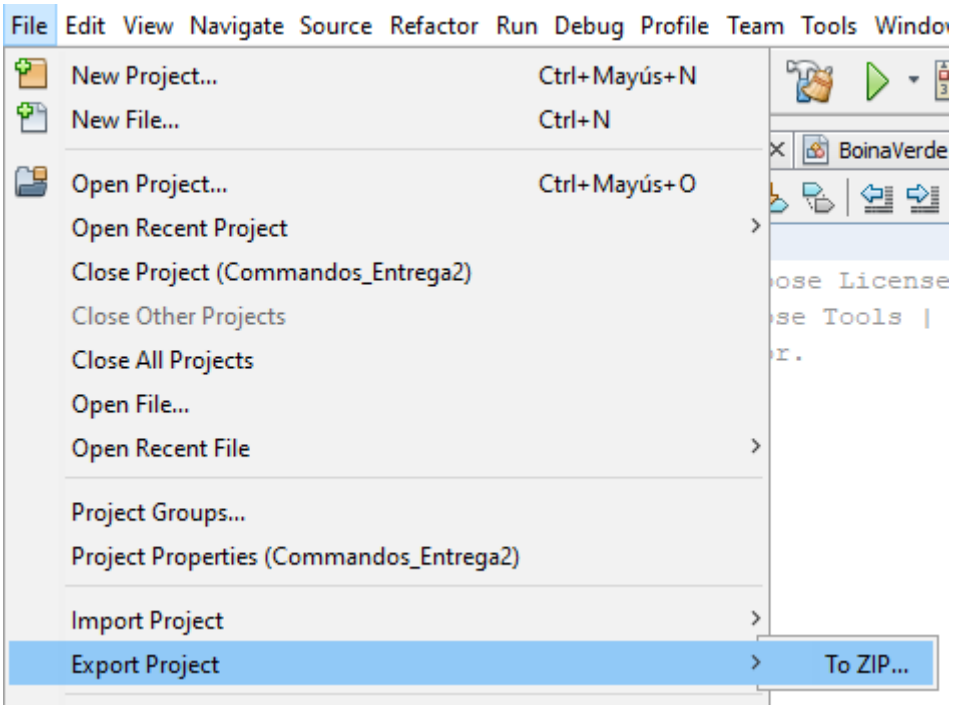
Una vez nos pongamos a implementar los ficheros, podemos realizar una prueba para comprobar si nuestra carga de datos se realiza correctamente. Para ello, disponemos del fichero *pgm ppal para probar Lectura.txt.* El resultado de la simulación debe coincidir, de nuevo, con la aportada en el fichero *SimulacionEntregaFinalSinFicheros.txt.*

Por último en cuanto al resultado completo de la ejecución de tu práctica para la Central Nuclear de 8x8, disponemos de los archivos *simulacionEntregaFinal.log* e *informes.log* (Recuerda que la ejecución de tu práctica debe coincidir completamente con la solución aportada en estos ficheros).

Entrega

La *Entrega Final: Robots, llaves y ficheros* se hará a través de una tarea que se habilitará días antes del examen trimestral en la plataforma Moodle (el formato de la entrega se hará tal y como se indicó en el enunciado de presentación).

Recuerda que para entregar tu proyecto debes exportarlo a ZIP (asegúrate de que tenga la extensión), tal y como se muestra en la siguiente imagen:



Para corregir la práctica es imprescindible que compile correctamente y ejecute. Además, toda práctica entregada fuera de plazo se considerará como no entregada.

Calificación

De forma general, los puntos a tener en cuenta para la evaluación del proyecto y su defensa son:

Bloque 0: Evaluación previa.

- Fecha de entrega.
- Interacción entre el profesor y el alumno.
- La ejecución de la práctica coincide completamente con la facilitada por el profesor.

Bloque 1: Aspectos formales y estéticos del código.

- Limpieza (No deja código que no se utilice, el código está ordenado, …).
- Sangría.
- Documentación interna.

Bloque 2: Código fuente.

- Corrección del código fuente.
- Estructuras de datos.
- Ficheros.
- Acciones.
- Constantes.
- Buenas prácticas de la Programación.
- Eficiencia y reutilización del código.
- etc.

Bloque 3: Funcionamiento.

- Ejecución de la práctica.
- Comparación de los ficheros generados.

Bloque 4: Defensa.

- Implementación eficiente.
- La ejecución de la defensa coincide completamente con la facilitada por el profesor.