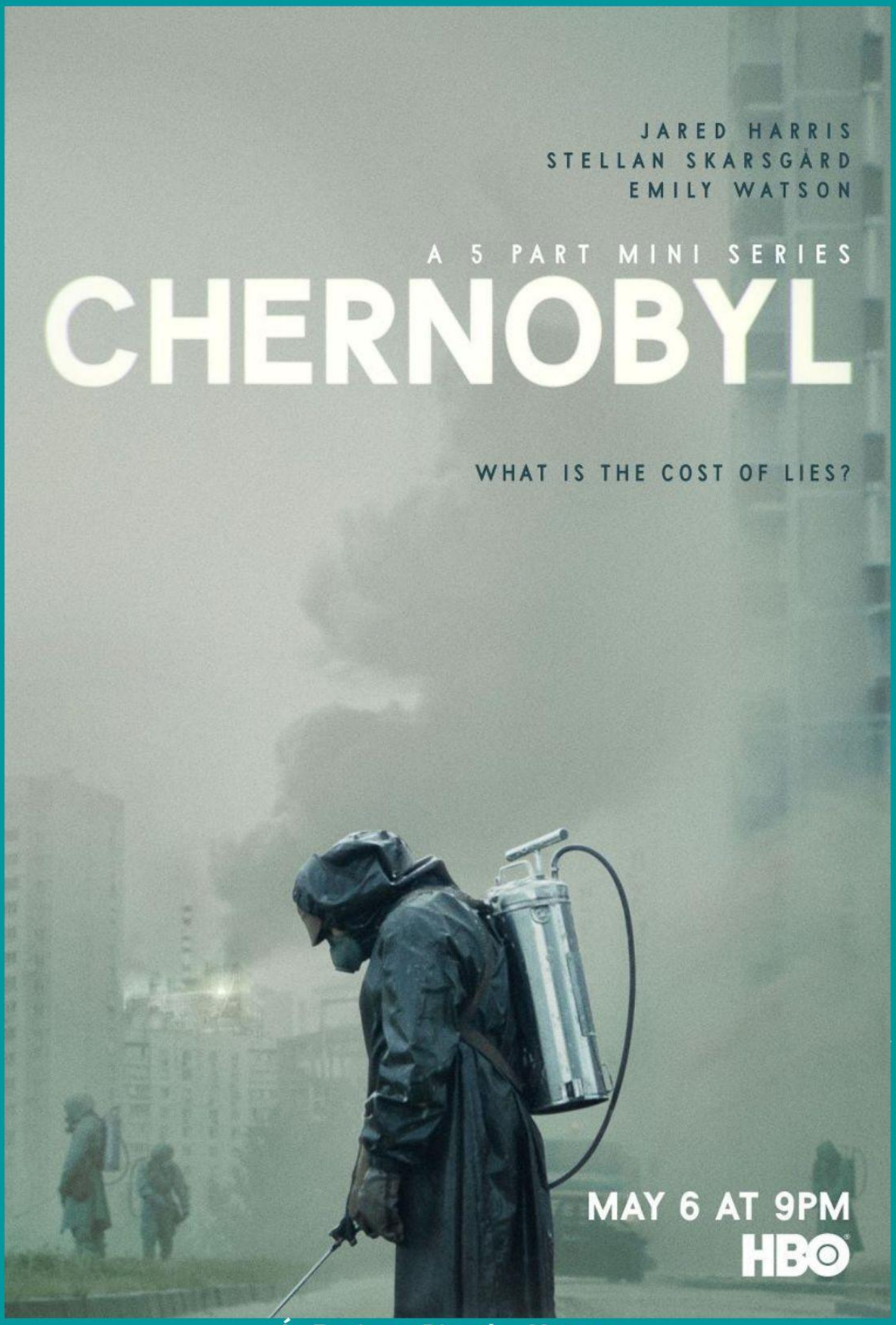
# ENTREGA 2




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

CURSO 2022/2023

*“¡NO! No hemos acabado (...) Solo hay un sitio en las instalaciones en el que hay grafito: dentro del núcleo. Si hay grafito en el exterior lo que explotó no fue el tanque de control sino el núcleo y está abierto”.*

Valery Legásov (Jared Harris), en la serie “Chernobyl”.

# índice

# Chernobyl

Nos enfrentamos a la segunda entrega de la práctica *Chernobyl*. En este enunciado se especificará todo lo necesario para que su desarrollo sea todo un logro.

En las especificaciones del enunciado anterior se detallaba cómo llevar a cabo la jerarquía de clases de los personajes, con sus atributos y métodos; además de la información necesaria para poder implementar la clase *Llave*, la cual será muy útil para que nuestros operadores nucleares puedan escapar de la central.

Recuerda que, en el enunciado anterior, las acciones de los personajes no hacían nada en especial. Simplemente, mostraban un mensaje. En este segundo enunciado nos centramos en la central nuclear y, ahora sí, en desarrollar algunas de las principales acciones de nuestros personajes, destacando sobre todas ellas la de *mover,* ya que en torno a ésta giran todas las demás.

# Patrón singleton e instanceof

### Patrón Singleton

El *patrón singleton* (o *instancia única*) es un patrón de diseño que permite restringir la creación de objetos pertenecientes a una clase.

Su objetivo consiste en garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a ella.

El patrón singleton se implementa creando en nuestra clase un método que crea una instancia del objeto sólo si todavía no existe alguna. Los constructores de dicha clase serán privados, para obligar así a utilizar el método que crea un objeto, garantizando de esta manera que existe una única instancia.

Ejemplo de implementación:

```
	//Atributo miCentral, que hace de instancia del patrón singleton
	prívate static CentralNuclear miCentral;

	//Constructor de la clase CentralNuclear.
	private CentralNuclear(){
		/* Aquí iría el código del Constructor, que es privado para asegurarnos que
se
			usa el método getInstancia*/
	}

	//Método para crear la instancia del patrón singleton
	public static CentralNuclear getInstancia(){
			if  (miCentral == null){
				miCentral = new CentralNuclear();
			}
			return miCentral;
	}
```

Ejemplo de utilización:

```
	//Si necesito la Central Nuclear en alguno de los métodos, utilizo el patrón
	//singleton mediante el método getInstancia().
	CentralNuclear central = CentralNuclear.getInstancia();
```

### Operador *instanceof*

Sirve para conocer si un objeto es de un tipo determinado. Por tipo, nos referimos a clase. Es decir, si el objeto *ES UN* para la clase especificada a la derecha del operador.

Por ejemplo, la siguiente sentencia *personaje instanceof Operador* retornaría *true* si el objeto de tipo Personaje ES UN Operador; y *false* en caso contrario.

# Turnos

La práctica *Chernobyl* está basada en turnos.

En cada turno, se realizarán todas las acciones de todos los personajes una única vez. Vamos a suponer que ejecutamos la práctica con 3 personajes: un Bombero (con su atributo turno a 1), un Científico (con su atributo turno a 3) y un Oficial (con su atributo turno a 1), donde las acciones de éstos personajes son las siguientes:

- Bombero: mover y recargarRefrigerador.
- Científico: mover y cogerLlave.
- Oficial: mover y catalogar.

Como la práctica está basada en turnos y en cada turno realizan las acciones todos los personajes, la ejecución de los primeros turnos sería la siguiente:

**Turno 1 de la simulación.**

- Turno del bombero: se mueve y recarga el refrigerador.
- Turno del Científico: no hace nada. Su turno (3) no coincide con el turno de la simulación..
- Turno del Oficial: se mueve y cataloga a un Operador Nuclear.

[Como ya les ha tocado el turno a todos los personajes de la ejecución - a los tres -, pasamos al siguiente turno de la simulación]

**Turno 2 de la simulación.**

- Turno del bombero: se mueve y recarga el refrigerador.
- Turno del Científico: no hace nada. Su turno (3) no coincide con el turno de la simulación.
- Turno del Oficial: se mueve y cataloga a un Operador Nuclear.

[Al igual que en el turno anterior, como para este segundo turno ya les ha tocado el turno a los tres personajes, pasaríamos al siguiente turno de la simulación]

**Turno 3 de la simulación.**

- Turno del bombero: se mueve y recarga el refrigerador.
- Turno del Científico: se mueve y coge una llave.
- Turno del Oficial: se mueve y cataloga a un Operador Nuclear.

[Y la simulación pasaría al siguiente turno]

**Turno 4 de la simulación.**

- Turno del bombero: se mueve y recarga el refrigerador.
- Turno del Científico: se mueve y coge una llave.
- Turno del Oficial: se mueve y cataloga a un Operador Nuclear.

**Turno 5 de la simulación.**

(...)

# Estructura de las clases

A estas alturas de curso ya sabrás que la organización de nuestro código es muy importante. Es por ello que, debido a que nuestra práctica va aumentando de tamaño, es recomendable que empecemos a organizarlo todo.

Crea dos paquetes (*Personajes* y *Genérico*) para organizar nuestras clases. Recuerda que la creación de paquetes la estudiamos en el tema 4.

Respecto a las clases de nuestro software, tendrás que analizar este apartado metódicamente para saber cómo implementar los métodos de cada clase.

## Constantes

Para esta entrega nos crearemos una nueva Clase llamada *Constantes*; esta clase no contendrá Constructores, ni getters, ni setters. Esta clase sólo declarará las constantes a utilizar en nuestra práctica. De momento, tendrás que crear *filas y columnas*, de tipo entero y con valor 8; y la constante *cero*, que será de tipo String y tendrá el valor que indica su propio nombre.

## Utilidad

Esta será una clase con dos únicos métodos (sin Constructores, ni getters, ni setters):

- calcularFila, que recibirá un identificador de celda y calculará, a partir de él, en qué fila se encuentra dicho identificador. Retornaremos ese valor.
- calcularColumna, que recibirá un identificador de celda y calculará, a partir de él, en qué columna se encuentra dicho identificador. Retornaremos ese valor.

## Llave

Ningún cambio para esta entrega.

## Puerta

A partir de este momento aparecen en nuestra práctica las puertas, para poder saber por donde pueden escapar los operadores de la central nuclear.

Atributos

- identificador de celda, que indicará en qué celda se encuentra situada la puerta.
- booleano que nos indicará si la puerta está abierta o cerrada. Una puerta nunca podrá ser creada ya abierta.

Métodos

- Ninguno en especial.

# Celda

Atributos

- identificador de celda, de tipo entero.

- arrayList de Personajes, que almacenará los personajes que se encuentren en la celda en cuestión.

- refrigerada, que será un booleano que nos indicará si la celda está ya refrigerada (ha pasado un bombero por ella) o no.

- escombros, número entero que indicará el peso, en kilos, de los escombros acumulados en la celda después de la explosión.

- Puerta, estará a null si no hay puerta de salida en la celda y, en cambio, almacenará un objeto de tipo puerta si en esa celda se encuentra la salida.

- Llave, a null si no hay llave en la celda y con el objeto correspondiente en caso de haber una llave en ella.

Pista: No podremos crear una celda que tenga una puerta ni una llave.

Métodos

- Método para insertar un nuevo Personaje al final de la lista de personajes.
- Método para buscar y retornar un Personaje de la lista de personajes a partir de su nombre.
- Método para borrar un Personaje a partir de su nombre.
- Método para buscar algún Operador en la Celda. Retornará *true* si existe al menos uno. *False* en caso contrario. Ver instanceof.

# CentralNuclear

Atributos

- miCentral, que será la instancia del patrón Singleton.

- turno, almacenará el turno de la simulación.

- adyacencia (constante facilitada por el profesor). Ver Recursos facilitados al alumno.

- matriz, que será una matriz donde almacenaremos objetos de la clase Celda.

Métodos

- Método *hayCamino*, que retorne *true* si podemos ir de una celda de origen a una celda de destino. *False* en caso contrario.
- Método que retorne el objeto Celda que se encuentre en la fila y la columna dadas.
- Método para insertar un Personaje en su celda., así como otro para insertar una puerta.
- Método para mostrar todas aquellas celdas que han sido refrigeradas.

# Personaje

Atributos

- ArrayList de caracteres (se llamará *lRuta*). Estos caracteres representarán la ruta a seguir por parte de los personajes de la práctica (‘N’ -Norte-, ‘S’ -Sur-, ‘E’ -Este-, ‘O’ -Oeste-).

Métodos

Debemos crear los siguientes métodos para poder trabajar con el arrayList:

- Método para cargar los movimientos en el atributo arrayList. Este método recibirá un vector de char e irá insertando todos los valores de dicho vector, uno a uno, en el arrayList.
- Método para borrar el primer movimiento de la ruta.
- Método para insertar un nuevo movimiento al final del arrayList.
- Método que retorne si el arrayList está vacío o no.
- Método *calcularSiguienteIdCelda*, que retornará el identificador de la celda a la que nuestro Personaje se debe dirigir según su ruta.
- Acción Mover. Mediante este método conseguiremos que nuestros personajes se vayan moviendo a lo largo de la central nuclear.

# Operador

Ya sabemos que los Operadores Nucleares se mueven a lo largo de la central hasta llegar a la puerta de salida. Una vez lleguen, no tendrán que volver a ejecutar ninguna acción más.

Métodos

- Acción comprobar puerta de salida.

# Minero

Uno de los cambios más significativos de esta entrega es que nuestros personajes empiezan a realizar sus acciones: comprobar si están en la salida, moverse, refrigerar en el caso de los bomberos, etc. Es por ello que los distintos Operadores Nucleares deben implementar el método *realizarAcciones*.

Acciones:

- 1ª acción: Comprobar Puerta Salida.
- 2ª acción: Mover. Si no está en la puerta de salida, se moverá. Si está en la salida, no tiene que hacer nada.
- 3ª acción: Desescombrar.

Atributos

- ArrayList de enteros *lEscombros*. En este arrayList se insertarán los kilos que vaya recogiendo el minero a su paso por las celdas.

Métodos

Debemos crear los métodos que creamos necesarios para poder trabajar con el arrayList.

## Bombero

Tal y como sucede con la clase anterior, debemos implementar el *realizarAcciones*. Para saber si el Bombero puede realizar las acciones que tiene asignadas, lo primero será comprobar la batería del refrigerador. Si ésta está descargada - batería a cero-, el bombero sólo realizará una acción: recargar la batería (no podrá moverse y, por tanto, no desaparecerá el movimiento de su ruta).

Acciones:

- 1ª acción: Comprobar Puerta Salida.
- 2ª acción: Mover.
- 3ª acción: Refrigerar.

## Científico

Al igual que en las clases Minero y Bombero, en esta clase Científico debemos implementar el método *realizarAcciones*. A continuación detallamos las acciones que debe realizar un científico.

Acciones:

- 1ª acción: Comprobar Puerta Salida.
- 2ª acción: Mover.

## Kgb

Todos los miembros de la KGB catalogarán a los Operadores Nucleares.

## Oficial

En esta entrega debemos implementar el método *catalogar*. Además, debemos implementar también el método *realizarAcciones*.

Acciones:

- 1ª acción: Mover.
- 2ª acción: Catalogar.

## Voluntario

En esta clase, al igual que en la anterior, debemos implementar el método *catalogar* y, además, implementar el *realizarAcciones*.

Acciones:

- 1ª acción: Mover.
- 2ª acción: Catalogar.

# Acciones de los personajes

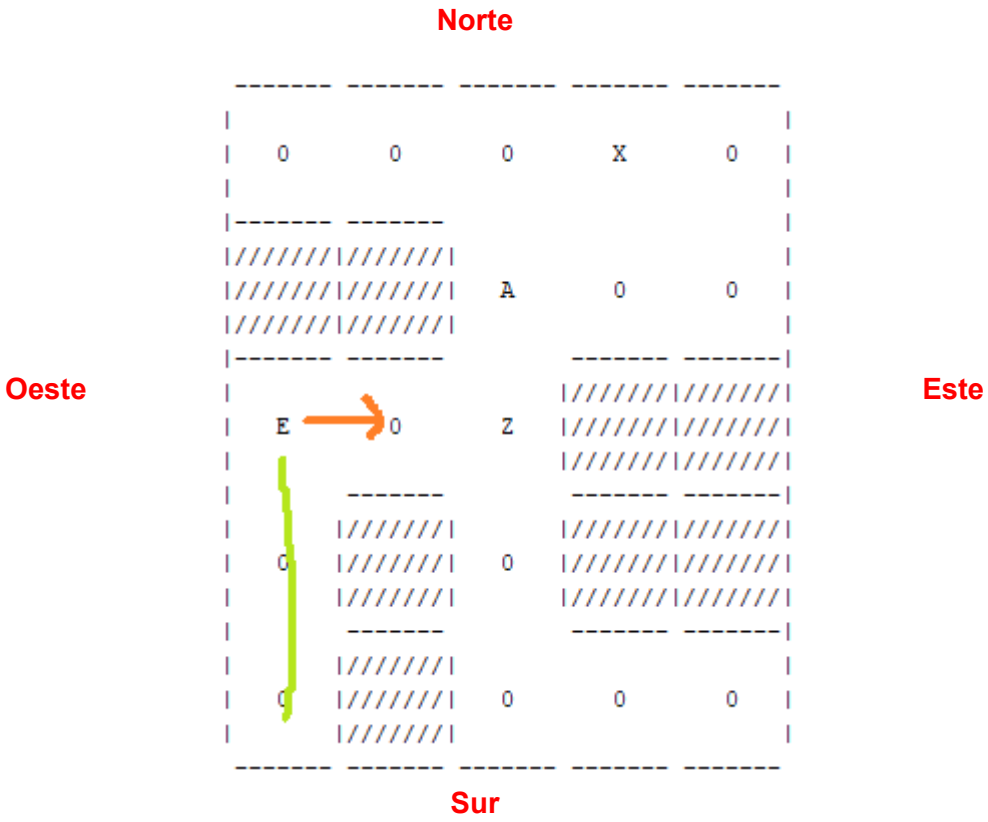
Siguiendo con las especificaciones de enunciados anteriores acerca de los personajes de la práctica, ampliamos en esta entrega las acciones que debe realizar cada uno de ellos.

# Acción Mover

Debemos tener en cuenta que las rutas de los personajes se especifican en función de los puntos cardinales.

Cada movimiento indica el punto cardinal al que debe dirigirse nuestro personaje. Por ejemplo, si la ruta es N-N-E-E-… (Norte, Norte, Este, Este, ...) y el personaje ha realizado ya dos movimientos, en su tercer movimiento debe moverse a la celda que se encuentre al Este de la celda en la que está actualmente.

La siguiente imagen muestra el ejemplo de ruta comentado anteriormente para un Bombero que **ha empezado a moverse en la celda 20 y ahora se encuentra en la 10**: su **siguiente movimiento le llevará a la 11**.




Ten en cuenta que, para obtener la información de la celda en la que se encuentra el personaje, necesitarás utilizar el *patrón singleton*.

Para recrear el movimiento de nuestros personajes, nuestro método *mover* debe, a grandes rasgos, comprobar si hay camino entre la celda en la que se encuentra el personaje (origen) y la celda a la que se tiene que mover (destino). Si hay camino, entonces debemos eliminar el personaje de la celda origen e insertarlo en la celda destino.

Debemos tener especial cuidado puesto que la acción de mover es la más importante de todas ellas, puesto que sin la correcta implementación de ésta acción, no podríamos probar el resto de acciones con garantías.

## Acción comprobar puerta de salida

Para saber si un Operador Nuclear se encuentra en una puerta de salida, lo único que debemos hacer es comprobar si el atributo *puerta* de la celda en la que se encuentra el personaje está a *null* o no. Si no está a null, quiere decir que ahí hay una puerta de salida.

## Acción Recargar refrigerador

Para simular que los bomberos recargan la batería del refrigerador, lo único que debemos hacer es poner a 5 el valor del atributo *bateriaRefrigerador*.

Ten en cuenta que si la batería de un refrigerador está totalmente descargada, el bombero no podrá realizar ninguna acción durante un turno y mostrará el mensaje "[nombre bombero]: Recargando batería refrigerador."

## Acción Refrigerar

Todos los bomberos tendrán que refrigerar la celda en la que se encuentren. Para llevar a cabo esta acción, los bomberos tendrán que poner a true el atributo refrigerada del objeto Celda en el que se encuentren. Para obtener la información de la celda, necesitaremos, de nuevo, el *Patrón Singleton*.

NOTA: No debemos olvidar decrementar la batería del refrigerador.

## Acción Catalogar

El objetivo de los miembros de la KGB no es otro que el de espiar a los operadores nucleares. Por eso, cuando un miembro de la KGB coincida con uno o más operadores nucleares en una celda, debe catalogar a ese operador u operadores.

En esta segunda entrega, la acción de catalogar consistirá en actualizar el atributo *catalogado* del Operador y mostrar por pantalla la información del Operador con el que coincida el miembro de la KGB, teniendo en cuenta que no recopilan la misma información un Oficial que un voluntario.

- Oficial: Mostrará por pantalla qué tipo de operador nuclear es (minero, bombero o científico), el nombre del operador, la celda en la que se lo ha encontrado y su marca.

  NombreOficial: TipoOperador#nombre#celdar#MarcaOperador

  Ejemplo: Vladimir: Cientifico#Legasov#22#L

- Voluntario: Mostrará el nombre del operador, la celda en la que se encuentra y su marca.

  NombreVoluntario: nombre#celdar#MarcaOperador

  Ejemplo: Vladimir: Legasov#22#L

# Acción Desescombrar

Es el objetivo principal de los mineros: retirar los escombros de la central nuclear derivados de la explosión.

En cada celda tendremos una serie de kilos de escombros (recuerda que hay un atributo en la clase Celda específico para ello) y nuestros mineros tendrán que retirarlo. Para simular esta acción, nuestros mineros utilizarán la lista de enteros que tienen como atributo, de manera que tendremos que insertar en ella los kilos de escombros de las celdas por las que pasan. Sin olvidarnos de poner a cero los kilos de la celda limpia de escombros.

Una vez recogidos los escombros de la celda, el minero debe mostrar el mensaje "[Nombre]: Llevo X kilos de escombros." donde refleje el total de kilos de escombros que lleva recogidos.

Ejemplo: "Glújov: Llevo 7 kilos de escombros."

# Orden de las acciones

Una vez haya comenzado la simulación de *Chernobyl*, cada vez que le toque el turno a uno de los personajes, éste realizará una serie de acciones en orden.

**Operadores Nucleares**

La secuencia de acciones para los Operadores Nucleares, sin tener en cuenta las acciones específicas de cada uno de ellos, será la siguiente:

- 1ª ACCIÓN: Comprobar Puerta Salida.
  Los Operadores deberán comprobar, antes de realizar ninguna otra acción, si la celda en la que se encuentran tiene una puerta de salida.
- 2ª ACCIÓN: Mover.
  Si no han llegado a la salida de la Central, nuestros Operadores deberán realizar el movimiento que les toque, tal y como estará marcado en su ruta, para llegar a la puerta de salida.

Los bomberos, en esta segunda entrega, tienen un par de acciones extra: *recargar refrigerador* y *refrigerar*, que será la última de sus acciones.

- ACCIÓN BOMBEROS: Recargar Refrigerador.
  Antes de nada, los Bomberos deberán comprobar si tienen agotada la batería del refrigerador. Si la tienen agotada, deberán recargarla.
- ÚLTIMA ACCIÓN: Refrigerar.
  Tendrán que refrigerar la celda a la que acaban de moverse.

**KGB**

La secuencia de acciones para los miembros del KGB son las siguientes:

- 1ª ACCIÓN: Mover.
  Realizarán su movimiento siguiendo la ruta marcada según su atributo correspondiente

- 2ª ACCIÓN: Catalogar.

Catalogarán a los Operadores con los que se encuentren en la misma celda.

# Recursos facilitados al alumno

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

## CentralNuclear

Constante matriz de adyacencia. La matriz de adyacencia es una matriz cuadrada que se utiliza como una forma de representar relaciones binarias, de manera que:

- Las filas y columnas representan las celdas en las que se divide la central nuclear.
- Las relaciones que hay entre los distintos pares de celdas se representan con un 0 (no hay camino entre ellas) ó un 1 (sí existe camino).

A continuación vamos a ver un ejemplo de matriz de adyacencia de una central nuclear de tamaño de 3x3:

| | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **1** | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **2** | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **3** | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| **4** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **5** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **6** | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| **7** | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| **8** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

idCelda

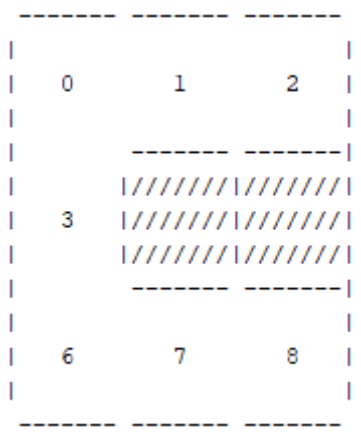
Ejemplos:

Desde la celda 1 podemos ir a la celdas 0 y 2.
Desde la celda 2 podemos ir a la celda 1.

Desde la celda 4 no podemos ir a ninguna celda.

Desde la celda 7 podemos ir a la celdas 6 y 8.

La matriz de adyacencia anterior nos generaría la siguiente central nuclear en nuestra práctica:




, donde los números de las celdas se corresponden con sus identificadores de celda.

Constructor clase CentralNuclear. Encargado de cargar los datos necesarios para la ejecución de la práctica.

**NOTA IMPORTANTE:** También se facilitarán al alumno los métodos necesarios para mostrar la central nuclear por pantalla.

# Celda

Para esta clase el profesor facilitará el método necesario para mostrar el contenido de la celda.

# Programa principal

Se facilita al alumno un archivo para la ejecución de la simulación en la central nuclear de 8x8:

- Simulación de Central 8x8 (Archivo *pgm ppal Entrega 2.txt*)

Para poder ejecutar la prueba final de *Chernobyl*, deberás copiar y pegar el archivo citado arriba y adaptarlo a tu práctica.

# Commits

Listado de commits obligatorios para esta segunda entrega de la práctica *Chernobyl*:

| nº Commit | Comentario |
|---|---|
| 1 | *Clase Constantes* |
| 2 | *Clase Utilidad* |
| 3 | *Clase Puerta* |
| 4 | *Clase Celda* |
| 5 | *metodo buscarPersonaje* |
| 6 | *metodo buscarAlgunOperador* |
| 7 | *metodo hayCamino* |
| 8 | *metodo mostrarCeldasRefrigeradas* |
| 9 | *metodo lRuta vacio* |
| 10 | *metodo calcularSiguienteIdCelda* |
| 11 | *métodos lEscombros* |
| 12 | *acción mover* |
| 13 | *mover finalizado* |
| 14 | *acción comprobar puerta salida* |
| 15 | *acción recargar refrigerador* |
| 16 | *acción refrigerar* |
| 17 | *acción catalogar oficiales* |
| 18 | *acción catalogar voluntarios* |
| 19 | *acción desescombrar* |
| 20 | *Finalizada Entrega 2* |

# 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 tres archivos que se recomiendan utilizar una vez terminemos la implementación del método mover. Estos archivos son:

- *PruebaMoverUnPersonaje.txt*
- *PruebaMoverDosPersonajes.txt*
- *PruebaMoverSeisPersonajes.txt*

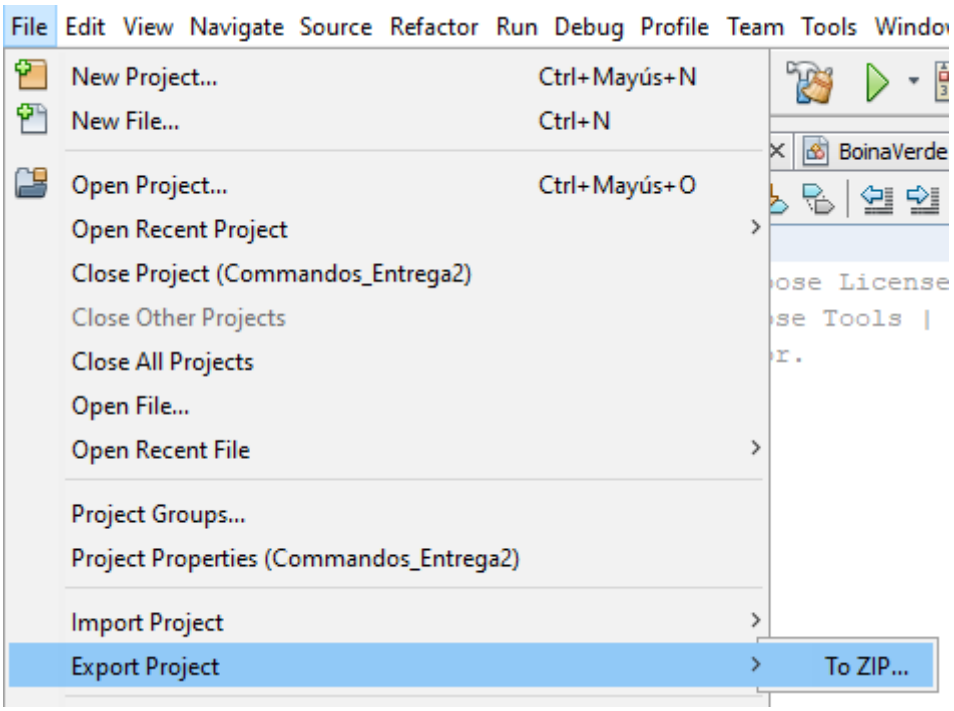
En estos ficheros tendremos, en la parte superior, la información de los personajes que van a participar en la prueba (información que deberemos copiar en nuestro programa principal) y, justo debajo, el resultado de la ejecución de la misma (el cual debemos contrastar con la ejecución de nuestra prueba).

En cuanto al resultado completo de la ejecución de tu práctica para la Central Nuclear de 8x8 se encuentra en el archivo *simulacionEntrega2.txt* (Recuerda que la ejecución de tu práctica debe coincidir completamente con la solución aportada en este fichero).

# Entrega

La *Entrega 2: Central nuclear, estructuras de datos y movimientos* 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, 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.
- Utilización de GitHub.
- Interacción entre el profesor y el alumno.

**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.**

- Código de la 1ª evaluación
- Clases Celda y CentralNuclear.
- Acciones.
- Eficiencia del código y de las estructuras de datos.
- Optimización del código.
- etc.

**Bloque 3: Funcionamiento.**

- Ejecución de la práctica.

**Bloque 4: Defensa.**

- Consistirá en una modificación de tu práctica. La calificación será de *Apto* o *No Apto*..