PRÁCTICA DE LA ASIGNATURA
DESARROLLO DE PROGRAMAS

Curso 2017/18

Grado en Ingeniería en Informática en:

Ingeniería del Software

Ingeniería de Computadores

(Idea original de Roberto Rodríguez Echeverría)

Entrega 2 (Lunes, 27 de noviembre)

En este documento se especifican las acciones que deben llevarse a cabo para implementar la segunda entrega del proyecto. El objetivo es construir un prototipo del sistema final, que incluya principalmente la definición estructural del mismo. Este prototipo debe incluir la siguiente funcionalidad:

1. Lectura del fichero de inicio de la configuración inicial del sistema.
2. Creación de la configuración inicial del mapa generando los caminos en el mismo.
3. Control de acciones iniciales que debe realizar cada uno de los personajes (en una entrega posterior se definirán algoritmos para el cálculo de la ruta de cada uno de ellos).
4. Control de errores en el sistema mediante mecanismos de control de errores existentes.
5. Primer juego de pruebas unitarias del sistema (**opcional en esta entrega, aunque recomendable**).
6. Registro de los resultados obtenidos a un fichero de texto con el formato especificado (**opcional en esta entrega, aunque recomendable**).

A continuación se detalla el contenido de cada una de las funcionalidades que debe incluir el sistema.

1.- Configuración inicial del sistema

La configuración inicial del sistema (mapa, personajes, etc.) se cargará al comienzo de la ejecución del sistema mediante la lectura de un fichero de texto que contiene esta configuración en un formato concreto. Cada línea del fichero de configuración definirá los detalles de una determinada entidad del sistema a simular. Así, cada línea tendrá un primer token que define el tipo de entidad (MAP, SHPHYSICAL, SHEXTRASENSORIAL, SHFLIGHT, VILLAIN, ...) y después una lista de tokens con los detalles de configuración de esa entidad. Cada uno de estos tokens estará separado por un carácter #.

Pueden existir además líneas adicionales de comentario que no serán tenidas en cuenta en el proceso (comienzan con --).

Los campos que forman la especificación del mapa son: ancho, alto, sala Daily Planet y altura de control del portal del Hombre Puerta. Por ejemplo: ***MAP#10#5#49#4#***
Mapa con 10 salas de ancho y 5 de alto, en el que la sala Daily Planet está en la sala 49. La altura de control del portal es 4 (condición de apertura).

--Comment example
Cualquier línea que comience por los caracteres –- representa un comentario en el fichero de configuración y por tanto no será tenida en cuenta en el proceso del concurso.

Los campos que forman la especificación de un personaje de tipo **SHPHYSICAL** son: nombre, marca y turno en el que comienza a moverse. Ejemplo: **SHPHYSICAL*#Daredevil#D#2#***
Personaje cuyo nombre es “Daredevil”, cuya marca es 'D' y que comienza a moverse en el turno 2 de la simulación.

Los campos que forman la especificación de un personaje de tipo **SHEXTRASENSORIAL** son: nombre, marca y turno en el que comienza a moverse. Ejemplo: **SHEXTRASENSORIAL*#ProfessorX#P#1#***
Personaje cuyo nombre es “ProfessorX”, cuya marca es 'P' y que comienza a moverse en el turno 1 de la simulación.

Los campos que forman la especificación de un personaje de tipo **SHFLIGHT** son: nombre, marca y turno en el que comienza a moverse. Ejemplo: **SHFLIGHT*#Eternity#E#3#***
Personaje cuyo nombre es “Eternity”, cuya marca es 'E' y que comienza a moverse en el turno 3 de la simulación.

Los campos que forman la especificación de un personaje de tipo ***VILLAIN*** son: nombre, marca y turno en el que comienza a moverse. Ejemplo: ***VILLAIN#Abomination#A#3#***
Villano cuyo nombre es “Abomination”, cuya marca es 'A' y que comienza a moverse en el turno 3 de la simulación.

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

```
--Map of the game
MAP#10#10#99#5#
--Characters in the simulation
SHPHYSICAL#Daredevil#D#2#
SHEXTRASENSORIAL#ProfessorX#P#1#
SHFLIGHT#Eternity#E#3#
VILLAIN#Abomination#A#3#
VILLAIN#Viper#V#2#
SHPHYSICAL#Wolverine#W#4#
```

Con el objetivo de centrar el trabajo en los puntos más interesantes, se proporcionará el código necesario (proyecto *Cargador*) 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 (esta acción será explicada en una sesión de laboratorio).

2.- Creación de la configuración inicial del sistema

Algoritmo de Kruskal.

Este algoritmo parte de una estructura base del mapa (ver ilustración 1) en la que todas las salas están aisladas (tienen paredes por todos lados) y su funcionamiento básico consiste en ir derribando paredes para crear pasadizos entre las salas. El resultado del algoritmo será una distribución de salas parecida a la mostrada en la ilustración 4. A continuación se explica de manera detallada el algoritmo:

1. Inicialmente cada sala está marcada con un valor numérico diferente (coincidente con el identificador de la sala, tal y como se muestra en la ilustración 1).
2. El algoritmo consiste en intentar derribar todas las paredes del mapa, teniendo en cuenta que no puede derribarse una pared si separa dos salas con la misma marca. En caso de separar dos salas con marcas diferentes, la pared puede ser derribada y las salas que quedan conectadas deben ser marcadas con el mismo identificador. Si una de estas salas estaba conectada con otras salas (tenían la misma marca), todas las salas conectadas por el camino creado quedarán marcadas con el mismo valor numérico (ver ejemplo en Figura 3, marca 13). Por otro lado, en caso de separar dos salas con marcas iguales, la pared no se derriba. Evidentemente, las paredes exteriores del mapa no se pueden derribar en este proceso.
3. La selección de una pared para ser derribada se debe hacer aleatoriamente tomando como base el conjunto de paredes que existen en el mapa. Hay que tener en cuenta que cada vez que se seleccione una pared, sea o no derribada finalmente, se elimina del conjunto total de paredes para no poder ser seleccionada de nuevo. Inicialmente, las paredes deben almacenarse en el conjunto de paredes siguiendo el orden de los identificadores de sala, es decir, primero se almacenan las paredes de la sala 0, después las paredes de la sala 1, hasta haber almacenado las paredes de la última sala del mapa. Además, de las 4 paredes posibles que puede tener una sala, se almacenan siguiendo el orden N, E, S, O (Norte, Este, Sur, Oeste). Un subconjunto de las paredes almacenadas sería el siguiente: (0-1)

(0-6) (1-2) (1-7) (1-0) (2-3) (2-8) (2-1) …

4. El proceso de derribo de paredes se repite mientras existan paredes cuyo derribo no ha sido aún valorado. Al final, todas las salas compartirán la misma marca.

0	1	2	3	4	5
6	7	8	9	10	11
12	13	14	15	16	17
18	19	20	21	22	23
24	25	26	27	28	29
30	31	32	33	34	35

Figura 1: Estructura base para el mapa

0	1	2	3	4	5
6	7	8	9	10	11
12	13	8	15	16	17
18	20	20	21	22	23
24	25	26	27	28	29
30	31	32	33	34	35

Figura 2: Mapa tras derribar dos paredes

0	1	2	3	10	5
6	7	13	9	10	11
12	13	13	15	16	17
18	20	20	21	22	23
24	31	26	27	28	29
30	31	32	33	35	35

Figura 3: Proceso de derribo de paredes en el mapa

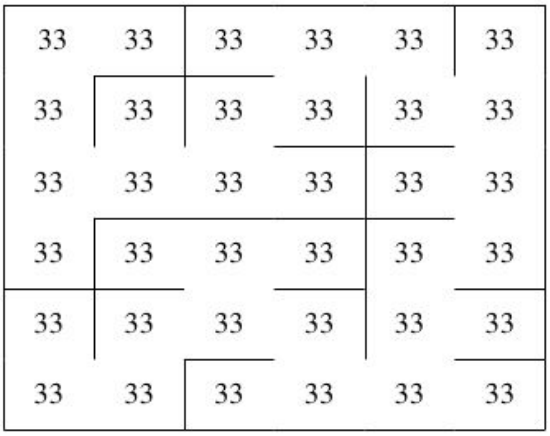

Figura 4: Resultado final del algoritmo

Una vez generada la estructura inicial del mapa, se crearán una serie de atajos en el mismo para garantizar que existan varios caminos en el mapa para ir de una sala a otra. Para la generación de estos atajos dentro del mapa, se derribarán ***n*** paredes del mapa elegidas de forma aleatoria, siendo ***n*** el **5%** del número total de salas del mapa (si hay 100 salas en el mapa, hay que derribar 5 paredes). Al igual que en el algoritmo de Kruskal, las paredes exteriores del mapa no se pueden derribar en este proceso. El proceso de selección de la pared a tirar es el siguiente:

1. Se elige una sala aleatoriamente.
2. Se busca un vecino no accesible (existe pared entre la sala seleccionada (SS) y el vecino), comprobando los vecinos, como muestra la figura 5, en el siguiente orden: Norte, Sur, Oeste y Este.

Figura 5: Orden de selección de vecinos

3. Si se encuentra vecino no accesible y se puede tirar la pared, sin crear espacios vacíos, se tira la pared. Un espacio vacío se forma cuando 4 salas están unidas sin existir ninguna pared entre ellas. Situaciones en las que se crearían espacios vacíos:

Figura 6: Tirar pared entre SS y vecino Norte

Figura 7: Tirar pared entre SS y vecino Sur

Figura 8: Tirar pared entre SS y vecino Oeste

Figura 9: Tirar pared entre SS y vecino Este

4. Si no hay vecino accesible o la pared no se puede tirar, se vuelve al paso 1.

3.- Los personajes

Todos los personajes, además de por su nombre, serán identificados por una marca identificativa (carácter) y deben ser tratados de manera uniforme por el algoritmo de la simulación. Sin embargo, en función del tipo de personaje, las acciones que llevará a cabo cada uno serán diferentes. A continuación se explican estas acciones:

Personajes con poderes físicos:
Estos personajes comenzarán la simulación en la sala 0 y deben realizar las siguientes acciones en cada turno:

1. **Intentar abrir el portal del Hombre Puerta.** Si está en la misma sala en la que está el Hombre Puerta, intentará abrir el portal a través del hombre puerta, tal y como se ha especificado en el documento EC1.
2. **Movimiento**. En esta entrega, todos los personajes se moverán siguiendo una ruta fija predefinida siguiendo el formato de los 4 puntos cardinales. Por ejemplo, si la ruta de un personaje es S-E-N-E-W, ésto indica que el personaje debe moverse en el primer turno a la sala Sur de la sala en la que se encuentra, a continuación a la sala Este de la sala en la que está en ese momento, y así sucesivamente hasta que en el último movimiento se mueva a la sala Oeste de la sala en la que está. La ruta estará preparada para no salirse de los límites del mapa (en esta entrega en un mapa de dimensiones 6x6).
3. **Recoger un arma.** Si hay armas en la sala donde está el personaje, cogerá el arma de la sala con mayor poder, tal y como se ha especificado en el documento EC1.
4. **Interactuar con otros personajes**. Capturará al primer Villano que se encuentre en la

sala sólo si este personaje posee un arma igual a la que posee el villano y tiene mayor poder.

Personajes con poderes extrasensoriales:
Estos personajes comenzarán la simulación en la sala 0 y deben realizar las siguientes acciones en cada turno:

1. **Intentar abrir el portal del Hombre Puerta.** El mismo comportamiento que el que se ha descrito en el personaje anterior.
2. **Movimiento.** En esta entrega, estos personajes se moverán según una ruta fija predefinida, tal y como se ha comentado en el tipo de personaje anterior.
3. **Recoger un arma.** Es el mismo comportamiento que el descrito en el tipo de personaje anterior.
4. **Interactuar con otros personajes**. Es el mismo comportamiento que el descrito en el tipo de personaje anterior.

Personajes con poderes de viaje:
Estos personajes comenzarán la simulación en la esquina inferior izquierda del tablero (esquina SurOeste) y deben realizar las siguientes acciones en cada turno:

1. **Intentar abrir el portal del Hombre Puerta.** El mismo comportamiento que el que se ha descrito en el personaje anterior.
2. **Movimiento.** En esta entrega, estos personajes se moverán según una ruta fija predefinida, tal y como se ha comentado en el tipo de personaje anterior.
3. **Recoger un arma.** Es el mismo comportamiento que el descrito en el tipo de personaje anterior.
4. **Interactuar con otros personajes**. Es el mismo comportamiento que el descrito en el tipo de personaje anterior.

Villanos
Estos personajes comenzarán la simulación en la esquina superior derecha del tablero (esquina Noreste) y deben realizar las siguientes acciones en cada turno:

1. **Intentar abrir el portal del Hombre Puerta.** Si está en la misma sala en la que está el Hombre Puerta, intentará abrir el portal a través del hombre puerta, tal y como se ha especificado en el documento EC1.
2. **Movimiento.** En esta entrega, estos personajes se moverán según una ruta fija predefinida, tal y como se ha comentado en el tipo de personaje anterior.
3. **Recoger un arma.** Si en la sala existe un arma con poder mayor que el arma del villano, el villano intercambiará dichas armas, tal y como se ha especificado en el documento EC1.
4. **Interactuar con otros personajes**. Si el primer superhéroe que se encuentra en la sala tiene su misma arma pero con un poder menor, entonces el superhéroe perderá su arma.

4.- Control de errores en el sistema

Para asegurar el correcto funcionamiento del sistema, en esta entrega se añadirán al sistema los

mecanismos necesarios para controlar posibles situaciones de error en el sistema. Como ejemplo, una posible situación de error en el sistema consiste en intentar crear un mapa con unas dimensiones incorrectas, por ejemplo, negativas. Para llevar a cabo este control de errores, se deben utilizar los mecanismos estudiados hasta ahora en la asignatura.

5.- Primer juego de pruebas unitarias del sistema (opcional en esta entrega, pero recomendable)

También para asegurar el correcto funcionamiento del sistema, en esta entrega se debe comenzar la implementación del juego de pruebas que se utilizará en el sistema. Estas pruebas deben asegurar el correcto funcionamiento de todas las clases que integran el sistema de modo que se desarrolle un conjunto de pruebas incremental (será completado en la cuarta entrega del proyecto). Para implementar estas pruebas, se deben utilizar los mecanismos estudiados hasta ahora en la asignatura.

6.- Registro de resultados en fichero de texto (opcional en esta entrega, pero recomendable)

Para que quede constancia de todo lo que ocurra en el sistema, se ha decidido registrar en un fichero de texto todos los datos de la simulación. A continuación se detalla el contenido que debe tener este fichero de log (el fichero generado debe llamarse *registro.log*).

Estructura del mapa

En primer lugar, al inicio de la simulación, se guardará en este registro el estado de la distribución de las salas del mapa antes de generar los atajos. En el fichero se guardará una representación del mapa similar al que se muestra a continuación:

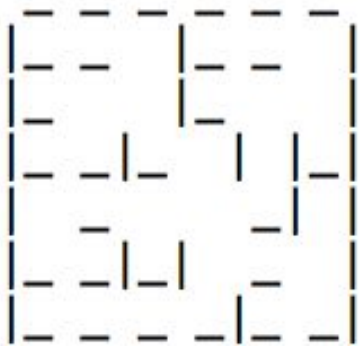

Figura 10: Representación del mapa

Rutas de personajes

A continuación se mostrará la ruta para los personajes existentes en el mapa. El orden en el que se muestran las rutas de los personajes será el mismo que se usa para el turno (según las salas en las que se encuentren los personajes). El formato para mostrar la ruta de un personaje es el siguiente:

(path:<marca del personaje>:<secuencia de orientaciones>)

Ejemplo para personaje con marca S:
(path:S: S S E E N E N E S E S S W S E E)

Ejemplo para personaje con marca T:
(ruta:T: E S S S W S E E N E S S E E)

Para el resto de personajes seguirán el mismo formato. Las rutas mostradas como ejemplo son orientativas, no corresponden al mapa mostrado en este ejemplo. Posteriormente, se facilitarán rutas de ejemplo en el aula virtual de la asignatura para mapas con diferentes dimensiones. Un ejemplo de salida de esta primera parte del fichero se muestra en la Figura 11.

```
 _ _ _ _ _ _
|_ _  |_ _  |
|_    |_    |
|_ _|_  | |_|
|  _     _| |
|_ _|_|  _  |
|_ _ _ _ _|_|
(path:J: E E S S E S S E E S)
(path:D: E E S W W E S W E N E S E S W W W S E W N E E S N E S S W W W E E E E E)
(path:S: E E S S E S S E E S)
(path:A: E E S W W E S W E N E S E S W W W S E W N E E S N E S S W W W E E E E E)
(path:I: E E E N N N W N N N W W E E S S E S E N N E N S W S S W S E E S W W W W W)
(path:T: N W W N E N N E N S W S S W N W N N W W E E S S E S S S W W W E E E E E)
```

Figure 11: Example of map and paths representations

Turno de simulación
Una vez mostrada la estructura inicial del mapa y las rutas de los personajes, se mostrará el estado del sistema en cada turno siguiendo el siguiente formato:

<para cada turno>
(turn:<turno>)
<para el mapa>
(map:<id sala Trono>)
(doorman:<estado>:<altura de apertura>:<armas en el contenedor>)
<pintar mapa 2D incluyendo marca de personajes>
<para cada sala del mapa que contenga armas>
(square:<id square>:<armas>)
<para cada personaje en el mapa según el identificador de sala en el que hayan quedado>
(<tipo de personaje>:<marca>:<id sala actual>:<turno>:<armas>:<villanos capturados (sólo superhéores)>)

Como ejemplo, en la siguiente imagen (Figura 12) puede verse el estado de una posible ejecución del sistema al comienzo de la simulación (turno 0).

```
(turn:0)
(map:35)
(doorman:closed:5:(Acido,1)(Anillo,11)(Antorcha,5)(Armadura,13)(Antorcha,5)(Baston,22)(Bola,3)
(CadenaFuego,11)(CampoEnergia, 5)(CampoMagnetico,5)(Capa,10)(Cetro,20)(Escudo,3)(Espada,11)
(Flecha,12)(Garra,22)(Gema,4))
 _ _ _ _ _ _
|3 _  |_ _ 2|
|_    |_    |
|_ _|_  | |_|
|  _     _| |
|_ _|_|  _  |
|A _ _ _ _ _|
(square:1:(Mjolnir,29)(Garra,27)(Red,25)(Armadura,3)(Anillo,1))
(square:2:(Lucille,23)(GuanteInfinito,21)(LazoVerdad,9)(Lawgiver,7)(Escudo,5))
(square:8:(CadenaFuego,19)(Flecha,17)(Antorcha,15)(Tridente,13)(Capa,11))
(square:14:(Baston,28)(MazaOro,26)(Tentaculo,24)(CampoMagnetico,4)(Latigo,2))
(square:15:(Cetro,22)(Laser,20)(Bola,10)(RayoEnergia,8)(CampoEnergia,6))
(square:21:(Nullifier,23)(Espada,18)(Acido,16)(Gema,14)(Sable,12))
(square:27:(Anillo,29)(Armadura,27)(Red,5)(Garra,3)(Mjolnir,1))
(square:35:(Escudo,25)(Lawgiver,23)(LazoVerdad,21)(GuanteInfinito,9)(Lucille,7))
(square:28:(Antorcha,28)(Capa,19)(Tridente,17)(Flecha,13)(CadenaFuego,11))
(square:29:(Latigo,26)(CampoMagnetico,24)(Baston,15)(Tentaculo,4)(MazaOro,2))
(square:33:(CampoEnergia,22)(RayoEnergia,20)(Bola,18)(Laser,8)(Cetro,6))
(square:34:(Sable,16)(Acido,12)(Espada,10)(Nullifier,3)(Gema,1))
(shphysical:D:0:1:)
(shphysical:R:0:2:)
(shextrasensorial:P:0:1:)
(villain:A:30:2:)
(shflight:E:5:1:)
(shflight:T:5:2:)
```

Figure 12: Estado de una posible ejecución al comienzo de la simulación (turno 0)

Adicionalmente, cuando la simulación termina, se mostrará la siguiente información con respecto a los personajes que han pasado a través del Hombre Puerta.

<para cada personaje que haya pasado a través del Hombre Puerta (el primero es el que abrió el portal y los siguientes han salido porque el portal estaba abierto)>
(teseractomembers)
(owneroftheworld:<tipo de personaje>:<marca>:1111:<turno>:<armas>)
(<tipo de personaje>:<marca>:1111:<turno>:<armas>)
(<tipo de personaje>:<marca>:1111:<turno>:<armas>)

Las líneas entre los símbolos <> de la ilustración anterior representan comentarios que explican el significado del resto de líneas. La sala “1111” es una sala en la que se almacenarían los personajes que han pasado a través del Hombre Puerta y han llegado a la sala del Teseracto. El primer personaje que consigue abrir la puerta es el “owner of the world”.

Ejemplo de salida:

Teniendo en cuenta el formato explicado, a continuación se muestra una posible salida por pantalla que un proyecto podría generar:

```
(turn:12)
(map:35)
(doorman:open:5:(Anillo,11)(Baston,22)(Bola,3)(CampoMagnetico,5)(Espada,11)(Gema,4))
 _ _ _ _ _ _
|D _  |_ _  |
|_   P|_   |
|_ _|A  |E|_|
|  _     _| |
|_ _|_|  _  |
|_ _ _ _ _ _|
(square:0:(Armadura,13)(CampoEnergia,5))
(square:1:(Anillo,11)(Acido,1))
(square:2:(Antorcha,5))
(square:10:(CadenaFuego,19)(LazoVerdad,9))
(square:14:(Espada,18)(Bola,10))
(square:15:(Lawgiver,23)(GuanteInfinito,9))
(shphysical:D:0:12: W)
(shextrasensorial:P:8:12:(Armadura,13)(CampoEnergia,5))
(villain:A:14:12:(CampoEnergia,5))
(shflight:E:16:12:(Acido,1)(Anillo,11)(Antorcha,5))
(teseractomembers)
(owneroftheworld:shphysical:R:1111:12:(Acido,1)(Anillo,11))
(shflight:T:16:12:(Acido,1)(Anillo,11)(Antorcha,5))
```

El ejemplo mostrado anteriormente representa una ejecución con los siguientes datos (datos ficticios, no se corresponden con un ejemplo real):

- La simulación ha finalizado en el turno 12.
- El mapa tiene la sala 35 como sala donde está el Daily Planet (el mapa es de 6x6).
- La puerta secreta ha quedado en estado abierta, la altura para la condición de apertura es 5 y el hombre puerta contiene las siguientes armas: (Anillo,11), (Baston,22), (Bola,3), (CampoMagnetico,5), (Espada,11) y (Gema,4).
- Al final de la ejecución han quedado armas en las siguientes salas: 10, 1, 2, 10, 14, 15.
- Como ejemplo, en la sala 1 quedan las armas: (Acido,1) y (Anillo,11).
- En el mapa han quedado los siguientes personajes:
    - Shphysical D. Ha quedado en la sala 0, su turno es 12, no tiene armas y ha capturado al villano W
    - Shextrasensorial P. Ha quedado en la sala 8, su turno es 12 y tiene las armas (Armadura,13) y (CampoEnergia,5).
    - Villain A. Ha quedado en la sala 14, su turno es 12 y tiene el arma (CampoEnergia,5).
    - Shflight E. Ha quedado en la sala 16, su turno es 12 y tiene las armas (Acido,1), (Anillo,11) y (Antorcha,5).
- Los siguientes personajes han pasado a través de la puerta secreta:
    - Shphysical R. Es el nuevo “owner of the world”. Ha quedado en la sala 1111 (este número de sala es simbólico para representar que ha llegado al Teseracto), su turno

es el 12 y tiene las armas (Acido,1) y (Anillo,11).
    - Shflight T. Ha quedado en la sala 1111, su turno es el 12 y tiene las armas (Acido,1), (Anillo,11) y (Antorcha,5).

Como podemos ver en la ilustración anterior, el mapa ha sido pintado como si fuera una cuadrícula de salas con conexión con sus 4 posibles vecinas. En caso de existir más de un personaje en una misma sala, se representará el número de personajes que hay, en lugar de sus marcas

Consideraciones

- Para **comprobar** el correcto funcionamiento del proyecto, el apéndice de este documento contiene un ejemplo del método principal que inicializa los diferentes personajes y estructuras de la simulación.
- Los estudiantes deben **probar** el proyecto con otras configuraciones y no solo con la utilizada en el ejemplo del apéndice. En ese sentido, los estudiantes pueden usar el **campus virtual** para discutir con sus compañeros los **resultados** obtenidos.

SEGUNDA ENTREGA DE LA EVALUACIÓN CONTINUA (EC2)

Consideraciones:

- El proyecto entregado no debe contener archivos binarios, sólo debe contener los archivos fuente y aquellos que son estrictamente necesarios para ejecutar el proyecto.
- El programa debe ejecutarse de principio a fin sin requerir la intervención del usuario.
- Los estudiantes deben hacer un uso correcto de los conceptos de **herencia y polimorfismo**.
- Además, los estudiantes deben seguir todas las mismas restricciones y requisitos especificados en la primera entrega.

La entrega de esta fase será: **Lunes, 27 de noviembre**.

*ANEXO. Ejemplo de código main que inicializa los diferentes objetos y pone en marcha la simulación del sistema (este código es **meramente orientativo**).*

```
import java.util.ArrayList;
import java.util.List;

/**
* Enum type that represents 4 possible directions for the characters' movements
* It should be defined into a separated file
*/
enum Dir {S, E, N, W};

/**
* D1 - 17_18 Project
* Implementation for the Map class
* @version 3.0
* @author
* <b> DP teachers </b><br>
* Program Development Subject<br/>
* Course 17/18
*/
public class Map {
        /**
        * Main program - D2.
        * @param args parameters for the main
        * @return Return the output of the program
        */
  public static void main (String args[]) {
      int dimX = 6;
      int dimY = 6;
      int dailyPlanetSquare = (dimX * dimY) – 1;
      int initialDepth = 4;
      int MAXTURNS = 50;
      // Creating the map
      // Parameters: dailyPlanet square, columns number, rows number,
```

```
// Initial depth for the lock
// The constructor must create the different squares for the map
Map map = new Map(dailyPlanetSquare, dimX, dimY, initialDepth);

// Generate the weapons and distribute them. In this stage, we pass to the map an array
// with the identifiers of the squares where the weapons are going to be distributed
// it was specified in the previous stage
int [] idSquaresWithWeapons =  {1, 2, 8, 14, 15, 21, 27, 35, 28, 29, 33, 34};
map.distributeWeapons (idSquaresWithWeapons);

// Creating and configuring the DoorMan character. It is not specified here since
// it was specified in the previous stage
DoorMan doorMan = new DoorMan(initialDepth);
map.addDoorMan(doorMan);

// Creating the characters
// Creating a SHPHYSICAL
// Parameters: name, mark, turn in which it will start the simulation and initial square
ShPhysical shPhDare = new ShPhysical("Daredevil", 'D', 1, 0);
// Creating the route for the SHPHYSICAL:
// (route:D: E E S S E S S E E S)
LinearDS<Dir> directionsDare = {Dir.E, Dir.E, Dir.S, Dir.S, Dir.E, Dir.S, Dir.S, Dir.E, Dir.E, Dir.S};
shPhDare.asignRoute(directionsDare);
// Adding the character into the map
map.addCharacter(shPhDare);

// Creating a SHEXTRASENSORIAL
// Parameters: name, mark, turn in which it will start the simulation and initial square
ShExtrasensorial shExProf = new ShExtrasensorial("ProfessorX", 'P', 1, 0);
// (route:P: E E S W W E S W E N E S E S W W W S E W N E E S N E S S W W W E E E E E)
LinearDS<Dir> directionsProf = {Dir.E, Dir.E, Dir.S, Dir.W, Dir.W, Dir.E, Dir.S, Dir.W, Dir.E,
                                Dir.N, Dir.E, Dir.S, Dir.E, Dir.S, Dir.W, Dir.W, Dir.W, Dir.S, Dir.E
                                Dir.W , Dir.N, Dir.E,, Dir.E, Dir.S, Dir.N, Dir.E, Dir.S, Dir.S, Dir.W
                                Dir.W, , Dir.W, Dir.E, Dir.E, Dir.E, Dir.E, Dir.E};
shExProf.asignRoute(directionsProf);
// Adding the character into the map
map.addCharacter(shExProf);

// Creating a SHFLIGHT
// Parameters: name, mark, turn in which it will start the simulation and initial square
ShFlight  shFli  = new ShFlight("Eternity", 'F', 1, map.getSouthWestCorner());
// (route:F: E E E N E E S)
LinearDS<Dir> directionsFli = {Dir.E, Dir.E, Dir.E, Dir.N, Dir.E, Dir.E, Dir.S};
shFli.asignRoute(directionsFli);
// Adding the character into the map
map.addCharacter(shFli);
```

```
        // Creating a Villain
        // Parameters: name, mark, turn in which it will start the simulation and initial square
       Villain villainAb  = new Villain("Abomination", 'A', 1, map.getNorthEastCorner());
        // (ruta:A: S S N W S S W S E E N S S)
        LinearDS<Dir> directionsA = {Dir.S, Dir.S, Dir.N, Dir.W, Dir.S, Dir.S, Dir.W, Dir.S, Dir.E, Dir.E,
                                     Dir.N, Dir.S, Dir.S};
        villainAb.asignRoute(directionsA);
        // Adding the character into the map
        map.addCharacter(villainAb);

        map.paint();
        // Executing the simulation
        // The process method must be executed turn after turn, traversing the map from square 0
        // to the last square and the characters stored in each square must execute their actions
        // in a chronologically order (the characters that arrived first are the first in leaving the square)
        for (int i=0; i<MAXTURNS;i++) {
              map.process(i);
         }

        map.paint();
    }
}
```