PROGRAMACIÓN II

PROYECTO

MundoEncantado

versión: 2.0.0

Grado en Ingeniería de Tecnologías de Telecomunicación

Curso 2019/20

HISTORIAL DE CAMBIOS

Versión	Cambios
2.0.0	Versión para la segunda oportunidad. Se incorpora un nuevo tipo de criaturas, las lamias, en el MundoEncantado. Se añaden 3 nuevas instrucciones (instrucciones 10, 14 y 15) para el modo 2 de funcionamiento (ejecución de instrucciones). Los cambios principales respecto a la versión de la primera oportunidad aparecen en rojo.
1.0.0	Versión inicial del enunciado del proyecto.

1. INTRODUCCIÓN

El objetivo del proyecto de la asignatura Programación II en el presente curso es desarrollar una aplicación para jugar a un juego denominado MundoEncantado. Este juego, en pocas palabras, es una simplificación de un juego de rol en el que criaturas fantásticas asignadas a jugadores se enfrentan en una secuencia de batallas.

En el presente documento se detallan, en el Apartado 2, las características y reglas del juego MundoEncantado y, en el Apartado 3, las particularidades de la aplicación a desarrollar. En el Apartado 4 se describen las normas de entrega del proyecto y el método de evaluación del mismo.

ATENCIÓN: *Con el fin de mantener una visión uniforme sobre la funcionalidad de la aplicación a desarrollar en el proyecto y de que todos los alumnos tengan la misma información respecto al mismo, cualquier duda que surja relativa a la interpretación de la presente especificación deberá ser planteada en el foro de consultas de FaiTIC destinado a tal efecto. Los profesores de la asignatura no atenderán a cuestiones de este tipo de manera individual.*

2. DESCRIPCIÓN DEL JUEGO

2.1 Criaturas

El **MundoEncantado** es un juego que contempla la existencia de diferentes criaturas mitológicas, cada una de ellas caracterizada por un ID (identificador único en el MundoEncantado), su nombre y su estado de salud (un valor entre 0 y 10). Una de estas criaturas puede atacar a otra criatura, que a su vez puede defenderse de la primera. El resultado principal de un proceso de ataque/defensa será la merma de salud de las criaturas implicadas, tanto del atacante como del defensor. Si la salud de una criatura desciende hasta 0 (valor mínimo posible), la criatura queda neutralizada. La merma de salud dependerá fundamentalmente del poder ofensivo de la atacante y de la capacidad defensiva de la defensora, aunque el propio hecho de atacar y defender supone un desgaste en la criatura que se refleja en una disminución de su salud. En concreto, **la salud de la defensora se verá reducida en un valor igual a la diferencia entre el poder ofensivo de la atacante y la capacidad defensiva de la defensora (siempre y cuando esta diferencia sea positiva)**. La reducción de salud por desgaste dependerá de la naturaleza de cada una de las criaturas que participan en la lucha. Un enfrentamiento en el que participa una criatura neutralizada no produce ningún efecto en las criaturas enfrentadas.

Tras una batalla (una secuencia de luchas), algunas criaturas, si están activas, se pueden recuperar ligeramente yendo o bien al **LagoSagrado** o bien al **TemploMaldito**. Una criatura puede ser **UsuarioLagoSagrado** o **UsuarioTemploMaldito** (o ninguna de las dos cosas), pero no ambas cosas cosas a la vez. Un UsuarioLagoSagrado puede bañarse en el LagoSagrado y un UsuarioTemploMaldito puede orar en el TemploMaldito después de una batalla para recuperar algo de salud e incluso parte de sus capacidades y atributos. Tanto en el LagoSagrado como en el TemploMaldito se mantiene un libro de visitas que registra cada una de las asistencias al lugar.

Existen 5 tipos de criaturas en el MundoEncantado, cada una de ellas con habilidades particulares y con un mecanismo de ataque y otro de defensa específico. En concreto:

- **Orcos**: son criaturas dotadas de fuerza (un atributo variable entre 1 y 10) que poseen un garrote para atacar (con un nivel ofensivo que varía entre 0 y 90) y un escudo para defenderse (con un nivel defensivo entre 0 y 90). El poder ofensivo y la capacidad defensiva de un orco vienen determinados por las siguientes relaciones:
    - poder ofensivo = (fuerza + garrote)/5[1]
    - capacidad defensiva = (fuerza + escudo)/20

  Tras una lucha, un orco reduce su salud en 1 unidad por desgaste atacante y en 3 unidades por desgaste defensivo. Asimismo, reduce en 3 unidades su garrote por ataque y en 3 unidades su escudo por defensa. Un orco es usuario del TemploMaldito, lugar al que va a orar tras una batalla si no está

[1] Para el cálculo del poder ofensivo y la capacidad defensiva de cualquier criatura siempre se considerarán divisiones enteras. Por ejemplo, si un orco tiene fuerza = 10 y garrote = 68, su poder ofensivo será 15.

neutralizado. La oración produce un nivel de mejora en el orco que dependerá del número de veces que haya visitado el templo según la siguiente relación:

- garrote: su nivel aumenta 2*nVisitas unidades (donde nVisitas es el número de veces que ha visitado el templo)[2].
- escudo: su nivel aumenta nVisitas unidades.

- **Brujas**: criaturas con sabiduría (entre 1 y 5) y magia (entre 1 y 5). Poseen un bastón mágico (valorado entre 1 y 10) y un vestido confeccionado con escamas de dragón como mecanismo de defensa (de un nivel entre 1 y 10). El poder ofensivo y la capacidad defensiva de una bruja vienen determinados por las siguientes relaciones:
  - poder ofensivo = (sabiduría + magia) * bastón / 5
  - capacidad defensiva = (sabiduría + magia) * vestido / 10

  Las brujas, tras una lucha, ven reducida su salud en 2 unidades por desgaste, tanto ofensivo como defensivo, y en 1 unidad su bastón al atacar y su vestido al defender. Al igual que los orcos, son usuarias del TemploMaldito, donde una oración produce los siguientes incrementos en sus armas:

  - bastón mágico: (2 + nVisitas) / 2 [3]
  - vestido: nVisitas/2

- **Elfos**: son criaturas con inteligencia (valorada entre 1 y 5) que poseen un arco (de un nivel entre 2 y 5) y una coraza metálica (con valor entre 1 y 5). Su poder ofensivo y su capacidad defensiva vienen determinados por las siguientes relaciones:
  - poder ofensivo = (inteligencia * arco^2) / 5
  - capacidad defensiva = (inteligencia * coraza^2) / 10

  Su salud se reduce en 3 unidades por desgaste en una lucha, así como sufren una reducción en un enfrentamiento de 1 unidad en sus armas (arco y coraza), independientemente de que estén atacando o defendiendo. Para recuperar su salud, tras una batalla, si no están neutralizadas, acceden al LagoSagrado, del que son usuarios, donde su salud se incrementa en 3*nVisitas unidades y sus armas en nVisitas/2.

- **Ninfas**: dotadas de divinidad (entre 0 y 2), velocidad (entre 0 y 2) y capacidad de engaño (entre 0 y

[2] Tenga en cuenta que no se deben superar los rangos definidos para los atributos de las criaturas. Si, por ejemplo, el nivel de garrote está a 90 (valor máximo), la oración en el TemploMaldito no producirá un aumento en este nivel. Del mismo modo, si, por ejemplo, el nivel de garrote está a 1, el desgaste por ataque dejará el garrote a 0 (no a -2).
[3] División entera.

1), estas criaturas poseen una varita mágica y una armadura de cuero (con un nivel ofensivo y defensivo entre 0 y 1000 ambas). Su poder ofensivo y su capacidad defensiva vienen determinadas por las siguientes relaciones:

- poder ofensivo = divinidad * (velocidad + engaño) + varita/100
- capacidad defensiva = divinidad * (velocidad + engaño) + armadura/200

Las ninfas reducen su salud en 2 unidades tras un combate simplemente por desgaste y su varita en 5 unidades al atacar y su armadura en 5 unidades al defender. Sin embargo, pueden recuperar parte de la misma bañándose en el LagoSagrado, donde su salud se incrementa en 2+nVisitas*2 unidades y sus armas (varita y armadura) en nVisitas*3 unidades.

- **Lamias**: dotadas de astucia (variable entre 1 y 10) y seducción (variable entre 1 y 10). Atacan con sus garras (con un nivel entre 0 y 100) y se defienden con un mitón mágico (con un nivel entre 0 y 100). El poder ofensivo y la capacidad defensiva de una lamia vienen determinados por las siguientes relaciones:
  - poder ofensivo = (astucia + garras) / 5
  - capacidad defensiva = (seducción + mitón) / 20

  Tras una lucha, una lamia reduce su salud en 1 unidad por desgaste atacante y en 3 unidades por desgaste defensivo. Asimismo, reduce en 5 unidades sus garras en un ataque y en 3 unidades su mitón por defensa. **Una lamia no es usuaria ni del TemploMaldito ni del LagoSagrado**.

2.2 Desarrollo de una partida

En el juego MundoEncantado participan entre 2 y 4 jugadores organizados secuencialmente (desde jug_1 a jug_4). Una partida consta de una serie de batallas en las que los jugadores van consiguiendo puntos. Una partida finaliza cuando al menos un jugador alcanza los 10 puntos.

Inicialmente, las criaturas existentes en el MundoEncantado están disponibles en el **Bosque.** Una batalla consta de los siguientes pasos:

1. Al iniciar una batalla cada jugador recibe 3 criaturas de manera aleatoria entre todas las existentes en el Bosque. Si no hubiese suficientes criaturas, se repartirá equitativamente las que haya entre los jugadores (por ejemplo, si hay 10 criaturas y juegan 4 jugadores, cado uno recibiría 2 criaturas). En caso de que no hubiese al menos una criatura para cada jugador, la partida no podrá celebrarse.
2. Los jugadores que hubiesen recibido sólo criaturas neutralizadas serán jugadores inactivos en la batalla, y no tomarán parte en las luchas.
3. Una vez realizada la asignación de criaturas, se producirán sucesivas luchas entre criaturas del siguiente modo (estrictamente):
   a. El jugador que le toca ser atacante (el jugador atacante irá rotando entre los jugadores

activos en la partida, siendo el primero el jug_1) elegirá una de sus criaturas, que se enfrentará con una criatura del jugador defensor. Escogerá, de entre las que tiene asignadas, y no están neutralizadas, la criatura con mayor poder ofensivo (en caso de empate, la de menor ID de entre las que tienen mayor poder ofensivo).

b. El jugador defensor será el primero que esté activo de los siguientes al atacante. Por ejemplo, suponiendo que hay 3 jugadores, si el jugador atacante es el jug_2, el defensor será el siguiente, es decir, el jug_3, siempre y cuando este estuviese activo (si el jug_3 estuviera inactivo, el defensor sería el jug_1 si estuviese activo).
c. El jugador defensor elegirá, de entre sus criaturas no neutralizadas, la criatura defensora. En concreto seleccionará la de mayor capacidad defensiva (en caso de empate, la de menor ID de entre las que tienen mayor capacidad defensiva).
d. Definidas las criaturas atacante y defensora se establece una lucha entre ellas, que provocará cambios en su salud y en determinados atributos, tal como se describió en el Apartado 2.1.

Las luchas en la batalla proseguirán secuencialmente hasta que se cumpla una de estas dos condiciones:

a. Quede, como máximo, un jugador activo. El jugador activo será el vencedor de la batalla y recibirá 2 puntos por cada criatura no neutralizada que le quedan. Si no quedan jugadores activos, ninguno recibirá puntos.
b. Se alcancen 10 luchas. En este caso, los jugadores activos recibirán 1 punto por cada criatura no neutralizada que le quedan.

4. Tras la secuencia de luchas que conforman la batalla, **las criaturas que no han sido neutralizadas** visitan el LagoEncantado o el TemploMaldito, según corresponda, para recuperarse.
5. Finalmente, **todas las criaturas** (activas y neutralizadas) que han participado en la batalla regresan al Bosque.

Una vez finalizada una batalla comenzará la siguiente, salvo que todas las criaturas del Bosque hayan quedado neutralizadas, o bien algún jugador haya alcanzado los 10 puntos. En ese caso, la partida finaliza y, si existe un jugador con más puntos que los demás, ese será el vencedor de la partida.

3. APLICACIÓN MundoEncantado

Los alumnos deberán desarrollar una aplicación Java siguiendo el paradigma de Programación Orientada a Objetos (POO) que implemente el juego MundoEncantado descrito en el apartado anterior. Esta aplicación funcionará en modo CLI (Command-Line Interface), es decir, de manera análoga a un comando en línea de Linux, con un comportamiento variable que dependerá de los parámetros utilizados en su invocación.

```
$> java MundoEncantado [parámetros]
```

En la aplicación se utilizarán, entre otros, un fichero en el que se define el conjunto de criaturas disponibles inicialmente en el Bosque, y un fichero en el que se definen los jugadores que participan en la partida. Los ficheros serán conformes al formato definido en los apéndices A.1 y A.2, respectivamente.

La aplicación dispondrá de 2 modos de funcionamiento: i) juego normal y ii) ejecución de instrucciones. En los siguientes sub-apartados se describen las particularidades de cada modo de funcionamiento.

3.1 Modo 1: Juego normal

3.1.1 Invocación

```
$> java MundoEncantado -j f_jugadores -c f_criaturas [-r f_reparto][-o
f_partida]
```

3.1.2 Funcionalidad

En este modo, la aplicación desarrollará una partida de la manera descrita en el Apartado 2. Los jugadores participantes en dicha partida serán los enumerados en el fichero "f_jugadores", especificado en el parámetro de invocación `-j f_jugadores`, y las criaturas que pueblan el Bosque serán las enumeradas en el fichero "f_criaturas", especificado en el parámetro `-c f_criaturas`. En el fichero "f_jugadores" se enumeran, de manera ordenada, los jugadores participantes (el que aparece en la primera línea será el jug_1, el de la segunda el jug_2, y así sucesivamente).

Si en la invocación de la aplicación se utiliza el parámetro opcional `-r f_reparto`, el reparto de criaturas entre los jugadores al principio de cada batalla dejará de ser aleatorio[4]. En este caso, las criaturas que se asignan a cada jugador en cada batalla vendrán determinadas por el contenido del fichero "f_reparto", cuyo formato se describe en el Apéndice A.3. En este fichero, cada línea define una asignación de criaturas para una batalla. Si el fichero no incluye tantas líneas como batallas ocurren en una partida, después de la última asignación se volverá a la primera línea. Por ejemplo, si en una partida ocurren 8 batallas, pero el fichero de reparto solo incluye 6 líneas (6 asignaciones de criaturas), las dos últimas batallas utilizarán el reparto definido en las dos primeras líneas del fichero.

El resultado de la partida se volcará en el fichero "f_partida", siempre y cuando se haya utilizado el parámetro

[4] En concreto, el reparto aleatorio indicado en amarillo en el Apartado 2.2.

opcional `-o f_partida`, o por salida estándar si este no hubiese sido establecido. En ambos casos, la salida deberá adherirse al formato especificado en el Apéndice A.5.

3.1.3 Ejemplo

A continuación se presenta un ejemplo de invocación del juego en modo normal que utiliza la información de jugadores contenida en el fichero “jugadores.txt” y de criaturas definidas en el fichero “criaturas.txt”. En este caso se llevan a cabo el reparto de criaturas de manera aleatoria y se muestra el resultado de la partida por salida estándar (pantalla).

```
$> java MundoEncantado -j jugadores.txt -c criaturas.txt

BATALLA_1 jA:{(b1,10),(n1,10)},jB:{(e0,10),(o1,10)},jC:{(b0,10),(o0,10)}
  LUCHA 1: jA-N:{n1,Belgica,D0,V874,R2,E1,A270} --> jB-E:{e0,Eurico,I2,A4,C4}
  jA:{(b1,10),(n1,8)},jB:{(e0,2),(o1,10)},jC:{(b0,10),(o0,10)}
  LUCHA 2: jB-O:{o1,Thrum,F2,G54,E31} --> jC-O:{o0,Grom,F1,G5,E45}
  jA:{(b1,10),(n1,8)},jB:{(e0,2),(o1,9)},jC:{(b0,10),(o0,0)}
  LUCHA 3: jC-B:{b0,Bellatrix,S1,M2,B1,V6} --> jA-N:{n1,Belgica,D0,V869,R2,E1,A270}
  jA:{(b1,10),(n1,6)},jB:{(e0,2),(o1,9)},jC:{(b0,8),(o0,0)}
  LUCHA 4: jA-N:{n1,Belgica,D0,V869,R2,E1,A265} --> jB-E:{e0,Eurico,I2,A4,C3}
  jA:{(b1,10),(n1,4)},jB:{(e0,0),(o1,9)},jC:{(b0,8),(o0,0)}
  LUCHA 5: jB-O:{o1,Thrum,F2,G51,E31} --> jC-B:{b0,Bellatrix,S1,M2,B1,V6}
  jA:{(b1,10),(n1,4)},jB:{(e0,0),(o1,8)},jC:{(b0,0),(o0,0)}
  LUCHA 6: jA-N:{n1,Belgica,D0,V864,R2,E1,A265} --> jB-O:{o1,Thrum,F2,G48,E31}
  jA:{(b1,10),(n1,2)},jB:{(e0,0),(o1,0)},jC:{(b0,0),(o0,0)}

  FIN BATALLA: Solo hay un jugador activo.
  PUNTOS CONSEGUIDOS: jA=4,jB=0,jC=0

BATALLA_2 jA:{(b1,10),(o1,0)},jB:{(b0,0),(n0,10)},jC:{(n1,6),(o0,0)}
  LUCHA 1: jA-B:{b1,Aughra,S1,M2,B3,V2} --> jB-N:{n0,Albania,D1,V463,R1,E0,A142}
  jA:{(b1,8),(o1,0)},jB:{(b0,0),(n0,8)},jC:{(n1,6),(o0,0)}
  LUCHA 2: jB-N:{n0,Albania,D1,V463,R1,E0,A137} --> jC-N:{n1,Belgica,D0,V862,R2,E1,A268}
  jA:{(b1,8),(o1,0)},jB:{(b0,0),(n0,6)},jC:{(n1,0),(o0,0)}
  LUCHA 3: jA-B:{b1,Aughra,S1,M2,B2,V2} --> jB-N:{n0,Albania,D1,V458,R1,E0,A137}
  jA:{(b1,6),(o1,0)},jB:{(b0,0),(n0,4)},jC:{(n1,0),(o0,0)}
  LUCHA 4: jB-N:{n0,Albania,D1,V458,R1,E0,A132} --> jA-B:{b1,Aughra,S1,M2,B1,V2}
  jA:{(b1,0),(o1,0)},jB:{(b0,0),(n0,2)},jC:{(n1,0),(o0,0)}

  FIN BATALLA: Solo hay un jugador activo.
  PUNTOS CONSEGUIDOS: jA=0,jB=2,jC=0

BATALLA_3 jA:{(e0,0),(e1,10)},jB:{(b1,0),(o1,0)},jC:{(b0,0),(n1,0)}

  FIN BATALLA: Solo hay un jugador activo.
  PUNTOS CONSEGUIDOS: jA=2,jB=0,jC=0

BATALLA_4 jA:{(n0,6),(o1,0)},jB:{(b0,0),(b1,0)},jC:{(e0,0),(e1,10)}
  LUCHA 1: jA-N:{n0,Albania,D1,V456,R1,E0,A135} --> jC-E:{e1,Alarico,I1,A3,C3}
  jA:{(n0,4),(o1,0)},jB:{(b0,0),(b1,0)},jC:{(e0,0),(e1,2)}
  LUCHA 2: jC-E:{e1,Alarico,I1,A3,C2} --> jA-N:{n0,Albania,D1,V451,R1,E0,A135}
  jA:{(n0,2),(o1,0)},jB:{(b0,0),(b1,0)},jC:{(e0,0),(e1,0)}

  FIN BATALLA: Solo hay un jugador activo.
  PUNTOS CONSEGUIDOS: jA=2,jB=0,jC=0

BATALLA_5 jA:{(b0,0),(n0,8)},jB:{(e0,0),(o0,0)},jC:{(e1,0),(o1,0)}

  FIN BATALLA: Solo hay un jugador activo.
  PUNTOS CONSEGUIDOS: jA=2,jB=0,jC=0
```

```
VISITAS A LOS LUGARES SAGRADOS:
  Lago Sagrado: {e1=1, n0=3, n1=1}
  Templo Maldito: {b1=1}

PUNTUACIONES:
  Pitufo Goloso (jA) = 10 (VENCEDOR)
  Bebe Pitufo (jB) = 2
  Pitufina (jC) = 0
```

Como se puede observar, el resultado mostrado describe la secuencia de batallas ocurridas en la partida y, para cada una de ellas, las luchas que tienen lugar. Tras la descripción de las batallas que conforman la partidas, se muestran las visitas recibidas por los lugares sagrados (el Lago Sagrado y el Templo Maldito). Finalmente aparecen los puntos obtenidos por cada jugador. El formato concreto de esta información se describe de manera detallada en el Apéndice A.5.

3.1.4 Gestión de errores

En las pruebas que se realicen a la aplicación MundoEncantado, los ficheros con información de entrada siempre serán correctos, tanto sintácticamente (es decir, cumplirán estrictamente el formato definido en los apéndices) como semánticamente (es decir, la información en ellos será siempre coherente). Por tanto, la aplicación no considerará, necesariamente, errores de parseado de ficheros.

De manera análoga, la invocación de la aplicación siempre se realizará de manera consistente en las pruebas: los parámetros siempre serán utilizados de manera correcta.

3.2 Modo 2: Ejecución de instrucciones

3.2.1 Invocación

```
$> java MundoEncantado -i f_instrucciones [-o f_salida]
```

3.2.2 Funcionalidad

En el modo de ejecución de instrucciones la aplicación no juega una partida, sino que lleva a cabo una serie de instrucciones o comandos predefinidos. Las instrucciones a ejecutar por parte de la aplicación se almacenan en el fichero especificado en el parámetro de invocación: `-i f_instrucciones`. En este fichero, cada instrucción ocupa una única línea, y sólo puede haber una instrucción por línea. Las líneas que

comiencen por el carácter "#" se consideran comentarios y se ignoran. Las líneas en blanco también se ignoran. Los resultados de la ejecución se presentan en la salida estándar, a no ser que se especifique el parámetro de invocación `-o f_salida`, en cuyo caso los resultados se depositan en el fichero "f_salida". En caso de que el fichero especificado ya exista en el directorio desde el que se invoca la aplicación, éste se sobreescribirá.

Las instrucciones que puede ejecutar la aplicación son las siguientes:

Instr 1	`CargarJugadores <fichero_jugadores>`
descripción	Carga en memoria la información de jugadores contenida en el fichero <fichero_jugadores> y se añade a la existente en la sesión. Si en esta sesión ya existían jugadores con el mismo ID, se sobreescriben los jugadores.
errores	• Si no se puede leer el fichero especificado se produce un error.

salida	En caso de éxito, se genera en una línea la expresión:
	`CargarJugadores <fichero_jugadores>: OK.`
	En caso de error, se genera la expresión:
	`CargarJugadores <fichero_jugadores>: FAIL.`

ejemplo	`CargarJugadores players.txt`

Instr 2	`CrearJugador <id> <nombre>`
descripción	Se crea un nuevo jugador con identificador <id> y nombre <nombre>.
errores	• En caso de que ya exista un jugador con el <id> especificado en la sesión se considerará un error.

salida	En caso de éxito, se genera en una línea la expresión:
	
	`CrearJugador <id> <nombre>: OK.`
	
	En caso de error, se genera la expresión:
	
	`CrearJugador <id> <nombre>: FAIL.`

ejemplo	`CrearJugador 432 Johann Sebastian Bach`

Instr 3	`BorrarJugador <id>`
descripción	Se elimina el jugador con identificador <id>.
errores	• En caso de que no exista un jugador con el identificador <id> especificado porque no se había creado previamente en esta sesión, se genera un error.

salida	En caso de éxito, se genera la expresión:
	
	`BorrarJugador <id>: OK.`
	
	En caso de fallo, la expresión:
	
	`BorrarJugador <id>: FAIL.`

ejemplo	`BorrarJugador 42`

Instr 4	`VolcarJugadores <fichero_jugadores>`
descripción	Se genera un fichero <fichero_jugadores> con el formato descrito en el Apéndice A.2, donde se almacenan todos los jugadores creados en esta sesión. **Se almacenarán por orden de creación**. Si ya existía un fichero <fichero_jugadores>, se sobreescribe con la nueva

	información.
errores	• Si no se puede escribir el fichero especificado se produce un error.

salida	En caso de éxito, se genera la expresión:
	`VolcarJugadores <fichero_jugadores>: OK.`
	En caso de fallo, la expresión:
	`VolcarJugadores <fichero_jugadores>: FAIL.`

ejemplo	`VolcarJugadores jugadores.txt`

Instr 5	`CargarCriaturas <fichero_criaturas>`
descripción	Incorpora al Bosque las criaturas definidas en el fichero <fichero_criaturas>. Si en el Bosque ya existían criaturas con el mismo ID, se sobreescriben.
errores	• Si no se puede leer el fichero especificado se produce un error.

salida	En caso de éxito, se genera en una línea la expresión:
	`CargarCriaturas <fichero_criaturas>: OK.`
	En caso de error, se genera la expresión:
	`CargarCriaturas <fichero_criaturas>: FAIL.`

ejemplo	`CargarCriaturas creatures.txt`

Instr 6	`CrearNinfa <id> <nombre> <divinidad> <velocidad> <engaño> <varita> <armadura>`
descripción	Se crea una nueva ninfa con los parámetros especificados y se incorpora al Bosque.
errores	• En caso de que ya exista una criatura con el <id> especificado en el Bosque se considerará un error.

salida	En caso de éxito, se genera en una línea la expresión:
	
	`CrearNinfa <id> <nombre> <divinidad> <velocidad> <engaño> <varita> <armadura>: OK.`
	
	En caso de error, se genera la expresión:
	
	`CrearNinfa <id> <nombre> <divinidad> <velocidad> <engaño> <varita> <armadura>: FAIL.`

ejemplo	`CrearNinfa n13 Africa 2 0 1 354 514`

Instr 7	`CrearOrco <id> <nombre> <fuerza> <garrote> <escudo>`
descripción	Se crea un nuevo orco con los parámetros especificados y se incorpora al Bosque.
errores	• En caso de que ya exista una criatura con el <id> especificado en el Bosque se considerará un error.

salida	En caso de éxito, se genera en una línea la expresión:
	
	`CrearOrco <id> <nombre> <fuerza> <garrote> <escudo>: OK.`
	
	En caso de error, se genera la expresión:
	
	`CrearOrco <id> <nombre> <fuerza> <garrote> <escudo>: FAIL.`

ejemplo	`CrearOrco o6 Karg 10 32 68`

Instr 8	`CrearBruja <id> <nombre> <sabiduria> <magia> <baston> <vestido>`
descripción	Se crea una nueva bruja con los parámetros especificados y se incorpora al Bosque.
errores	• En caso de que ya exista una criatura con el <id> especificado en el Bosque se considerará un error.

salida	En caso de éxito, se genera en una línea la expresión:
	
	`CrearBruja <id> <nombre> <sabiduria> <magia> <baston> <vestido>: OK.`
	
	En caso de error, se genera la expresión:
	
	`CrearBruja <id> <nombre> <sabiduria> <magia> <baston> <vestido>: FAIL.`

ejemplo	`CrearBruja b13 Malefica 2 3 7 6`

Instr 9	`CrearElfo <id> <nombre> <inteligencia> <arco> <coraza>`
descripción	Se crea un nuevo elfo con los parámetros especificados y se incorpora al Bosque.
errores	• En caso de que ya exista una criatura con el <id> especificado en el Bosque se considerará un error.

salida	En caso de éxito, se genera en una línea la expresión:
	
	`CrearElfo <id> <nombre> <inteligencia> <arco> <coraza>: OK.`
	
	En caso de error, se genera la expresión:
	
	`CrearElfo <id> <nombre> <inteligencia> <arco> <coraza>: FAIL.`

ejemplo	`CrearElfo e0 Eurico 2 4 5`

Instr 10	`CrearLamia <id> <nombre> <astucia> <seduccion> <garras> <miton>`
descripción	Se crea una nueva lamia con los parámetros especificados y se incorpora al Bosque.
errores	• En caso de que ya exista una criatura con el <id> especificado en el Bosque se considerará un error.

salida	En caso de éxito, se genera en una línea la expresión:
	
	`CrearLamia <id> <nombre> <astucia> <seduccion> <garras> <miton>: OK.`
	
	En caso de error, se genera la expresión:
	
	`CrearLamia <id> <nombre> <astucia> <seduccion> <garras> <miton>: FAIL.`

ejemplo	`CrearLamia l30 Lilith 5 7 90 70`

Instr 11	`BorrarCriatura <id>`
descripción	Se elimina del Bosque la criatura con identificador <id>.
errores	• En caso de que no exista una criatura con el identificador <id> especificado en el Bosque, porque no se había creado previamente en esta sesión, se genera un error.

salida	En caso de éxito, se genera la expresión:
	
	`BorrarCriatura <id>: OK.`
	
	En caso de fallo, la expresión:
	
	`BorrarCriatura <id>: FAIL.`

ejemplo	`BorrarCriatura n13`

Instr 12	`VolcarCriaturas <fichero_criaturas>`
descripción	Se genera un fichero <fichero_criaturas> con el formato descrito en el Apéndice A.1, donde se almacenan todas las criaturas existentes en el Bosque. **Se almacenan por orden de identificador**. Si ya existía un fichero <fichero_criaturas>, se sobreescribe con la nueva información.
errores	• Si no se puede escribir el fichero especificado se produce un error.

salida	En caso de éxito, se genera la expresión:
	`VolcarCriaturas <fichero_criaturas>: OK.`
	En caso de fallo, la expresión:
	`VolcarCriaturas <fichero_criaturas>: FAIL.`

ejemplo	`VolcarCriaturas criaturas.txt`

Instr 13	`MostrarCriatura <id>`
descripción	Muestra los datos de la criatura con ID <id>, con el formato: <Tipo>:{<ID>,<Nombre>,<AtributoValor>,<AtributoValor>,...,<PoderOfensivo>,<CapacidadDefensiva>,<Salud>} de manera análoga a lo descrito en el Apéndice A.1, añadiendo al final el valor del poder ofensivo, la capacidad defensiva y la salud de la criatura.
errores	• Si no existe una criatura con el ID especificado en el Bosque se producirá un error.

salida	En caso de éxito, se genera la expresión:
	
	`MostrarCriatura <id>: <descripcion_extendida_criatura>.`
	
	En caso de fallo, la expresión:
	
	`MostrarCriatura <id>: FAIL.`

ejemplo	`MostrarCriatura orco10`
	
	Presentaría el resultado, en caso de éxito:
	
	`MostrarCriatura orco10: O:{orco10,Gorrum,F5,G80,E16,17,1,9}`

Instr 14	`CriaturaMasOfensiva`
descripción	Muestra los datos de la criatura con mayor poder ofensivo de las que residen en el Bosque (en caso de empate, se mostrará la de menor ID de entre las que tienen mayor poder ofensivo). La criatura se mostrará con el mismo formato de la instrucción `MostrarCriatura`.
errores	• Si el Bosque está vacío se considerará un error.

salida	En caso de éxito, se genera la expresión:
	
	`CriaturaMasOfensiva: <descripcion_extendida_criatura>`
	
	En caso de fallo, la expresión:
	
	`CriaturaMasOfensiva: FAIL.`

ejemplo	`CriaturaMasOfensiva`
	
	Presentaría el resultado, en caso de éxito:
	
	`CriaturaMasOfensiva: E:{e1,Alarico,I5,A5,C2,25,2,10}`

Instr 15	`CriaturaMasDefensiva`
descripción	Muestra los datos de la criatura con mayor capacidad defensiva de las que residen en el Bosque (en caso de empate, se mostrará a de menor ID de entre las que tienen mayor capacidad defensiva). La criatura se mostrará con el mismo formato de la instrucción `MostrarCriatura`.
errores	• Si el Bosque está vacío se considerará un error.

salida	En caso de éxito, se genera la expresión:
	`CriaturaMasDefensiva: <descripcion_extendida_criatura>`
	En caso de fallo, la expresión:
	`CriaturaMasDefensiva: FAIL.`

ejemplo	`CriaturaMasDefensiva`
	Presentaría el resultado, en caso de éxito:
	`CriaturaMasDefensiva: B:{b0,Bellatrix,S5,M5,B3,V10,6,10,10}`

Instr 16	`Atacar <id_atacante> <id_defensora>`
descripción	Se generará una lucha en la que la criatura con ID `id_atacante` atacará a la criatura con ID `id_defensora`.
errores	• Si no existe alguna de las dos criaturas en la sesión se producirá un error.

salida	En caso de éxito, se genera la expresión:
	`Atacar <id_atacante> <id_defensora>: OK.`
	En caso de fallo, la expresión:
	`Atacar <id_atacante> <id_defensora>: FAIL.`

ejemplo	`Atacar o34 b45`

Instr 17	`VisitarLugarSagrado <id>`
descripción	Hará que la criatura con ID `<id>` se bañe en el LagoSagrado u ore en el TemploMaldito, según corresponda. En caso de que la criatura no fuese usuaria de uno de estos dos lugares, no se produciría ninguna visita.
errores	• Si no existe la criatura con id `<id>` en la sesión se producirá un error.

salida	En caso de éxito, se genera la expresión:
	`VisitarLugarSagrado <id>: OK.`
	En caso de fallo, la expresión:
	`VisitarLugarSagrado <id>: FAIL.`

ejemplo	`VisitarLugarSagrado o34`

Instr 18	`GenerarAsignacionCriaturas <num> <fichero_reparto>`
descripción	Se genera un fichero de asignación de criaturas a jugadores de nombre `<fichero_reparto>`, con un total de `<num>` asignaciones obtenidas aleatoriamente, con los jugadores y criaturas existentes en la sesión.
errores	• Si no existen jugadores en la sesión se produce un error. • Si no hay al menos tantas criaturas como jugadores se produce un error. • Si no se puede escribir el fichero especificado se produce un error.

salida	En caso de éxito se muestra la expresión:
	`GenerarAsignacionCriaturas <num> <fichero_reparto>: OK.`
	En caso de fallo, la expresión:
	`GenerarAsignacionCriaturas <num> <fichero_reparto>: FAIL.`

ejemplo	`GenerarAsignacionCriaturas 20 criaturasXjugador.txt`

Instr 19	`JugarPartida <fichero_reparto> <fichero_partida>`
descripción	Se desencadena una partida entre los jugadores de la sesión con la asignación de criaturas especificado en el fichero <fichero_reparto>. El jug_1 será el que se ha introducido en primer lugar, el jug_2 el segundo, y así sucesivamente. La evolución de la partida se reproducirá en el fichero <fichero_partida>, utilizando el formato descrito en el Apéndice A.5.
errores	• Si no se puede leer el <fichero_reparto> se produce un error. • Si no se puede escribir el <fichero_partida> se produce un error.

salida	En caso de éxito se muestra la expresión:
	`JugarPartida <fichero_reparto> <fichero_partida>: OK.`
	En caso de fallo, la expresión:
	`JugarPartida <fichero_reparto> <fichero_partida>: FAIL.`

ejemplo	`JugarPartida mireparto.txt mipartida.txt`

3.2.3 Ejemplo

Ejemplo de invocación de la aplicación en modo ejecución de instrucciones donde las instrucciones se especifican en el fichero instrucciones.txt y la salida se deposita en el fichero resultado.txt:

```
$> java MundoEncantado -i instrucciones.txt -o resultado.txt
```

Si el fichero “instrucciones.txt” contiene la siguiente información:

```
# Generación de un nuevo fichero de jugadores
CrearJugador 10 Hugh Jackman
CrearJugador 20 Charlize Theron
CrearJugador 1 Uma Thurman
CrearJugador 2 Jean-Claude Van Damme
CrearJugador 2 Joaquin Phoenix
```

```
VolcarJugadores misjugadores.txt
```

Se generará el fichero “misjugadores.txt”, según el formato especificado en el Apéndice A.2.

```
10:{Hugh Jackman}
20:{Charlize Theron}
1:{Uma Thurman}
2:{Jean-Claude Van Damme}
```

y el fichero de salida “resultados.txt” con el siguiente contenido:

```
CrearJugador 10 Hugh Jackman: OK.
CrearJugador 20 Charlize Theron: OK.
CrearJugador 1 Uma Thurman: OK.
CrearJugador 2 Jean-Claude Van Damme: OK.
CrearJugador 2 Joaquin Phoenix: FAIL.
VolcarJugadores misjugadores.txt: OK
```

3.2.4 Gestión de errores

Mismas consideraciones que en el Apartado 3.1.4.

4. DESARROLLO, ENTREGA Y EVALUACIÓN

4.1 Desarrollo del proyecto

El desarrollo del proyecto se llevará a cabo siguiendo el paradigma de Programación Orientada a Objetos. Este desarrollo se articula en dos fases principales:

1. La primera fase consiste en el diseño de un **diagrama UML de clases de alto nivel** adecuado para el proyecto, en el que se identificarán:
    a. Las principales clases e interfaces, así como las relaciones en el dominio del problema (relaciones de agregación, composición, herencia, realización de interfaces, etc.).
    b. Los atributos de cada una de dichas clases/interfaces, multiplicidad, restricciones de cardinalidad, roles, etc.
2. La segunda fase consiste en la implementación de la aplicación. De manera iterativa e incremental, durante esta fase:

a. El diagrama de clases será refinado para la obtención de un diseño detallado mediante un **diagrama UML de clases de bajo nivel** en el que se añadirán todos los métodos necesarios para poder implementar la funcionalidad exigida en el proyecto.

b. Se codificará el modelo orientado a objetos diseñado utilizando el lenguaje de programación multiplataforma Java.

Los alumnos podrán utilizar cualquier herramienta (plugin o standalone) conforme con UML para la realización del diseño de la solución. Se recomienda, no obstante, el uso de Visual Paradigm for UML Community Edition (disponible de manera gratuita para entornos educativos en https://www.visual-paradigm.com/download/community.jsp). Para la codificación se utilizará el Java SE Development Kit 8 de Oracle (disponible en https://www.oracle.com/technetwork/pt/java/javase/downloads/jdk8-downloads-2133151.html). Para la edición del código se podrá utilizar uno de los siguiente entornos o editores de programación[5]:

- El editor de código abierto Visual Code Studio (disponible en https://code.visualstudio.com/)
- El Entorno de Desarrollo Integrado (IDE) de código abierto Eclipse (disponible en https://www.eclipse.org/downloads/packages/).
- Cualquier editor de texto disponible en el entorno del laboratorio.

4.2 Entregas

Los alumnos deberán entregar, dependiendo del modo de evaluación escogido, lo siguiente:

- **Evaluación Continua (EC)**: El desarrollo de la aplicación se realizará en grupos de dos alumnos, constituidos a principios de curso para la realización de las prácticas de iniciación en Java. En esta modalidad de evaluación se realizarán dos entregas:

    1. *Contenido*: Diseño de clases de alto nivel.
    Fecha: fecha oficial aprobada por la CAG y publicada al comienzo del cuatrimestre de impartición de la asignatura.
    Formato: PDF.
    Calificación Máxima Global: 0,5 puntos.

    2. *Contenido*: diagrama de clases de bajo nivel (en el que se hayan incorporado todas las modificaciones que el alumnado considere oportunas desde la primera entrega) junto con el código y la documentación Javadoc del proyecto.
    Fecha: A final de curso (en la fecha oficial aprobada por la CAG).
    Formato: fichero ZIP.

[5] Tenga en cuenta que en el examen práctico deberá realizar una modificación de su aplicación MundoEncantado. Por ello, es conveniente que verifique que su aplicación es editable con las herramientas disponibles en el laboratorio.

Calificación Máxima Global: 3,5 puntos.

La entrega en ambos casos se realizará a través de FaiTIC, mediante un único fichero por grupo.

- **Evaluación Única (EU)**: El desarrollo del proyecto se realizará de forma individual, efectuando una única entrega al final del cuatrimestre (en la fecha oficial aprobada por la CAG), en la que se aportará el diagrama de clases de bajo nivel, el código y su documentación Javadoc. La calificación máxima global será de 5 puntos, 1 punto para el diagrama UML y 4 puntos para el código y documentación.

4.3 Evaluación

Todos los alumnos, con independencia de la modalidad de evaluación elegida (EC o EU) deberán realizar un examen práctico individual (en la fecha oficial aprobada por la CAG) en el que se les pedirá una modificación menor de la implementación del proyecto entregado. La calificación de dicha prueba será APTO o NO APTO. Sólo los alumnos que superen el examen práctico optarán a la evaluación del proyecto; al resto se les asignará directamente una nota de 0 puntos en la implementación del proyecto. La calificación se basará en los criterios especificados en el documento "Criterios evaluación Proyecto", disponible en Faitic.

APÉNDICES. FORMATOS DE FICHEROS

A.1 Fichero de criaturas

El fichero de criaturas contiene información sobre un grupo de criaturas. Se compone de un conjunto de líneas, donde en cada una de ellas se define una única criatura. Cada línea tiene el formato:

<Tipo>:{<ID>,<Nombre>,<AtributoValor>,<AtributoValor>,...}

Siendo:

- <Tipo>: tipo de la criatura: O = Orco, B = Bruja, E = Elfo, N = Ninfa o L = Lamia.
- <ID>: identificador de la criatura.
- <Nombre>: nombre de la criatura.
- <AtributoValor>: asignación del valor para un determinado atributo de la criatura. Las asignaciones se describen con la inicial del atributo y el valor establecido (e.g. R2 establece que la velocidad de una ninfa tiene el valor 2). Cada tipo de criatura soporta un conjunto particular de atributos:
    - Ninfas: D (divinidad), V (varita), R (velocidad/rapidez), E (engaño), A (armadura).
    - Orcos: F (fuerza), G (garrote), E (escudo).
    - Brujas: S (sabiduría), M (poder mágico), B (bastón), V (vestido).
    - Elfos: I (inteligencia), A (arco), C (coraza).
    - Lamia: A (astucia), S (seducción), G (garras), M (mitón).

Ejemplo:

```
O:{o0,Grom,F5,G62,E59}
O:{o1,Thrum,F3,G41,E8}
O:{o2,Drog,F3,G79,E70}
N:{n0,Albania,D0,V674,R2,E0,A422}
N:{n1,Belgica,D0,V874,R0,E0,A532}
N:{n2,Rumania,D2,V360,R0,E1,A144}
E:{e0,Eurico,I3,A3,C1}
E:{e1,Alarico,I2,A3,C5}
E:{e2,Leovigildo,I2,A3,C1}
B:{b0,Bellatrix,S4,M1,B3,V4}
B:{b1,Aughra,S4,M3,B5,V8}
B:{b2,Rhuman,S1,M3,B2,V7}
L:{l0,Lilith,A10,S5,G85,M87}
L:{l1,Xana,A10,S9,G34,M79}
L:{l2,Aelosa,A5,S2,G20,M88}
```

A.2 Fichero de jugadores

El fichero de jugadores contiene información sobre los jugadores que participan en una partida. Se compone de un conjunto de líneas, donde en cada una de ellas se define un único jugador. Cada línea tiene el formato:

<ID>:{<Nombre>}

Siendo:

- <ID>: identificador del jugador.
- <Nombre>: nombre del jugador.

Ejemplo:

```
jA:{Pitufo Goloso}
jB:{Bebe Pitufo}
jC:{Pitufina}
```

En este caso, en la partida participan 3 jugadores. La primera línea describe al jug_1, la segunda la jug_2 y la tercera al jug_3 (así denominados en el Apartado 2.2). En las diversas batallas que conforman una partida en la que participan estos jugadores, siempre será “Pitufo Goloso”, el jugador con el identificador jA, el que empieza como atacante (siempre y cuando hubiese recibido alguna criatura no neutralizada).

A.3. Fichero de reparto

En el fichero de reparto se especifica la asignación de criaturas a cada jugador. Cada línea del fichero contiene

una asignación para una batalla con el siguiente formato:

<ID_jug_1>:{<ID_criatura1_1>, <ID_criatura2_1>, …}, <ID_jug_2>:{<ID_crtatura1_2>, <ID_criatura2_2>, …}, ...

donde:

- <ID_jug_X> es el identificador del jugador jug_X.
- <ID_criaturaY_X> es el identificador de una criatura Y asignada al jugador jug_X.

Ejemplo:

```
jA:{b0,b3,n1},jB:{b1,n2,o3},jC:{e3,n3,o1}
jA:{b0,e2,n0},jB:{e0,e1,o1},jC:{b2,n1,o0}
jA:{n0,n1,o0},jB:{e1,n3,o2},jC:{b1,b3,e0}
jA:{b2,n2,n3},jB:{b1,e1,n1},jC:{b0,e0,o3}
jA:{e1,n3,o3},jB:{b0,b2,b3},jC:{e0,n2,o0}
jA:{b0,e3,n1},jB:{e2,n3,o0},jC:{n0,o1,o2}
jA:{b3,n2,o0},jB:{n0,n1,o1},jC:{b2,e2,o2}
jA:{e0,n2,o0},jB:{b2,e3,o3},jC:{b3,n1,o1}
jA:{b0,o0,o3},jB:{e0,n3,o2},jC:{e2,n1,n2}
jA:{e1,o2,o3},jB:{e2,e3,n3},jC:{b2,e0,n1}
```

A.4. Fichero de instrucciones

Incluye la relación de instrucciones a ejecutar cuando se invoca el programa MundoEncantado en modo "ejecución de instrucciones" (ver Apartado 3.2). Cada instrucción ocupa una única línea del fichero, y sólo puede haber un comando en una línea dada. Las líneas que comienzan por el carácter "#" se consideran comentarios. Las líneas con comentarios y las líneas en blanco se ignoran.

Ejemplo de fichero de comandos:

```
#######################################
# Fichero de instrucciones de ejemplo #
#######################################

# Creamos algunos jugadores
CrearJugador J01 Pepa Blanco
CrearJugador J02 Antonio Cuadrado

# Creamos algunas criaturas
CrearNinfa c1 Etiopia 2 1 1 500 500
CrearOrco c2 Grom 8 90 20
CrearBruja c3 Bellatrix 3 4 7 6
CrearElfo c4 Ataulfo 2 3 4
CrearLamia c5 Xana 2 3 40 50
```

```
# Generamos fichero de asignación de criaturas a jugadores
GenerarAsignacionCriaturas 100 mireparto.txt

# Jugamos partida
JugarPartida mireparto.txt mipartida.txt
```

A.5. Fichero de partida

El fichero de partida describe el desarrollo de una partida. A continuación se muestra un ejemplo:

```
BATALLA_1 jA:{(b1,10),(l2,10),(o2,10)},jB:{(b0,10),(e2,10),(o0,10)},jC:{(b2,10),(e1,10),(o1,10)}
  LUCHA 1: jA-O:{o2,Drog,F3,G79,E70} --> jB-O:{o0,Grom,F5,G62,E59}
  jA:{(b1,10),(l2,10),(o2,9)},jB:{(b0,10),(e2,10),(o0,0)},jC:{(b2,10),(e1,10),(o1,10)}
  LUCHA 2: jB-B:{b0,Bellatrix,S4,M1,B3,V4} --> jC-E:{e1,Alarico,I2,A3,C5}
  jA:{(b1,10),(l2,10),(o2,9)},jB:{(b0,8),(e2,10),(o0,0)},jC:{(b2,10),(e1,7),(o1,10)}
  LUCHA 3: jC-O:{o1,Thrum,F3,G41,E8} --> jA-B:{b1,Aughra,S4,M3,B5,V8}
  jA:{(b1,5),(l2,10),(o2,9)},jB:{(b0,8),(e2,10),(o0,0)},jC:{(b2,10),(e1,7),(o1,9)}
  LUCHA 4: jA-O:{o2,Drog,F3,G76,E70} --> jB-B:{b0,Bellatrix,S4,M1,B2,V4}
  jA:{(b1,5),(l2,10),(o2,8)},jB:{(b0,0),(e2,10),(o0,0)},jC:{(b2,10),(e1,7),(o1,9)}
  LUCHA 5: jB-E:{e2,Leovigildo,I2,A3,C1} --> jC-E:{e1,Alarico,I2,A2,C4}
  jA:{(b1,5),(l2,10),(o2,8)},jB:{(b0,0),(e2,7),(o0,0)},jC:{(b2,10),(e1,4),(o1,9)}
  LUCHA 6: jC-O:{o1,Thrum,F3,G38,E8} --> jA-B:{b1,Aughra,S4,M3,B5,V7}
  jA:{(b1,0),(l2,10),(o2,8)},jB:{(b0,0),(e2,7),(o0,0)},jC:{(b2,10),(e1,4),(o1,8)}
  LUCHA 7: jA-O:{o2,Drog,F3,G73,E70} --> jB-E:{e2,Leovigildo,I2,A2,C1}
  jA:{(b1,0),(l2,10),(o2,7)},jB:{(b0,0),(e2,0),(o0,0)},jC:{(b2,10),(e1,4),(o1,8)}
  LUCHA 8: jC-O:{o1,Thrum,F3,G35,E8} --> jA-L:{l2,Aelosa,A5,S2,G20,M88}
  jA:{(b1,0),(l2,4),(o2,7)},jB:{(b0,0),(e2,0),(o0,0)},jC:{(b2,10),(e1,4),(o1,7)}
  LUCHA 9: jA-O:{o2,Drog,F3,G70,E70} --> jC-B:{b2,Rhuman,S1,M3,B2,V7}
  jA:{(b1,0),(l2,4),(o2,6)},jB:{(b0,0),(e2,0),(o0,0)},jC:{(b2,0),(e1,4),(o1,7)}
  LUCHA 10: jC-O:{o1,Thrum,F3,G32,E8} --> jA-L:{l2,Aelosa,A5,S2,G20,M85}
  jA:{(b1,0),(l2,0),(o2,6)},jB:{(b0,0),(e2,0),(o0,0)},jC:{(b2,0),(e1,4),(o1,6)}

  FIN BATALLA: Ya se han producido 10 luchas.
  PUNTOS CONSEGUIDOS: jA=1,jB=0,jC=2

BATALLA_2 jA:{(e2,0),(n1,10),(n2,10)},jB:{(l0,10),(l2,0),(o0,0)},jC:{(b2,0),(e0,10),(e1,7)}
  LUCHA 1: jA-N:{n1,Belgica,D0,V874,R0,E0,A532} --> jB-L:{l0,Lilith,A10,S5,G85,M87}
  jA:{(e2,0),(n1,8),(n2,10)},jB:{(l0,3),(l2,0),(o0,0)},jC:{(b2,0),(e0,10),(e1,7)}
  LUCHA 2: jB-L:{l0,Lilith,A10,S5,G85,M84} --> jC-E:{e1,Alarico,I2,A2,C3}
  jA:{(e2,0),(n1,8),(n2,10)},jB:{(l0,2),(l2,0),(o0,0)},jC:{(b2,0),(e0,10),(e1,0)}
  LUCHA 3: jC-E:{e0,Eurico,I3,A3,C1} --> jA-N:{n1,Belgica,D0,V869,R0,E0,A532}
  jA:{(e2,0),(n1,3),(n2,10)},jB:{(l0,2),(l2,0),(o0,0)},jC:{(b2,0),(e0,7),(e1,0)}
  LUCHA 4: jA-N:{n1,Belgica,D0,V869,R0,E0,A527} --> jB-L:{l0,Lilith,A10,S5,G80,M84}
  jA:{(e2,0),(n1,1),(n2,10)},jB:{(l0,0),(l2,0),(o0,0)},jC:{(b2,0),(e0,7),(e1,0)}
  LUCHA 5: jC-E:{e0,Eurico,I3,A2,C1} --> jA-N:{n1,Belgica,D0,V864,R0,E0,A527}
  jA:{(e2,0),(n1,0),(n2,10)},jB:{(l0,0),(l2,0),(o0,0)},jC:{(b2,0),(e0,4),(e1,0)}
  LUCHA 6: jA-N:{n2,Rumania,D2,V360,R0,E1,A144} --> jC-E:{e0,Eurico,I3,A2,C1}
  jA:{(e2,0),(n1,0),(n2,8)},jB:{(l0,0),(l2,0),(o0,0)},jC:{(b2,0),(e0,0),(e1,0)}

  FIN BATALLA: Solo hay un jugador activo.
  PUNTOS CONSEGUIDOS: jA=2,jB=0,jC=0

BATALLA_3 jA:{(b2,0),(l2,0),(n1,0)},jB:{(b0,0),(n0,10),(o2,6)},jC:{(e2,0),(l0,0),(o0,0)}

  FIN BATALLA: Solo hay un jugador activo.
  PUNTOS CONSEGUIDOS: jA=0,jB=4,jC=0

BATALLA_4 jA:{(e2,0),(l1,10),(o1,6)},jB:{(b1,0),(e1,0),(n0,10)},jC:{(b2,0),(e0,0),(o2,6)}
  LUCHA 1: jA-L:{l1,Xana,A10,S9,G34,M79} --> jB-N:{n0,Albania,D0,V677,R2,E0,A425}
  jA:{(e2,0),(l1,9),(o1,6)},jB:{(b1,0),(e1,0),(n0,2)},jC:{(b2,0),(e0,0),(o2,6)}
  LUCHA 2: jB-N:{n0,Albania,D0,V677,R2,E0,A420} --> jC-O:{o2,Drog,F3,G73,E73}
  jA:{(e2,0),(l1,9),(o1,6)},jB:{(b1,0),(e1,0),(n0,0)},jC:{(b2,0),(e0,0),(o2,0)}

  FIN BATALLA: Solo hay un jugador activo.
```

```
  PUNTOS CONSEGUIDOS: jA=4,jB=0,jC=0

BATALLA_5 jA:{(n0,0),(n1,0),(o0,0)},jB:{(l2,0),(o1,6),(o2,0)},jC:{(b1,0),(b2,0),(e1,0)}

  FIN BATALLA: Solo hay un jugador activo.
  PUNTOS CONSEGUIDOS: jA=0,jB=2,jC=0

BATALLA_6 jA:{(e0,0),(l0,0),(n0,0)},jB:{(b0,0),(l1,9),(n1,0)},jC:{(b1,0),(l2,0),(o0,0)}

  FIN BATALLA: Solo hay un jugador activo.
  PUNTOS CONSEGUIDOS: jA=0,jB=2,jC=0

BATALLA_7 jA:{(e0,0),(n1,0),(o0,0)},jB:{(b0,0),(l0,0),(o1,6)},jC:{(e1,0),(n2,10),(o2,0)}
  LUCHA 1: jB-O:{o1,Thrum,F3,G41,E14} --> jC-N:{n2,Rumania,D2,V358,R0,E1,A147}
  jA:{(e0,0),(n1,0),(o0,0)},jB:{(b0,0),(l0,0),(o1,5)},jC:{(e1,0),(n2,2),(o2,0)}
  LUCHA 2: jC-N:{n2,Rumania,D2,V358,R0,E1,A142} --> jB-O:{o1,Thrum,F3,G38,E14}
  jA:{(e0,0),(n1,0),(o0,0)},jB:{(b0,0),(l0,0),(o1,0)},jC:{(e1,0),(n2,0),(o2,0)}

  FIN BATALLA: No hay jugadores activos.
  PUNTOS CONSEGUIDOS: jA=0,jB=0,jC=0

BATALLA_8 jA:{(b2,0),(e1,0),(o1,0)},jB:{(e0,0),(l2,0),(n0,0)},jC:{(e2,0),(l1,9),(o2,0)}

  FIN BATALLA: Solo hay un jugador activo.
  PUNTOS CONSEGUIDOS: jA=0,jB=0,jC=2

BATALLA_9 jA:{(b0,0),(o1,0),(o2,0)},jB:{(l1,9),(n0,0),(n2,0)},jC:{(b1,0),(b2,0),(e2,0)}

  FIN BATALLA: Solo hay un jugador activo.
  PUNTOS CONSEGUIDOS: jA=0,jB=2,jC=0

VISITAS A LOS LUGARES SAGRADOS:
  Lago Sagrado: {e1=1, n0=1, n2=1}
  Templo Maldito: {o1=3, o2=2}

PUNTUACIONES:
  Pitufo Goloso (jA) = 7
  Bebe Pitufo (jB) = 10 (VENCEDOR)
  Pitufina (jC) = 4
```

Como se puede observar, el grueso del fichero describe la secuencia de batallas que han ocurrido en la partida. Por cada batalla se incluye la siguiente información:

1. La primera línea especifica el número de batalla y el reparto inicial de criaturas para cada jugador participante. Se utiliza el siguiente formato:

   BATALLA_<N> <id_jug_1>:{(id_c1_1, salud_c1_1), (id_c2_1, salud_c2_1), …}, <id_jug_2>:{(id_c1_2, salud_c1_2), …}, ...

   donde <N> es el número de batalla, <id_jug_X> es el id del jug_X, <id_cY_X> es el ID de la criatura Y asignada al jugador X, y <salud_cY_X> la salud de esa misma criatura.

2. A continuación se describen las luchas que ocurren en la batalla. Cada lucha incluye 2 líneas (**indentadas con 2 espacios**):
   a. La primera especifica el número de lucha y las criaturas que intervienen en la misma. Se utiliza el siguiente formato:

      LUCHA <N>: <id_jug_atacante>-<criatura_atacante> --> <id_jug_defensor>-<criatura_defensora>

      donde <criatura_atacante> y <criatura_defensora> se corresponden a descripciones de las correspondientes criaturas con el mismo formato que aparecen en el fichero de criaturas (ver Apéndice A.1)

b. La segunda línea muestra el estado de salud de las criaturas de cada jugador después de producirse el enfrentamiento. Se utiliza el mismo formato que en la primera línea de la batalla:

<id_jug_1>:{(id_c1_1, salud_c1_1), (id_c2_1, salud_c2_1), …}, <id_jug_2>:{(id_c1_2, salud_c1_2), …}, …

3. Tras la descripción de las luchas y una línea en blanco, se muestra la condición de finalización de la batalla. Puede aparecer uno de los siguientes mensajes (nótese que indentados con 2 espacios en blanco):

   a. " FIN BATALLA: Solo hay un jugador activo."

   b. " FIN BATALLA: No hay jugadores activos."

   c. " FIN BATALLA: Ya se han producido 10 luchas."

   En el caso de que se hubiesen producido 10 luchas y no quedasen jugadores suficientes para establecer una nueva lucha se presentarían el mensaje a o el b (dependiendo de cuantos jugadores activos queden).

   En el ejemplo que se muestra a continuación se observa que en la primera batalla de una partida, tras finalizar la décima lucha solo queda un jugador activo. En ese caso la condición de finalización de la batalla es la insuficiencia de jugadores para realizar una nueva lucha.

```
BATALLA_1
jA:{(b6,10),(b9,10),(o1,10)},jB:{(e4,10),(o2,10),(o4,10)},jC:{(b3,10),(n7,10),(n9,10)}
  LUCHA 1: jA-O:{o1,Thrum,F3,G82,E73} --> jB-O:{o4,Harg,F3,G49,E42}
  jA:{(b6,10),(b9,10),(o1,9)},jB:{(e4,10),(o2,10),(o4,0)},jC:{(b3,10),(n7,10),(n9,10)}
  LUCHA 2: jB-O:{o2,Drog,F10,G56,E0} --> jC-N:{n9,Portugal,D1,V426,R2,E1,A659}
  jA:{(b6,10),(b9,10),(o1,9)},jB:{(e4,10),(o2,9),(o4,0)},jC:{(b3,10),(n7,10),(n9,1)}
  LUCHA 3: jC-B:{b3,Endora,S2,M5,B8,V1} --> jA-B:{b6,Laveau,S3,M4,B9,V8}
  jA:{(b6,2),(b9,10),(o1,9)},jB:{(e4,10),(o2,9),(o4,0)},jC:{(b3,8),(n7,10),(n9,1)}
  LUCHA 4: jA-O:{o1,Thrum,F3,G79,E73} --> jB-E:{e4,Ervigio,I2,A5,C1}
  jA:{(b6,2),(b9,10),(o1,8)},jB:{(e4,0),(o2,9),(o4,0)},jC:{(b3,8),(n7,10),(n9,1)}
  LUCHA 5: jB-O:{o2,Drog,F10,G53,E0} --> jC-N:{n9,Portugal,D1,V426,R2,E1,A654}
  jA:{(b6,2),(b9,10),(o1,8)},jB:{(e4,0),(o2,8),(o4,0)},jC:{(b3,8),(n7,10),(n9,0)}
  LUCHA 6: jC-B:{b3,Endora,S2,M5,B7,V1} --> jA-B:{b6,Laveau,S3,M4,B9,V7}
  jA:{(b6,0),(b9,10),(o1,8)},jB:{(e4,0),(o2,8),(o4,0)},jC:{(b3,6),(n7,10),(n9,0)}
  LUCHA 7: jA-O:{o1,Thrum,F3,G76,E73} --> jB-O:{o2,Drog,F10,G50,E0}
  jA:{(b6,0),(b9,10),(o1,7)},jB:{(e4,0),(o2,0),(o4,0)},jC:{(b3,6),(n7,10),(n9,0)}
  LUCHA 8: jC-B:{b3,Endora,S2,M5,B6,V1} --> jA-O:{o1,Thrum,F3,G73,E73}
  jA:{(b6,0),(b9,10),(o1,0)},jB:{(e4,0),(o2,0),(o4,0)},jC:{(b3,4),(n7,10),(n9,0)}
  LUCHA 9: jA-B:{b9,Myrtle Snow,S1,M1,B10,V1} --> jC-N:{n7,Holanda,D2,V354,R0,E1,A514}
  jA:{(b6,0),(b9,8),(o1,0)},jB:{(e4,0),(o2,0),(o4,0)},jC:{(b3,4),(n7,8),(n9,0)}
  LUCHA 10: jC-B:{b3,Endora,S2,M5,B5,V1} --> jA-B:{b9,Myrtle Snow,S1,M1,B9,V1}
  jA:{(b6,0),(b9,0),(o1,0)},jB:{(e4,0),(o2,0),(o4,0)},jC:{(b3,2),(n7,8),(n9,0)}

  FIN BATALLA: Solo hay un jugador activo.
  PUNTOS CONSEGUIDOS: jA=0,jB=0,jC=4
```

4. Por último, se especifica el número de puntos conseguidos por cada jugador al finalizar la batalla.

Tras la descripción de las batallas, se muestra información sobre el número de visitas que ha realizado cada criatura a los lugares sagrados. Solo aparecen aquellas criaturas que lo han visitado al menos en 1 ocasión. En este ejemplo:

```
VISITAS A LOS LUGARES SAGRADOS:
  Lago Sagrado: {e1=1, n8=1, n9=3}
  Templo Maldito: {o2=3}
```

Se puede observar que la criatura con id n9 ha visitado el LagoSagrado en 3 ocasiones y la criatura con id o2 ha visitado el TemploMaldito también 3 veces. Mientras que las criaturas e1 y n8 se han bañado en el LagoSagrado 1 vez. **La lista de criaturas que han visitado el lugar sagrado aparecen ordenadas por ID (de menor a mayor)**.

Por último se muestran las puntuaciones alcanzadas por cada uno de los jugadores que han participado en la partida. Los jugadores aparecen en orden jug_1, jug_2, etc. En el caso de que alguno de los jugadores tenga más puntos que los demás, ese será el jugador vencedor, y estará identificado con el mensaje “(VENCEDOR)”.

```
PUNTUACIONES:
  Pitufo Goloso (jA) = 4
  Bebe Pitufo (jB) = 6
  Pitufina (jC) = 10 (VENCEDOR)
```