
	<p align="center">IES LUIS VIVES – Grupo: 2º DAM Evaluación Inicial</p>	
<p>Nombre y Apellidos</p>		<p>Nota:</p>

El señor de los anillos: Las minas de Moria



En su viaje hacia Mórdor, Gándalf, Légolas y Frodo tienen que meterse en la ciudad de Moria y atravesar sus entrañas llena de salas con múltiples peligros...

Pero como Gandalf es un viejo precavido te ha contratado porque ha decidido buscar ayuda en el alumnado de 2ºDAM para saber si podrán pasar en base a realizar distintas simulaciones en Kotlin/JAVA que le informe de los resultados y con ellos tomar una decisión final.

El futuro de la comunidad del anillo está en tus manos. Recuerda que el ojo de Sauroon todo lo ve.

Como bien sabemos tenemos **personajes**, que como propiedades tienen el nombre y su estado (vivo o muerto). Además, **Gándalf** (de raza mago) tiene una **vara** con un poder energía total parametrizable entre 1 y un máximo (que puede ser 30). **Légolas** (que es un elfo) por su parte tiene un **carcaj** de flechas con una carga inicial entre 1 y un máximo parametrizable (por ejemplo 20) y **Frodo** (un hobbit) tiene un **anillo** que puede estar puesto o no.

Por otro lado, tenemos las **salas**, que a parte del número de sala, tiene asociado un **peligro**. Los peligros tienen un atributo que identifica el tipo de peligro: “mágico”, “acción” y de “habilidad. Además, la sala que depende del peligro tiene asociado: un **poder maligno** entre 1 y un máximo parametrizable, por ejemplo 10 (si es mágica). Si el peligro de la sala es de acción, habrá una serie de flechas desperdigadas de intentos anteriores de la otros visitantes a Moria que va entre 1 y un máximo parametrizable (por ejemplo 10) y por supuesto una serie de enemigos que va de 1 a un máximo parametrizable (por ejemplo 10).

Por otro lado, cada personaje tendrá una serie de habilidades propias que solo podrá usar él aunque debemos estar preparados por si podrían pasar a usarlas otros personajes. Así tendremos que:

Mágicos: están asociados a Gándalf.

- ✓ `recargarVara(energía)`. Recarga la vara con la energía que le pasemos (esa energía será consumida del total de la vara)
- ✓ `poderVara()`. Devolverá el nivel actual de energía de la vara.

Debemos tener en cuenta que: si el poder de la vara es mayor que la energía maligna de la sala, ganamos siempre; si es igual ganamos el 60% de las veces y si es menor el 30%. Antes de la lucha podemos recargar la vara con una cantidad de energía aleatoria (de 1 a 10) que dependerá de lo inspirado que esté Gándalf en ese momento, esa energía será consumida el total de la vara. Si Gándalf pierde, tratarán de huir a otra sala; lo conseguirán el 80% de las veces.

Acción: están asociados a Légolas.

- ✓ `lanzarFlecha()`. Lanza una flecha y se restarán al total del carcaj.
- ✓ `recargarCarcaj(flechas)`. Aumentará las flechas que tenemos en el carcaj con una cantidad indicada.

Lanzará flechas hasta que no queden enemigos en la sala. Si se queda sin flechas, tratarán de huir; lo conseguirán el 80% de las veces. Tras la batalla, y sólo si hemos ganado, Légolas recargará su carcaj con todas las flechas encontradas en esa sala; justo antes de pasar a la siguiente sala.

Habilidad: Se encarga Frodo.

- ✓ `ponerseAnillo()`
- ✓ `quitarseAnillo()`

Si decide ponerse el anillo (al 50%) se supera el peligro el 90% de las veces si no se lo pone, sólo se supera el 20% de las veces. Si no se supera, igual que antes, pueden huir a la siguiente sala y lo conseguirán el 80% de las veces.

La simulación terminará cuando hayamos recorrido las 36 salas de Moria o cuando no superemos los peligros de alguna sala. Las salas se recorren siguiendo **una cola**, es decir una estructura FIFO (First In, First Out) y se debe implementar el conjunto de sala siguiendo esta directriz.

Cuando acabe la simulación se debe informar qué ha pasado: si nuestros héroes han atravesado Moria o han fallado en su intento y cuantas salas se han superado y cuantas no. Además, deberás guardar en un fichero llamado moria.txt el resultado de la simulación añadiendo la fecha de inicio y fin de esta. El fichero se irá rellenando con las sucesivas simulaciones.

Se debe entregar:

- ✓ Descomposición del problema con todos los elementos relevantes encontrados y los que consideres necesarios para desarrollarlo, razonando la respuesta.
- ✓ Diagrama de clases con los elementos encontrados aplicando técnicas de diseño orientado a objetos y patrones de diseño adecuados.
- ✓ Implementación de la simulación en JAVA o Kotlin usando las técnicas consideres oportunas.
- ✓ Capturas de distintas ejecuciones de este cambiando el valor de los parámetros de configuración iniciales indicados.
- ✓ Enlace al vídeo explicativo en YouTube donde muestres los elementos más importantes de diseño de tu código, su desarrollo y distintas ejecuciones. Se recomienda que el vídeo lo pongas como oculto o visible, pero nunca como privado para que pueda ser visualizado.
- ✓ Gestión del código y repositorio vía Git y GitHub aplicado el modelo GitFlow.
- ✓ Documentación del código usando JDoc.
- ✓ Aplicación de técnicas CleanCode que consideres oportunas.

"Un mago nunca llega tarde Frodo Bolsón. Tampoco llega temprano. Llega justo cuando se lo propone", dijo Gándalf, de la misma manera un buen programador siempre aparece cuando se lo propone y las circunstancias lo requieren.