BLADE OF DARKNESS

El primer ejercicio va a ser un pequeño juego donde tendremos un jugador que se enfrenta a unos monstruos y con unas armas que se equipo debe enfrentarse a unos cuantos monstruos. Tenéis que implementar las siguientes clases:

Clase Arma

Propiedades:

- nombre
- tipo (enum de ESPADA, HACHA, BASTON, ARCO)
- puntosD (puntos de daño)
- dosManos (boolean)

Métodos:

- Constructor parametrizado con todas las propiedades.
- Getters, setters y toString.

Clase Jugador

Propiedades:

- nombre
- clase (enum de MAGO, BRUJO, BARBARO, CABALLERO)
- nivel
- experiencia
- salud (inicialmente a 200)
- Arma armaDerecha
- Arma armaIzquierda

Métodos:

- Debes hacer el constructor parametrizado, menos nivel que será por defecto 1, salud por defecto 200, experiencia 0 por defecto, ni las armas que serán null.
- Getters, setters y toString.
- Un método para subir de nivel, subirNivel(), que incremente el nivel en 1 y suba su salud en 2.5 elevado a nivel. El nivel máximo es 10.
- Un método equipar(Arma arma). Si están libres el arma derecha o izquierda, asignará esa arma a uno de los dos y devolverá true. Si están ocupados los dos devolverá false pues no se puede poner el arma. Si lo que intentas equipar es un arma a dos manos, solo se puede poner si están los dos brazos libres, y se pone la misma arma en los brazos. Se empieza equipando por la derecha.
- Un método tomarPocion(int puntosS): método que sube la salud del jugador tanto como indica puntosS, hasta un máximo de 10000.
- Un método reducirVida(int puntosD): reduce la propia salud del jugador tanto como indica puntosD. Si la salud no es cero tras restar devuelve false, si la salud queda a cero o menos, la salud se pone a cero y se devuelve true (muerto).
- Un método golpear(Monstruo monstruo): reduce la salud del monstruo tanto como sea el valor de la propiedad puntosD de las armas que lleve equipada el jugador, si el arma es doble solo quita el valor de uno

de los brazos. Para reducir la salud debes llamar al método correspondiente reducirVida de la clase Monstruo. Si del golpe matas a un monstruo tu experiencia sube 10 por el nivel del monstruo. Además, cada vez que tu experiencia suba una centena (100, 200, 300, …) subes de nivel. El máximo de experiencia será por tanto 1000.

A modo de ayuda te pongo cómo sería una parte del método *golpear*:

```
public void golpear(Monstruo monstruo) {

    if (this.getArmaDerecha() != null) {
        monstruo.reducirVida(this.getArmaDerecha().getPuntosD());
        if (! this.getArmaDerecha().isDosmanos()) {
            if (this.getArmaIzquierda() != null) {
                monstruo.reducirVida(this.getArmaIzquierda().getPuntosD());
            }
        }
    }

    //Comprobar si has matado al monstruo

    //Subir la experiencia y el nivel si correspondiera

}
```

Clase Monstruo

Propiedades:

- nombre
- clase (enum de GOBLIN, TROLL, SKRALL, DEMONIO, FANTASMA)
- nivel
- salud (inicialmente a 100)
- puntosD (puntos de daño que hace el monstruo al golpear)

Métodos:

- Debes hacer el constructor parametrizado (menos nivel y salud que serán por defecto 1 y 100).
- Getters, setters y toString.
- Un método para subir de nivel, subirNivel(), que incremente el nivel en 1 y suba su salud en 2 elevado a nivel. El nivel máximo es 10.
- Un método reducirVida(int puntosD): reduce la propia salud del monstruo tanto como indica puntosD. Si la salud no es cero tras restar devuelve false, si la salud queda a cero o menos, la salud se pone a cero y se devuelve true (muerto).
- Un método golpear(Jugador jugador): reduce la salud del jugador tanto como sea el valor de la propiedad puntosD del monstruo. Para reducir la salud debes llamar al método correspondiente de la clase Jugador.

Crea una aplicación **TestJuego**, crea un Jugador, equípalo con las armas que desees. Luego crea cuatro monstruos diferentes con niveles y puntos de daño diferentes, y prueba a que combatan contra el jugador. Prueba que algunos monstruos sean fáciles para ver como el jugador sube su experiencia y nivel.

ACADEMIA DE MAGIA

En este juego, vais a crear un sistema donde los magos entrenan y compiten para lanzar hechizos y superar pruebas. Tiene una dificultad añadida que es utilizar las clases List y ArrayList que veremos en el tema siguiente, pero os ayudo a que empecéis a usarlas. El programa consta de tres clases principales:

Clase Mago

Propiedades:

- nombre (String): Nombre del mago.
- energia (int): Energía mágica disponible (de 0 a 100).
- hechizos (**List**<Hechizo>): Lista de hechizos aprendidos.

Métodos:

- Constructor, parametrizado con nombre y energía. La lista se inicializa en el constructor de la siguiente manera:

  this.hechizos = new ArrayList<>();

- Getters, setters y toString
- aprenderHechizo(Hechizo hechizo): Añade un hechizo al repertorio del mago. Se haría así:

  this.hechizos.add(hechizo);

- lanzarHechizo(String nombreHechizo, Prueba prueba): Consume energía y usa el hechizo **si está en la lista**. Si el hechizo supera la prueba, devuelve un éxito. Si no, pierde energía adicional.

  Para ver si el hechizo está en la lista podéis hacer un método nuevo de la siguiente manera:

```
// Método buscar
public Hechizo buscar(String nombreHechizo) {
    for (Hechizo hechizo : hechizos) {
        if (hechizo.getNombre().equalsIgnoreCase(nombreHechizo)) {
            return hechizo; // Hechizo encontrado
        }
    }
    return null; // Si no se encuentra, devuelve null
}
```

- recargarEnergia(int cantidad): Aumenta la energía del mago (sin superar el máximo de 100).

Clase Hechizo

Propiedades:

- nombre (String): Nombre del hechizo (p. ej., "Bola de fuego", "Escudo mágico").
- energiaNecesaria (int): Energía necesaria para lanzarlo.
- potencia (int): Nivel de potencia del hechizo (1-10).

Métodos:

- Constructor, con todos los parámetros
- Getters, setters y toString
- esEfectivo(Prueba prueba): Compara la potencia del hechizo con el nivel de dificultad de la prueba. Si es mayor o igual, el hechizo es efectivo.

Clase Prueba

Propiedades:

- descripcion (String): Descripción de la prueba (p. ej., "Apagar un incendio", "Romper un muro").
- nivelDificultad (int): Nivel de dificultad del reto (1-10).
- recompensa (int): Energía que el mago gana al superar la prueba.

Métodos:

- Constructor, con todos los parámetros
- Getters, setters y toString
- resolver(Hechizo hechizo): Comprueba si el hechizo puede superar la prueba basándose en su potencia.

Reglas del Juego:

1. Un mago puede aprender hechizos y usarlos para resolver pruebas.
2. Si la energía del mago no es suficiente para lanzar un hechizo, debe recargar energía antes de intentarlo.
3. Al resolver una prueba:
    - Si el hechizo es efectivo, el mago gana energía equivalente a la recompensa.
    - Si no, pierde energía equivalente a la dificultad de la prueba.
4. El juego termina cuando el mago se queda sin energía o supera todas las pruebas.

Crea una aplicación TestJuego que se cree un Mago con 100 de energía. Debes añadirle tres Hechizos: bola de fuego (20 energía y 7 potencia), escudo mágico (15 de energía y 5 de potencia), rayo eléctrico (25 de energía y 9 de potencia). Luego crea tres Pruebas: romper muro (5 dificultad y 10 de recompensa), derrotar orco (8 dificultad y 15 recompensa), derrotar elfo oscuro (10 dificultad y 20 de recompensa). Haz que el mago se enfrente a las tres pruebas con un hechizo cada vez a tu elección, comprueba si puede hacerlo y si se queda sin energía.

Se valorará el uso de excepciones, la claridad del código, el uso de comentarios explicativos al principio de los métodos, etc. Si se pega código de IA no se superará el ejercicio.