"JUEGO DEL *AHORCADO*"

Implementa en Java el juego del "***Ahorcado***", cumpliendo las siguientes especificaciones:

- Crea una clase **Ahorcado** que ofrezca la funcionalidad del juego
- Crea una clase **TestAhorcado** donde se probará la clase Ahorcado. Esta clase solo lleva un main.
- Crea una propiedad estática ***palabras*** que sea un array de palabras posibles a adivinar, de forma que el ordenador seleccione de forma aleatoria una de las palabras almacenadas en el array. Ésta será la palabra que el usuario deberá acertar.
- La clase Ahorcado no debe pintar nada con System.out, lo que necesites pintar se devolverá como String y en clase TestAhorcado se pintará el resultado.
- Como propiedades de la clase Ahorcado tendríamos:
    - String **palabraAdivinar**: será una palabra sacada aleatoriamente del array estático de palabras y será lo que el usuario tenga que adivinar.
    - String **palabraIntentada**: será un String que lleve lo que va intentando el jugador, por ejemplo, si palabraAdivinar = "casa", palabraIntentada = "-a-a", si has dicho la letra "a". Inicialmente se rellena con "-".
    - int **numFallos**: el número de intentos que lleva el que esté jugando.
    - String[] **letras**: un array con las letras que va diciendo el jugador, por si queremos comprobar que no repita letras. Este campo no es obligatorio.
- Tendremos un **constructor** sin parámetros que haga lo siguiente:
    - Elija una palabra del array de palabras estático y la grabe en palabraAdivinar.
    - Incialice con "-" palabraIntentada, de la misma longitud que palabraAdivinar.
    - Inicialice numFallos a cero.
    - Cree el array de letras. Tamaño 7, no puede haber más letras que intentos.
- Tendremos los siguientes métodos:
    - **Getters** y **setters** de las propiedades palabraAdivinar, palabraIntentada, numFallos.
    - boolean **probarLetra**(String letra): recorremos la palabraAdivinar carácter a carácter, si aparece la letra parámetro del método en palabraIntentada cambiamos "-" por la letra en cada aparición de la misma y devolvemos true. Si no aparece la letra incrementamos el numFallos y devolvemos false. *Opcional*: se comprueba previamente que la letra no se haya intentado ya. Y luego, tanto en caso de acertar o no, se añade la letra al array *letras*.
    - boolean **probarPalabra**(String palabra): comprueba toda la palabra parámetro del método con palabraAdivinar, si son iguales (equals) devuelve true (has ganado), sino devuelve false (has perdido).
- El funcionamiento del main en TestAhorcado debe ser algo así:
    - Cuando se empiece a jugar, el ordenador informará sobre el número de letras que tiene la palabra.
    - Mostrará un menú con dos opciones: 1. probarLetra, 2. probarPalabra Este menú se repetirá en un bucle hasta acertar, fallar o superar el número de intentos (break).
    - Si usa la opción 1, tanto si ha acertado como si no se mostrará palabraIntentada (lleva guiones en lo que no se ha acertado) y el número de fallos.
    - Si usa la opción 2, si acierta, muestra la palabra, mensaje de enhorabuena y termina el programa. Si falla muestra la palabra, mensaje de lo siento, y termina el programa.
- Prueba el juego

“BLADE OF DARKNESS”

Crea una 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.

Crea una nueva 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:

```java
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

}
```

Crea una nueva 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**, con 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 algunso monstruos sean fáciles para ver como el jugador sube su experiencia y nivel.