Segunda Práctica
Implementación de los métodos simples de las clases

C) Segunda sesión: Implementar métodos básicos de las clases desarrolladas en la sesión anterior en Java y en Ruby.

En la clase Player

- **isDead():boolean**

  Devuelve `true` si el jugador está muerto, `false` en caso contrario.

- **getCombatLevel():int**

  Devuelve el nivel de combate del jugador, que viene dado por su nivel más los bonus que le proporcionan los tesoros que tenga equipados, según las reglas del juego.

- **bringToLife():void**

  Devuelve la vida al jugador, modificando el atributo correspondiente.

- **incrementLevels(i:int):void**

  Incrementa el nivel del jugador en `i` niveles, teniendo en cuenta las reglas del juego.

- **decrementLevels(i:int):void**

  Decrementa el nivel del jugador en `i` niveles, teniendo en cuenta las reglas de juego.

- **setPendingBadConsequence(b:BadConsequence):void**

  Asigna el mal rollo al jugador, dándole valor a su atributo pendingBadConsequence.

- **canIBuyLevels(l:int):boolean**

  Devuelve `true` si con los niveles que compra no gana la partida y `false` en caso contrario.

- **dieIfNoTreasures():void**

  Cambia el estado de jugador a muerto si no tiene ni tesoros visibles ni ocultos, modificando el correspondiente atributo.

- **validState():boolean**

  Devuelve `true` cuando el jugador no tiene ningún mal rollo que cumplir (`pendingBadSConsequece.isEmpty() == true`) y no tiene más de 4 tesoros ocultos y `false` en caso contrario.

- **hasVisibleTreasures():boolean**

  Devuelve `true` cuando el jugador tiene algún tesoro visible y `false` en caso contrario.

- **howManyVisibleTreasures(tKind:TreasureKind):int**

  Devuelve el número de tesoros visibles que tiene del tipo `tKind.`

- **getLevels():int**

  Devuelve los niveles del jugador.

En la clase BadConsequence

- **isEmty():boolean**

  Devuelve `true` cuando el mal rollo que tiene que cumplir el jugador está vacío, eso significa que el conjunto de atributos del mal rollo indican que no hay mal rollo que cumplir, plantéate qué valores deberán tener.

- **myBadConsequenceIsDeath():boolean**

  Devuelve `true` si es un mal rollo es muerte, `false` en caso contrario.

En la clase CardDealer **(puede que algunos ya los tengas implementados):**

- **initTreasureCardDeck()**

  Inicializa en mazo de cartas de Tesoros (`unusedTreasures`) con todas las cartas de tesoros proporcionadas en el pdf de cartas de tesoros.

- **InitMonsterCardDeck()**

  Inicializa el mazo de cartas de monstruos (`unusedMonsters`), con todas las cartas de monstruos proporcionadas en el pdf de cartas de monstruos. Se recomienda reutilizar el código desarrollado en la primera práctica para construir las cartas de monstruos.

- **ShuffleTreasures()**

  Baraja el mazo de cartas de tesoros `unusedTreasures.`

- **ShuffleMonsters()**

  Baraja el mazo de cartas de monstruos `unusedMonsters.`

- **giveTreasureBack(t:Treasure)**

  Introduce en el mazo de descartes de tesoros (`usedTreasures`) el tesoro `t`.

- **giveMonsterBack(m:Monster)**

  Introduce en el mazo de descartes de monstruos (`usedMonsters`) al monstruo `m`.

En la clase Dice

- **nextNumber():int**

  Genera un número aleatorio entre 1 y 6.

En la clase Monster

- **kills():boolean**

  Devuelve `true` cuando el mal rollo del monstruo es muerte y `false` en caso contrario.

- **getLevelsGained():int**

  Devuelve el número de niveles ganados proporcionados por su buen rollo.

- **getTreasuresGained():int**

  Devuelve el número de tesoros ganados proporcionados por su buen rollo.

Complata lo que falta de la clase Treasure. Teniendo en cuenta que todos sus métodos son los consultores básicos.

Define el enumerado CombatResult. De la misma forma que se definió TreasureKind.

Entrega:

- Antes de que comience la sesión en la que se realiza el examen.

Examen de la segunda práctica en la semana del 10 al 14 de noviembre.

- **Lugar:** Aula en la que se desarrolla la correspondiente sesión de prácticas.
- **Día:** El correspondiente a la sesión de cada uno de la semana indicada.
- **Duración**: 20 minutos al comienzo de la sesión.
- **Tipo examen:** Realizar pequeñas modificaciones sobre el propio código.