Apellido y nombre: _____ Legajo: _____

PRIMER EXAMEN PARCIAL 1/07/2024

Resolvé el siguiente ejercicio utilizando el lenguaje **Java** y el paradigma orientado a objetos**.** Lee el enunciado al menos dos veces antes de intentar confeccionar la solución.

ENUNCIADO

Nos han encargado colaborar en la primera versión de un juego de pelea online, llamado UTN Fight. Sus jugadores se almacenan en una única colección y de cada uno de ellos se sabe su ID y su personaje favorito, de los cuales se sabe su nombre elegido y por ahora solo pueden ser de solo alguna de las siguientes variedades:

- **Guerrero**:
  - Puntos de vida: Inicia en 100
  - Guarda una fuerza (valor numérico con decimales)
  - Ataque: Retorna el valor de su fuerza. Si su vida está por debajo del 20% de su valor inicial, su fuerza se incrementa un 10%.
  - Defensa: Recibe el valor de ataque de otro personaje. Debe intentar esquivar el ataque, lo cual se logra obteniendo un valor decmial al azar entre 0.0 y 1.0, el cual se multiplica por el valor de ataque recibido y cuyo valor resultante debe restarse de sus puntos de vida.
- **Arquero**:
  - Puntos de vida: Inicia en 100
  - Guarda una cantidad de flechas y una potencia del arco (valor numérico con decimales)
  - Ataque: Devuelve la potencia del arco. Por cada ataque pierde una flecha, y si no tiene flechas, el valor del ataque será cero.
  - Defensa: Igual que los guerreros

***Ejemplo de defensa para cualquier personaje**: Si recibe un ataque de 40 y el valor aleatorio es 0.3, el daño será 12 (40 * 0.3). Sus puntos de vida se reducirán en 12, y si luego de la resta queda en cero o menos, ha perdido.*

Durante una batalla entre dos personajes, comenzará uno de manera aleatoria a atacar. En el siguiente turno se invertirán los roles y el otro personaje atacará al primero. Este proceso se repite y corta inmediatamente cuando alguno de los personajes queda sin vida, sin importar quien empezó.

Cada jugador debe almacenar el historial de partidas que jugó contra otros. De cada partida se conoce contra quién jugó y el resultado desde su punto de vista (victoria, empate o derrota).

A partir del modelo enunciado, se pide:

1) Desarrollar el diagrama de clases con todas las entidades y relaciones encontradas, en formato **.uxf** (de UMLetino)
2) Desarrollar un método llamado `getResumenDePartidas()` que debe **retornar** (no imprimir por consola) la cantidad de triunfos, derrotas y empates de un jugador.

3) Desarrollar un método llamado `batallar` que reciba el ID de dos jugadores y permita desarrollar una batalla entre ambos hasta obtener un resultado, actualizando el estado de todos las instancias involucradas. Debe retornar si la batalla se pudo desarrollar.

4) Desarrollar un test en el método `main` del proyecto que permita crear un juego y cinco jugadores, cuyos valores quedan a tu gusto. Ejecutar las batallas necesarias para que cada jugador pelee solo una vez contra cada uno de los otros. Finalmente, obtener e imprimir el resumen de partidas de cada jugador (un solo renglón por jugador)

FORMATO DE ENTREGA

Archivo **.zip** exportado desde NetBeans y nombrado de la siguiente manera (reemplazando lo que está en verde):

INSPT-ProgII-p1-2024-UTNFight-TUAPELLIDO-TUNOMBRE

El **.zip** debe contener el proyecto Java, el cual debe tener la misma nomenclatura y el diagrama de clases del modelo en formato **.uxf**.

No cumplir con alguna indicación del formato de entrega disminuye la nota del examen.

CRITERIOS DE EVALUACIÓN

Para considerar aprobado el examen, el mismo debe demostrar la correcta aplicación de los siguientes conceptos de la programación orientada a objetos y el lenguaje Java:

- Definición de clases y asignación adecuada de sus responsabilidades.
- Encapsulamiento, ocultando detalles de implementación y utilizando métodos getters y setters sólo cuando corresponda.
- Modularización reutilizable y mantenible, usando funciones con correcta parametrización, aplicando alta cohesión y bajo acomplamiento.
- Correcta implementación de los constructores.
- Validación de los datos que ingresan al sistema.
- Aplicación de herencia y polimorfismo, incluyendo interfaces.
- Aplicación de los principios S.O.L.I.D
- Algoritmos precisos, finitos y eficientes.
- Buenas prácticas de programación:
  - Uso de constantes en lugar de números mágicos.
  - Nombres de variables, métodos y clases descriptivos, que sigan la convención del lenguaje.
  - Uso adecuado de los ciclos. No alterar la naturaleza de los mismos utilizando, por ejemplo, instrucciones de control como `break` y `continue`.
  - Código ordenado e indentado correctamente.