Tarea 1 - Batallas Pokémon(c)

Profesor: Alexandre Bergel
Auxiliar: Sebastián Ferrada

En esta tarea se le pedirá que implemente un modelo de batallas Pokémon. Su aplicación debe estar debidamente testeada y documentada y debe utilizar buenas prácticas de diseño. Especificaciones serán dadas más adelante.

1. Requisitos

Un pokémon tiene un nombre, un número que indica sus puntos de vida totales, un número a modo de contador de daño recibido, un tipo y un número que indica cuánto daño hace al atacar (considere que solo tiene **un** ataque que es del mismo tipo del pokémon). Considere que existen solo 8 tipos de pokémon: planta, fuego, agua, eléctrico, tierra, psíquico, lucha y normal.

Un pokémon debe ser capaz de atacar a otro. Para esto se debe descontar el valor del ataque de los puntos de vida del pokémon atacado. Sin embargo, antes de realizar la resta, se deben considerar efectos de debilidad y resistencia. La debilidad y resistencia depende de los tipos del pokémon atacante y el pokémon atacado. Si el pokémon atacado es débil al tipo del pokémon atacante, el valor de ataque se duplica. Si el pokémon atacado es resistente, se le restan 20 puntos al valor de ataque. En el Cuadro 1 encontrará las relaciones de debilidad/resistencia que se definirán para este juego. Tenga presente que si el contador de daño de un pokémon iguala o supera sus puntos de vida, el pokémon queda fuera de competencia (y no puede seguir atacando).

Tipo Pokémon	Debilidad	Resistencia
Planta	Fuego	Agua, Tierra
Fuego	Agua, Tierra	Planta
Agua	Planta, Eléctrico	Fuego
Eléctrico	Tierra	Eléctrico
Tierra	Agua, Planta	Normal, Eléctrico
Psíquico	Psíquico	-
Lucha	Psíquico	Tierra
Normal	Lucha	Psíquico

Cuadro 1: Resumen de debilidad y resistencia

2. Requerimientos Adicionales

Además de una implementación basada en las buenas prácticas y patrones de diseño visto en clases, usted además debe considerar:

- **Cobertura.** Cree los tests unitarios, en JUnit, que sean necesarios para alcanzar un 90 % de coverage por paquete. Estos tests deben estar en la carpeta `test` de su proyecto.
- **Javadoc.** Cada clase pública y cada método público deben estar debidamente documentados siguiendo las convenciones de Javadoc[1].
- **Resumen.** Debe entregar un archivo `pdf` que contenga su nombre y un diagrama UML que muestre las clases y métodos que usted definió en su proyecto.

Todos estos ítemes serán considerados para la nota final.

3. Evaluación

- **Código fuente (3.5 puntos)** Se analizará que su código provea la funcionalidad descrita y esté implementada utilizando un buen diseño.
- **Tests (1 punto)** Sus casos de prueba deben crear diversos escenarios y contar con un coverage de 90 % por paquete
- **Javadoc (1 punto)** Cada clase y método público debe ser documentado. Se descontará por cada falta.
- **Resumen (0.5 puntos)** El resumen mencionado en la sección anterior. De no enviarlo, se considerará que no utilizó un buen diseño y será penalizado.

4. Entrega

Usted debe subir a U-Cursos o bien el proyecto de eclipse realizado o bien las carpetas `src` y `test` de su proyecto. El plazo de entrega es hasta el 9 de abril a las 23:59 hrs. No se aceptarán peticiones de extensión del plazo.

[1] `http://www.oracle.com/technetwork/articles/java/index-137868.html`