Tarea 2 - Pokémon TCG: Electric Boogaloo

Profesor: Alexandre Bergel
Auxiliar: Juan-Pablo Silva, Ignacio Slater

En el capítulo anterior de *metodologías de diseño y programación* se implementaron las estructuras básicas para el juego de cartas. En esta iteración, se espera que extiendan esa implementación agregando nuevas funcionalidades a su programa.

1. Requisitos

Teniendo las entidades básicas de juego, ya se puede comenzar a implementar una lógica que relacione las entidades con un jugador real, para esto, se requiere que defina un controlador que se encargará de mantener el estado del juego en todo momento y de comunicar las acciones que debe hacer cada una de las entidades que componen la aplicación. Además, en esta tarea comenzaremos a hablar del concepto de turno. Los requisitos del controlador se explicarán en detalle más adelante.

Primero, se deberán definir algunas entidades y funcionalidades adicionales.

1.1. Entidades

1.1.1. Entrenador (continuado)

Ya se definió el entrenador, pero no está completo, es necesario que además de los componentes que ya tenía, ahora tenga un *mazo* de cartas, una *pila de descarte* a la que irán los Pokémon derrotados y las cartas que deban ser descartadas por algún motivo, y hasta 6 *cartas de premio*. El mazo del jugador comienza con 60 cartas. Ningún jugador debe ser capaz de ver las cartas que están en su mazo o en sus premios a excepción de que alguna condición especial se cumpla.

1.1.2. Cartas de energía (continuado)

Considere ahora que las cartas de energía pueden agregarse también a Pokémon de la banca y que sólo se puede jugar 1 por turno.

1.1.3. Evoluciones

Los Pokémon no sólo tienen un tipo que define sus debilidades y resistencias, además tienen una fase, éstas pueden ser: *básico*, *fase 1* y *fase 2*. Los Pokémon básicos pueden ser jugados directamente a la banca, pero los otros necesitan que su preevolución esté presente en el campo, para identificar a la preevolución se puede utilizar el id que se definió en la entrega anterior. Un Pokémon fase 1 necesita que un Pokémon básico esté en juego, mientras que un fase 2 requiere uno de fase 1. Al jugar un Pokémon evolucionado, esta carta pasa a reemplazar a su preevolución en el juego y la preevolución pasa a la pila de descarte, todas las energías asociadas al Pokémon original pasan a su evolución.

1.1.4. Habilidades

Hasta el momento sólo se han definido ataques, los ataques son un tipo particular de habilidad. Una habilidad tiene los mismos atributos que un ataque, pero no tiene un daño base. Una habilidad puede usarse por el jugador o activarse por algún otro motivo. Utilizar una habilidad cambia de alguna manera el estado del juego, y el efecto que tienen, así como las condiciones necesarias para usarla estarán definidas dentro de la habilidad misma.

Como se mencionó, un ataque es un tipo particular de habilidad, así que también pueden tener un efecto sobre el estado del juego. Adicionalmente, un ataque es lo último que puede hacer un jugador en su turno (i.e. si un entrenador ataca, su turno se termina).

1.1.5. Cartas de entrenador

Se agregará al juego un nuevo tipo de carta, las cartas de entrenador. Estas cartas alteran el estado del juego al momento de ser jugadas, deben tener un nombre y un texto descriptivo. Adicionalmente, existen 3 tipos de cartas de entrenador.

- **Objetos:** Estas cartas tienen un efecto sobre alguno de los Pokémon en juego, estas pueden tener un efecto instantáneo (en cuyo caso se descartan luego de usarse) o pueden quedar asociadas al Pokémon, con la condición de que no se puede equipar un objeto a un Pokémon que ya tenga otro equipado.
- **Estadio:** Este tipo de cartas tiene un efecto sobre el campo de juego, y por lo tanto, sobre todas las cartas en juego. Sólo puede haber una de estas cartas en juego a la vez, y si se juega otra, ésta reemplaza a la anterior.
- **Soporte:** Todas estas cartas son de efecto instantáneo, al momento de jugarlas se activa su efecto y luego se descartan. Sólo puede jugarse una carta de soporte por turno.

1.2. Controlador

El controlador manejará la lógica de una partida de cartas, definirá la estructura de los turnos y servirá de interfaz para manejar las entidades del juego.

Para esta entrega, se requiere que el controlador permita simular un turno del juego[1]. Para esto, el controlador debe tener conocimiento de los entrenadores que participan de la partida y de qué jugador es el turno actual. Un turno se estructura como sigue:

1. El jugador roba una carta del mazo y la agrega a su mano
2. Luego de robar, el entrenador puede:
   - Ver las cartas de su mano y jugarlas
   - Ver a los Pokémon en juego (tanto los suyos como los de su oponente)
   - Utilizar alguna habilidad de su Pokémon activo

3. Terminar su turno o atacar

De momento no es necesario que las acciones de un turno se ejecuten en secuencia, basta con que el controlador pueda realizar cada acción de manera independiente.

Tenga en cuenta que el controlador es el que mantiene el juego, así que los cambios que se hagan al estado del juego por parte de las entidades deben ser notificadas a este.

1.3. Requisitos específicos

Además de lo pedido en las secciones anteriores, se le solicitará que implemente algunas de las funcionalidades siguientes. Queda a su criterio cuales implementar, pero debe tener al menos 1 por cada tipo de habilidad (5).

Si implementa más de las funcionalidades mínimas solicitadas (hasta 3), puede obtener una bonificación de hasta 0.5 puntos en su nota final. Los nombres de las habilidades y cartas son referencias a su contraparte en el juego de cartas realen en las que están basadas, pero no son exáctamente iguales a ellas.

1.3.1. Habilidades pokémon

- **Wing Buzz:** Una vez por turno, si éste Pokémon es el activo, puedes descartar una carta de tu mano. Si lo haces, descarta la carta superior del mazo de tu oponente
- **Damage Swap:** Tantas veces como quieras, puedes mover un contador de daño[2]de uno de tus Pokémon a otro, siempre y cuando el otro Pokémon no quede sin hp.
- **Rain Dance:** Tantas veces como quieras, puedes utilizar una carta de energía del tipo del Pokémon sobre uno de tus Pokémon del mismo tipo (ignorando el límite de 1 energía por turno).
- **Energy Burn:** Todas las energías asociadas al Pokémon se vuelven del tipo del Pokémon.
- **Strikes Back:** Cuando el ataque de un oponente dañe a este Pokémon, el Pokémon atacante recibe x de daño.
- **Energy Trans:** Tantas veces como quieras, puedes transferir una energía del tipo de este Pokémon a otro en juego.
- **Buzzap:** Si este es tu Pokémon activo, en cualquier momento de tu turno, puedes descartar a éste Pokémon y agregarle 2 energías de cualquier tipo a otro Pokémon.
- **Invisible Wall:** Cuando un ataque hace x o más daño al Pokémon previene ese daño.

[1]No se requiere que se pueda jugar aún, sólo que se implementen los métodos necesarios para ejecutar todas las acciones de un turno.

- **Shift:** Una vez por turno, puedes cambiar el tipo de este Pokémon por el tipo de cualquier otro Pokémon en juego.
- **Heal:** Una vez por turno, puedes lanzar una moneda. Siresulta cara, puedes remover 1 contador de daño de uno de tus Pokémon.

1.3.2. Ataques

- **Afterimage Assault:** Busca en tu mazo hasta 2 Pokémon de la misma especie que el que usó esta habilidad y colocalos en la banca, luego revuelve el mazo.
- **Hydro Pump** Hace x de daño más 10 por cada energía del tipo del Pokémon asociada al Pokémon que no haya sido usada para pagar por este ataque. No se puede agregar más de y de daño de esta forma.
- **Fire Spin:** Descarta x energías asociadas al Pokémon o no puede atacar.
- **Electric Shock:** Lanza una moneda, si sale sello, este Pokémon recibe x de daño.
- **Meditate:** Hace x de daño más 10 por cada contador de daño en el Pokémon defensor.

1.3.3. Objetos

- **Exp. Share:** Cuando tu Pokémon activo es derrotado por el ataque de un enemigo, puedes mover una energía de ese Pokémon al que tiene equipada esta carta.
- **Super Rod:** Lanza una moneda. Si sale cara, coloca un Pokémon evolucionado de tu pila de descartes en tu mano. Si sale sello, coloca un Pokémon básico de tu pila de descartes en tu mano.
- **Rare Candy:** Selecciona un Pokémon básico en juego. Si tienes en tu mano un Pokémon fase 1 o fase 2 que evolucione de éste, juega la carta[3].
- **Super Scoop Up:** Lanza una moneda. Si sale cara pon uno de tus Pokémon en juego y todas las cartas asociadas a él en tu mano.
- **Potion:** Remueve hasta x contadores de daño de uno de tus Pokémon.
- **Great Ball:** Busca un Pokémon básico en tu mazo y juegalo a tu banca. Luego revuelve el mazo.
- **Trainers' Mail:** Mira las x primeras cartas de tu mazo. Puedes elegir entre ellas una carta de entrenador, mostrarsela a tu oponente y ponerla en tu mano. Luego revuelve el mazo.

[2]En este contexto, se considerará un contador de daño a cada 10 hp que haya perdido un Pokémon.
[3]Note que esto significa que un Pokémon básico podría evolucionar directamente a uno de fase 2 en este caso

1.3.4. Estadios

- **Pokémon Park:** Durante el turno de cada jugador, cuando se juega una carta de energía sobre un Pokémon, se remueve hasta un contador de daño de dicho Pokémon.
- **Lucky Stadium:** Una vez durante su turno, el jugador puede lanzar una moneda. Si sale cara, el jugador roba una carta.
- **Frozen City:** Cada vez que un jugador agrega una energía de su mano a alguno de sus Pokémon, pon x contadores de daño en ese Pokémon.
- **Training Center:** Cada Pokémon fase 1 o fase 2 en juego tiene $+x$ de hp.
- **Chaos Gym:** Cada vez que un jugador juega una carta de entrenador que no sea un estadio, lanza una moneda. Si sale cara, el jugador juega esa carta. Si sale sello, el jugador no puede jugar esa carta. Si la carta no se juega, el oponente puede usarla, si cumple con los requisitos para jugarla. En ambos casos, la carta se va a la pila de descartes del dueño original de la carta.

1.3.5. Soporte

- **Professor Cozmo's Discovery:** Lanza una moneda. Si sale cara roba las 3 últimas cartas de tu mazo. Si sale sello roba las 2 primeras.
- **Professor Elm's Training Method:** Busca en tu mazo un Pokémon evolucionado, muestraselo a tu oponente y colocalo en tu mano. Luego, revuelve el mazo.
- **N:** Cada jugador de vuelve su mano al mazo y lo revuelve. Luego saca tantas cartas como cartas de premio le queden.
- **Iris:** Durante este turno, tu Pokémon activo hace 10 más de daño por cada carta de premio que tu oponente haya tomado.
- **Professor Juniper:** Descarta tu mano y roba 7 cartas.
- **Ghetsis:** Tu oponente muestra su mano y devuelve todas las cartas de objeto que tenga a su mazo y luego lo revuelve. Luego, roba tantas cartas como tu oponente haya devuelto a su mazo.
- **Ninja Boy:** Selecciona uno de tus Pokémon básicos en juego. Busca en tu mazo algún Pokémon básico y reemplázalo con el otro (todas las cartas, energías y contadores de daño pasan al Pokémon nuevo). Retorna la primera carta al mazo y luego revuelvelo.

2. Requerimientos adicionales

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

- **Cobertura:** Cree los tests unitarios, usando JUnit 4, que sean necesarios para tener al menos un coverage del 90 % de las líneas por paquete. Todos los tests de su proyecto deben estar en el paquete `test`.
- **Javadoc:** Cada interfaz, clase pública y método público deben estar debidamente documentados siguiendo las convenciones de Javadoc[4]. En particular necesita `@author` y una pequeña descripción para su clase e interfaz, y `@param`, `@return` (si aplica) y una descripción para los métodos.
- **Resumen:** Debe entregar un archivo **pdf** que contenga su nombre, rut, usuario de Github, un link al repositorio de su tarea y un diagrama UML que muestre las clases, interfaces y métodos que usted definió en su proyecto. **No debe incluir los tests** en su diagrama.
- **Git:** Debe hacer uso de git para el versionamiento de su proyecto. Luego, esta historia de versionamiento debe ser subida a Github.
- **Readme:** Modifique el readme de la tarea anterior para incluir los detalles de la nueva implementación. Es sumamente importante que especifique cuales de las funcionalidades opcionales implementó y señalar qué test prueba que funcionen.

3. Evaluación

- **Código fuente (4.0 puntos):** Este ítem se divide en 2:
  - **Funcionalidad:** Se analizará que su código provea la funcionalidad pedida. Para esto, se exigirá que testee las funcionalidades que implementó. **Si una funcionalidad no se testea, no se podrá comprobar que funciona y, por lo tanto, NO SE ASIGNARÁ PUNTAJE por ella.**
  - **Diseño:** Se analizará que su código provea la funcionalidad implementada utilizando un buen diseño.
- **Coverage (1.0 puntos):** Sus casos de prueba deben crear diversos escenarios y contar con un coverage de las líneas de al menos 90 % por paquete. No está de más decir que sus tests deben testear algo (es decir, no ser tests vacios o sin asserts).
- **Javadoc (0.5 puntos):** Cada clase, interfaz y método público debe ser debidamente documentado. Se descontará por cada falta.

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

- **Resumen (0.5 puntos):** El resumen mencionado en la sección anterior. Se evaluará que haya incluido el diagrama UML de su proyecto. **En caso de no enviarse el resumen con el link a su repositorio su tarea no será corregida**[5].

Entrega

Suba el código de su tarea al mismo repositorio en el que trabajó la tarea 1. Si quieren pueden crear un nuevo repositorio, pero deben seguir las mismas reglas que con la tarea 1.

Usted debe subir a U-Cursos **solamente el resumen**, con su nombre, rut, usuario de Github, link a su repositorio y diagrama UML. Además, debe marcar con el tag *#tarea2* el commit que quiere que sea revisado para facilitar su corrección. El código de su tarea será bajado de Github directamente. El plazo de entrega es hasta el 29 de mayo a las 23:59 hrs. Se verificará que efectivamente el último commit se haya hecho antes de la fecha límite. **No se aceptarán peticiones de extensión de plazo**, pero se mantiene la regla de las 72 horas de la tarea anterior.

Recomendaciones

No estamos interesados en un programa que solamente funcione. Este curso contempla el diseño de su solución y la aplicación de buenas prácticas de programación que ha aprendido en el curso. No se conforme con el primer diseño que se le venga a la mente, intente ver si puede mejorarlo y hacerlo más extensible.

No comience su tarea a último momento. Esto es lo que se dice en todos los cursos, pero es particularmente importante/cierto en este. Si usted hace la tarea a último minuto lo más seguro es que no tenga tiempo para reflexionar sobre su diseño, y termine entregando un diseño deficiente o sin usar lo enseñado en el curso.

Haga la documentación de su programa en inglés (no es necesario). La documentación de casi cualquier programa open-source se encuentra en este idioma. Considere esta oportunidad para practicar su inglés.

Les pedimos encarecidamente que las consultas referentes a la tarea las hagan por el **foro de U-Cursos**, en la categoría "Consultas Tarea: Tarea 2". En caso de no obtener respuesta en un tiempo razonable, pueden hacernos llegar un correo al auxiliar o ayudantes.

¡Éxito!

[5] porque no tenemos su código.