PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC2233 Programación Avanzada (II/2016)

Tarea 1

1. Objetivos

- Aplicar conceptos de programación orientada a objetos (OOP) para modelar un problema.
- Trabajar con archivos de texto para leer y procesar datos.
- Desarrollar algoritmos para la resolución de problemas complejos.

2. Introducción

Emocionado con la salida de Pokémon Go, vas rápidamente a la Play Store y al clickear descargar, te encuentras con el siguiente mensaje:

"Tu dispositivo no es compatible con esta versión"

Estás enojado. Sueñas con esos días de tu niñez en que podías jugar Pokémon Red sin preocuparte de que el servidor no estuviese funcionando. Justo en ese momento de ira y desesperación, te llega el siguiente correo:

¡Hola! Bienvenidos al mundo de los programones. Mi nombre es Ivania pero las personas me llaman Profesora Programón. Este mundo está habitado por criaturas llamadas programones. Para algunas personas, éstos son mascotas. Otros los utilizan para batallas. Yo ... yo estudio programones como profesión. Este mundo como lo conocemos está a punto de extinguirse. Una nueva ola de programones está invadiendo nuestro mundo y es mi deber detenerlo y revivir uno de los juegos clásicos de este mundo: Programón Rojo, un sistema de batallas entre programones. Para esto, los he seleccionado a ustedes, brillantes alumnos de Programación Avanzada para que me ayuden a renacer este juego del pasado. Prepárense, su propia leyenda Programón está a punto de desplegarse. ¡Un mundo de sueños y aventuras con Programón espera! ¡Vamos!

No lo piensas dos veces, y corres a encerrarte a programar día y noche, hasta que Programón Rojo sea un éxito mundial y la profesora Ivania muestre una profunda admiración por tu gran trabajo.

3. Problema

En esta tarea deberás modelar un sistema de batallas entre programones mediante programación orientada a objetos. Esto implica el uso de clases y la relación correcta entre ellas. Además, deberás crear un programa en consola que permita a los jugadores formar parte del mundo y hacer uso de sus funcionalidades.

4. Programón Rojo

4.1. Estructura

Programón Rojo funciona mediante la interacción entre las siguientes entidades:

- Programón: Todo programón tiene un identificador, un nombre, una lista de movimientos, un tipo, y una serie de estadísticas como vida, velocidad, ataque, ataque especial, defensa y defensa especial. También contiene el nivel actual en el que se encuentra el programón, en cuál evoluciona y a qué programón evoluciona[1].
- Prograbola: aparato con el cual se captura a un **programón salvaje** (que se encuentre en la hierba). Cada prograbola puede capturar a lo más un programón (puede pasar que no sea capturado) y luego ésta queda inutilizable. Se compran en la tienda de prograbolas.
- Progradex: Contiene información acerca de los programones organizada en dos tipos: Los que han sido vistos (en batallas), donde sólo se tiene el nombre e identificador de éste y aquellos que han sido capturados (a través de una prograbola en una batalla) y se conoce toda la información detallada en el ítem anterior.
- PC de Bastián: Contiene todos los programones capturados a través de una prograbola y contiene toda la información de estos por cada jugador. Cuando uno de estos evoluciona, se debe actualizar su información por la del programón evolucionado. Cada jugador ingresa al PC de Bastián con su nombre y contraseña.
- Jugador: Tiene un identificador único de jugador, un nombre (también único), contraseña, dinero disponible (todos los jugadores nuevos comienzan con 1.000 yenes), una progradex, medallas que se obtienen al ganarle una batalla al líder de un gimnasio y prograbolas (todos los jugadores nuevos comienzan con 10 prograbolas).
- Entrenador: Tiene un nombre y un equipo de programones.
- Ataques: Contienen puntos de poder, tipo, nombre y precisión.
- Ciudad: contiene un nombre, un id, una tienda de prograbolas, un centro programón y un Gimnasio. Se especifica a continuación qué contiene cada uno de éstos:
  - Tienda de Prograbolas: es donde se compran las prograbolas.
  - Centro programón: es donde se puede acceder al PC de Bastián para cambiar del equipo a algún programón capturado que se encuentre en el PC (equipo tiene un máximo de 6 programones).
  - Gimnasio: Es donde deberás pelear contra los entrenadores del Gimnasio antes de poder batallar contra el líder. En caso de derrotar el gimnasio deberás recibir una medalla que demuestre esto.

4.2. Funcionalidades

Programón Rojo debe tener las siguientes funcionalidades:

- Antes de ingresar al sistema:
  - Sistema de ingreso y registro: Cada jugador será capaz de hacer uso del sistema luego de autentificarse (con nombre y contraseña). Además, se debe dar la opción de registrar un nuevo jugador llenando todos los datos pertinentes para que sean agregados al sistema (notar que su sistema debe aceptar nuevos jugadores que contengan un nombre de usuario que no haya sido utilizado previamente, en caso contrario, no se permitirá crearlos). Todos los jugadores nuevos comienzan

[1] Totalmente original

con 1.000 yenes, sin medallas y el jugador debe elegir uno de los siguientes programones: Charmander, Squirtle y Bulbasaur, que han de comenzar en nivel 5. Además, todos comienzan en Pallet Town (notar que esta ciudad no contiene gimnasio ni tienda de prograbolas).

- Luego de ingresar al sistema, el jugador debe situarse en el lugar donde quedó por última vez (o Pallet Town si es un jugador nuevo) y las siguientes funcionalidades deben ser implementadas (se debe poder acceder a ellas a través de un menú inicial y siempre se debe dar la opción de cancelar y volver un estado anterior en el menú de navegación):

  - Programones en la progradex: se deben mostrar tanto lo programones vistos como los capturados, con todos los datos disponibles dependiendo de si fue visto o capturado. Se debe mostrar en orden, primero todos los capturados y luego los vistos. Un programón se declara visto cuando el jugador se enfrenta a él en una batalla (aún cuando decida abandonarla o es derrotado).
  - Caminar: al decidir caminar, se debe dar la opción de caminar hacia adelante o hacia atrás. Una vez elegida una opción, se avanza en una unidad y pueden ocurrir 3 eventos. Entrar a una ciudad, encontrar un programón en la hierba y comenzar una batalla o seguir en la hierba. Se especifica que ocurre en cada caso:
    - Entra a una ciudad: Se debe dar la opción de ir al centro programón, al gimnasio, la tienda de programones o salir de la ciudad en dirección a la siguiente si se cumplen con los requisitos. Ustedes deciden como manejar este último punto (si se da la opción sólo cuando es posible salir de la ciudad o una vez elegida esta opción se alerta en caso de no cumplir con los requisitos).
      - Centro Programón: se debe dar la opción de ir al PC de Bastián a intercambiar a los programones del equipo.
      - Gimnasio: se deben mostrar todos los entrenadores de dicho gimnasio y elegir a uno para comenzar una batalla, luego de derrotarlos a todos se podra batallar con el lider del Gimnasio.
      - Tienda de Prograbolas: se da la opción de comprar prograbolas. Cada una cuesta 500 yenes. Puede asumir que el stock de la tienda es infinito.
    - Encontrar un programón en la hierba: al encontrar un programón, se inicia automáticamente una batalla y en cada turno se debe dar la opción de pelear y elegir un movimiento (siempre y cuando los puntos de poder del movimiento sea mayor a 0), cambiar al programón por otro del equipo y finalmente, ~~acobardarse~~ salir de la batalla y volver a la hierba. Si se cambia de programón se pierde automáticamente el turno y se deberá esperar hasta el próximo para utilizar a este nuevo programón.
  - Datos del jugador: desplegar todos los datos del jugador.
  - Salir del sistema: Cuando un jugador desee dejar de utilizar el sistema, el programa debe cerrarse y todo lo realizado (programones capturados (con nivel, stats actuales, etc), batallas realizadas, lugar del mapa en donde quedó, medallas, etc.) deben guardarse (se recomienda crear archivos nuevos ya que todos los entregados corresponden a una base para cualquier jugador nuevo).
  - Consultas: El entrenador es capaz de realizar consultas posteriormente señaladas.

4.3. Consultas

- Batalla por programón involucrado: Ingresando el nombre del programón, se deberá mostrar un resumen de todas las batallas en la que este se haya visto involucrado y el resultado de estas (ganador o perdedor). Tomar en cuenta si este fue un programón salvaje con el que se peleó, un programón de algún entrenador (en caso de esto, indicar el entrenador), o propio del entrenador. Indicar su contrincante y resultado.
- Ranking de programones: Se deberá mostrar los 10 programones[2] con mayor porcentaje de batallas ganadas, pero solo califican donde el programón ha participado en más de 10 batallas.

[2]Entiéndase programón como Çlasez no como la instancia de este

- Jugador: Para un jugador mostrar sus medallas ganadas, dinero, programones con sus niveles con sus stats respectivos, y todas las peleas que este ha luchado con el resultado correspondiente.

4.4. Cálculo de Stats

Cada vez que el jugador captura un programón o uno de los que ya tiene aumenta de nivel, se deben calcular las nuevos stats de este a partir de los stats base (que se encuentran en programones.json) y el nivel actual del programón, Para el cálculo de estadísticas de los programones usted deberá hacer el calculo mediante de las siguientes formulas:

$$HP = \left\lfloor \frac{\left((BASE + IV) \times 2 + \left\lfloor \frac{\lceil \sqrt{EV} \rceil}{4} \right\rfloor\right) \times Nivel}{100} \right\rfloor + Nivel + 10$$

$$OtroStat^{3} = \left\lfloor \frac{\left((BASE + IV) \times 2 + \left\lfloor \frac{\lceil \sqrt{EV} \rceil}{4} \right\rfloor\right) \times Nivel}{100} \right\rfloor + 5$$

- BASE: es el valor base del stat a calcular que tiene el programón, este se encuentra dado por los archivos entregados.
- IV: este valor deberá ser generado al momento de encontrarse con algún programón o entrar en batalla y deberá ser guardado guardado, este deberá variar entre $(0 - 15)$
- EV: al igual que los IV's , este modificador debe variar entre $(0 - 65535)$

5. Sistema de Batallas

- La batalla consistirá en turnos donde, el programón más rápido será el que obtenga el primer ataque en cada turno, luego el jugador podrá elegir el movimiento que usara su programón para atacar, y la computadora lo hará de forma random, escogiendo ataques que todavía tengan pp.
- La probabilidad de acertar un ataque esta dada por el accuracy del ataque a utilizar.
- Ataques: Después de utilizar un ataque en cada turno, se deberá disminuir los puntos de poder del ataque utilizado en una unidad.
- Daño: Determina cuanto daño le hace un programón a otro (disminuyendo el HP del programón atacado). Este depende del valor de ataque o ataque especial, del valor de defensa o defensa especial del oponente y el poder base del movimiento elegido. Debe ser calculado de la siguiente manera:

$$\text{Daño } = \left(\frac{2 \times Nivel + 10}{250} \times \frac{Ataque}{Defensa} \times Base + 2\right) \times Modificador$$

Donde:

- Nivel: nivel del programón atacante.
- Ataque y Defensa: estadísticas ataque y defensa de los programones correspondientes. En caso de que el ataque o la defensa sea especial, se ocupan dichos valores.

[3]Entiendase otro stat como ataque , defensa, speed, etc

- Base: poder base del ataque

Y Modificador se compone de lo siguiente:

$$Modificador = STAB \times Tipo \times Critico \times random \in [0{,}85; 1]$$

Donde:

- STAB (same-type attack bonus): Este valor sera equivalente a 1.5, si el ataque seleccionado es del mismo tipo que el programón, 1 en otro caso.
- Tipo: Este valor puede ser 0, 0.5, 1 o 2. Depende del tipo del ataque y el tipo del programón defensor.
- Crítico: 2 en caso de ocurrir un crítico, 1 en otro caso. Para calcular la ocurrencia de un ataque critico:
$$T = \frac{BaseSpeed}{2}$$
Se genera un número aleatorio P entre 0 y 256. Luego, si $P \leq T$, el ataque es crítico.
- Random: Es un número aleatorio entre 0,85 y 1,00.

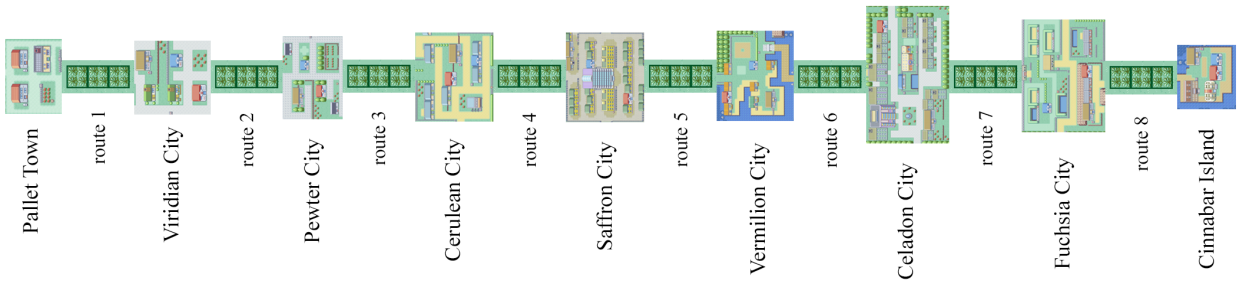
Nota: es posible hacer cero daño.

- Si se encuentra en una batalla contra un programón salvaje (en la hierba), se da la posibilidad de tratar de capturarlo lanzándole una prograbola, y la probabilidad de atraparlo esta dada por:
$$0{,}2 + \frac{HP_{total} - HP_{actual}}{HP_{total}} \times 0{,}8$$
Donde:
  - HP_{total}: Equivale al HP total del programón a capturar.
  - HP_{actual}: Equivale al HP restante del programón a capturar.

  No es posible capturar a los programones que se encuentran en los gimnasios.
- Término de Batalla: una batalla se da por finalizada cuando el programón o todos los programones del equipo del oponente o del propio llegan a hp cero o se quedan sin movimientos (dependiendo si es un programón salvaje que se encuentra en la hierba o si está en un gimnasio). Al ganar, todos los programones que participaron de la batalla (si salieron de la prograbola) automáticamente suben de nivel en una unidad y el jugador gana 200 yenes. En el caso de que se enfrente ante el líder del gimnasio y gane, además de ganar dinero y subir de nivel, gana una medalla. En el caso de perder la batalla, pierde 100 yenes. Pueden asumir que luego de la batalla, los pp de cada movimiento y los stats de cada programón vuelven al máximo (como estaban antes de la batalla). Además, si se gana la batalla, se debe calcular el nuevo hp y las nuevas stats por las fórmulas dadas en la sección 4. Si un entrenador pierde y no tiene suficiente dinero para pagar los 100 yenes, queda con salgo negativo.

6. Mapa

Entre cada ciudad existen 3 cuadros de hierba en donde pueden aparecer los programones salvajes. En el archivo routes.json se encuentra la ruta que une a la ciudad inicial con la ciudad final y el nivel de los programones salvajes que pueden aparecer en cada ruta. Para ir de una ciudad a otra, se deberá seleccionar la opción caminar 4 veces (ciudad-hierba, hierba-hierba, hierba-hierba y hierba-ciudad).



En cada ruta existe la probabilidad de que aparezca un programón en un rango de nivel, dado en el archivo routes.json. En una ruta puede aparecer cualquier programón con probabilidad uniforme con la restricción de que éste pueda ser encontrado en el rango de nivel de la ruta. En cada cuadro de hierba existe un 35 % de probabilidad de encontrarse con un programón salvaje y este puede ser cualquiera que cumpla con las condiciones descritas.

Ejemplo: Charmeleon evoluciona a Charizard en nivel 36, luego solo pueden aparecer Charizard's en una zona en donde el rango inferior sea mayor igual a 36 o en donde el rango de nivel lo contenga ejemplo [32 - 40], pero el nivel en el cual puede aparecer es entre 36 y 40.

7. Evolución programones

Dentro del archivo programones.json (explicado en la sección 9 más detalladamente), se encuentran dos atributos importantes por cada programón: `evolveLevel` y `evolveTo`. Una vez que el programón ha llegado a su evolveLevel (ya que ganó una cierta cantidad de batallas), el programón **debe** evolucionar al programón cuyo id es `evolveTo`. Se deben calcular los stats del nuevo programón con el nivel correspondiente a `evolveLevel`. Se debe **actualizar** la información del programón en el PC de Bastián y **agregar** éste a la progradex (se mantiene también el programón antes de la evolución en la progradex pero no en el PC de Bastián). Notar que no todos los programones evolucionan.

8. Diagrama de Clases

Junto con el programa pedido, se debe incluir en la carpeta de la tarea (Tareas/T01) el diagrama de clases con todo el modelamiento del problema. Esto incluye clases junto con sus métodos y atributos, y todas las relaciones existentes entre estas (asociación, composición y herencia).

9. Carga de Datos

Junto con el enunciado se encuentran muchos archivos de texto. Se detallará el contenido de cada uno a continuación:

- `programones.json`
  Contiene la información de los programones del juego.
  - id [int]: Id del programón.
  - name [str]: Nombre del programón.
  - speed [int]: Velocidad base.
  - hp [int]: Puntos de vida del programón.

- defense [int]: Estadística base que indica la defensa del programón.
- special_defense [int]: Estadística base que indica la defensa especial del programón.
- special_attack [int]: Estadística base que indica el ataque del programón.
- attack [int]: Estadística base que indica el ataque especial del programón.
- moves [list]: Lista que contiene el nombre de los movimientos que puede realizar el programón.
- type [str]: tipo de programón.
- evolveLevel [int]: nivel al que debe llegar el programón para evolucionar.
- evolveTo [int]: id del programón al que evoluciona el actual.

Attack y defense corresponde a movimientos de categoría "physical_moves", mientras que special attack y special defense corresponde a movimientos de categoría "special_moves".

- `programonesMoves.json`
  Contiene la información de los movimientos que pueden realizar los programones.
  - name [str]: Nombre del movimiento.
  - power [int]: Poder base del movimiento.
  - pp [int]: Puntos de poder del movimiento.
  - type [str]: Tipo de movimiento.
  - accuracy [float]: Exactitud del movimiento.

- `gyms.json`
  Contiene la información de las ciudades, los gimnasios y los entrenadores de cada uno.
  - city [string]: Nombre de la ciudad
  - id [int] : ID del gimnasio
  - leader [dict]
    - name [string]: Nombre del lider de gimnasio
    - programones [list]
      - id [int]: ID del programón correspondiente
      - level [int]: Nivel del programón
  - trainers [list]
    - name [string]: Nombre del entrenador del gimnasio
    - programones [list]
      - id [int]: ID del programón correspondiente
      - level [int]: Nivel del programón

- `types.json`
  Contiene información necesaria para el cálculo de daño en una batalla y está relacionada al tipo de ataque, tipo del programón que se defiende y por cuanto se ve afectado el daño final.
  - tipo_movimiento_atacante [dict]: Diccionario con todos los tipos de programones atacados que se ven afectados con el tipo del movimiento del atacante.
    - tipo_programón_atacado [float]: La key corresponde al tipo de programón atacado y el value corresponde al valor del tipo de ataque (corresponde a tipo dentro de la mórmula de modificador).

Nota: si la combinación tipo movimiento atacante - tipo programón atacado no se encuentra en este archivo, asuma que su valor es 1.

- `moveCategories.json`
  Contiene las categorías de los movimientos
  - physical_moves [list]: lista con los tipos de movimientos de esta categoría.
  - special_moves [list]: lista con los tipos de movimientos de esta categoría.
- `routes.json`
  Contiene la información de las ciudades, rutas y el nivel de los programones que pueden aparecer en cada ruta.
  - starting_point [str]: Ciudad de donde comienza la ruta.
  - destination [str]: Ciudad donde finaliza la ruta.
  - route [int]: Numero de ruta que conecta la ciudad de partida y la de destino.
  - levels [list]: lista que contiene el rango de niveles (incluidos) en los que puede aparecer un programón en dicha ruta.

Para cargar estos archivos en su programa, deberá crear un parser único que sea capaz de transformar el archivo de texto a un **dicionario de Python** para luego trabajar con este en la inicializacion de los distintos objetos que deberán modelar. Para este item, cuenta con el archivo `jsonReader.py` que incluye dos métodos:

- `jsonToDict(path)`: recibe como parámetro el path al archivo que se desea leer y retorna un diccionario.
- `dictToJson(path, data)`: recibe como parámetro el path al archivo que se desea crear o sobrescribir y un diccionario y genera este archivo con el contenido del diccionario.

Usted deberá crear un método equivalente a `jsonToDict` pero no puede utilizar la librería json (puede utilizar libremente el método `dictToJson`). Puede utilizar el método `jsonToDict` para probar su programa pero debe generar uno propio. Si no lo crea, no se entregará puntaje en este ítem. Debe detallar en su `README.md` como intercambiar este método (para probar con el que fue generado por usted y el entregado por nosotros).

Además de los archivos antes mencionados se les entrega un archivo nombrado `jugadoresEjemplo.json`, el cual les podría dar una idea de como guardar la información, pero recalcar que es a modo de ejemplo, pueden faltar datos que sean necesarios para la resolución de la tarea.

10. Restricciones y alcances

- Tu programa debe ser desarrollado en Python 3.5
- Esta tarea es estrictamente individual, y está regida por el Código de Honor de la Escuela: Clickear para Leer.
- Su código debe seguir la guía de estilos `PEP8`
- Si no se encuentra especificado en el enunciado, asuma que el uso de cualquier librería Python está prohibido. Pregunte por foro si se pueden usar librerías específicas.
- El ayudante puede castigar el puntaje[4] de tu tarea, si le parece adecuado. Se recomienda ordenar el código y ser lo más claro y eficiente posible en la creación algoritmos.

[4]Hasta −5 décimas.

- La revisión de la tarea será realizada con distintos archivos `.txt`.
- Debe adjuntar un archivo `README.md` donde comente sus alcances y el funcionamiento de su sistema (*i.e.* manual de usuario) de forma *concisa* y *clara*.
- Cree un módulo para cada conjunto de clases. Divídalas por las relaciones y los tipos que poseen en común. **Se descontará hasta un punto si se entrega la tarea en un solo módulo[5].**
- Cualquier aspecto no especificado queda a su criterio, siempre que no pase por sobre otro.

11. Entrega

- **Fecha/hora:** 26 de Agosto del 2016, 23:59 horas.
- **Lugar:** GIT - Carpeta: Tareas/T01

Tareas que no cumplan con las restricciones señaladas en este enunciado tendrán la calificación mínima (1.0).

[5]No agarre su código de un solo y lo divida en dos módulos, module su código de forma inteligente