Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2233 Programación Avanzada (2022-2)

Tarea 1

Entrega

- **Avance de tarea**
  - **Fecha y hora:** martes 6 de septiembre de 2022, 20:00
  - **Lugar:** Repositorio personal de GitHub — Carpeta: `Tareas/T1/`
- **Tarea**
  - **Fecha y hora:** martes 13 de septiembre de 2022, 20:00
  - **Lugar:** Repositorio personal de GitHub — Carpeta: `Tareas/T1/`
- `README.md`
  - **Fecha y hora:** martes 13 de septiembre de 2022, 22:00
  - **Lugar:** Repositorio personal de GitHub — Carpeta: `Tareas/T1/`

Objetivos

- Aplicar conceptos de programación orientada a objetos (POO) para modelar y resolver un problema.
- Utilizar *properties*, clases abstractas y polimorfismo como herramientas de modelación.
- Comunicar diseños orientados a objetos a través de documentación externa.
- Procesar *input* del usuario de forma robusta, manejando potenciales errores de formato.

Índice

1. *DCCampeonato Programón*

Tras unos duros días **programando la Tarea 0**, y posteriormente **festejar, en la Gala,** que lograste proteger exitosamente el Templo de maestros ayudantes de las bestias Nexus, el malvado Canciller Hernan4444 ha abandonado sus planes de convertirse en Emperador. Humillado por el fracaso de su plan, Hernan4444 decide abandonar este mundo y viajar a una dimensión paralela (la magia del ~~isekai~~[1] cine), la cual justamente era un mundo donde las personas suelen salir de sus casas a lo 10 años para atrapar y entrenar "mascotas", y decir que sus sueños es ser Maestro Programón.

Estando en este nuevo mundo, Hernan4444 decide desafiar a tu yo y los ayudantes de esta dimensión, a un *DCCampeonato Programón* para coronarse como campeón y pasar al Hall de la Fama del DCC junto a personajes ilustres como el gran maestro Cruz, y las leyendas Hadani, Brujaviera, Vicho, Blanca, entre otros, recordando las viejas glorias de la **AB 2019-2**. Sabiendo esto, decides viajar también a la nueva dimensión a ayudar a tu yo Maestro Programón de este mundo. Para prepararlo en esta competencia, decides crear un programa que simule un *DCCampeonato Programón* y así practicar todo lo posible.

Figura 1: Logo del *DCCampeonato Programón*

2. Flujo del programa

El juego consiste en simular un *DCCampeonato Programón*, donde los mejores 16 entrenadores programón compiten por el título de campeón. Deberás modelar todas las entidades involucradas y posteriormente simular rondas del campeonato, entrenamiento de tus programones, creación y uso de objetos, entre otros.

La interacción con el programa será exclusivamente mediante Menús impresos en la consola, en los cuales podrás tomar distintas decisiones. Como entrenador tendrás **energía**, la cuál podrás usar para **crear objetos** de distintos tipos y luego **utilizarlos** en alguno de tus programones para aumentar algunos de sus atributos. También tendrás la opción de utilizar tu **energía** para entrenar a tus programones y aumentar su **experiencia**, mejorando así sus posibilidades de ganar. Finalmente podrás elegir uno de tus programones para **luchar** en una ronda del *DCCampeonato Programón*. En caso de **perder** la ronda, se debe notificar a tu entrenador y dar la opción de comenzar nuevamente con el Menú de Inicio o salir del programa. En caso de **ganar** pasarás a la siguiente ronda junto al resto de entrenadores victoriosos. En ambos casos, después de cada ronda los entrenadores recuperan toda su **energía**. El objetivo es ganar el *DCCampeonato Programón*, por lo que necesitas planificar tu estrategia para vencer a los demás entrenadores programón a partir de las opciones entregadas.

Al iniciar el programa se abrirá el Menú de Inicio, donde el jugador puede seleccionar al entrenador que utilizarás en el *DCCampeonato Programón*. Cada entrenador tendrá uno o más programones, los cuales podrán usar durante el juego. Cada programón puede ser de tipo **Agua**, **Fuego** o **Planta**, y tendrán

[1] Si quiere saber que es isekai vea el siguiente **enlace**

distintos atributos, lo que afectará en la probabilidad de ganar una batalla. Luego de seleccionar un entrenador se mostrará el Menú Entrenador donde el jugador tendrá la opción de **entrenar**, **crear o usar objetos**, **simular una ronda**, y **ver el estado** de su entrenador y del campeonato. Por último, al momento de cumplir tu objetivo y ganar la ronda final del *DCCampeonato Programón* o en caso de perder, se deberá mostrar un mensaje informando la victoria o derrota y se debe reiniciar completamente el programa mostrando nuevamente el Menú de Inicio.

3. Menús

Para esta tarea, la simulación del *DCCampeonato Programón* será realizada a través de una serie de Menús en la consola. Estos menús deben ser a prueba de **todo tipo de errores de usuario**, es decir, en caso de que se ingrese un input no válido, se deberá advertir correctamente al usuario y volver a desplegar las opciones del menú. Adicionalmente, cada uno debe tener la opción de **volver atrás y salir**, a menos que se diga lo contrario. A continuación se explicarán los menús mínimos a incluir.

3.1. Menú de Inicio

Al iniciar el programa se deberá desplegar todas las opciones de entrenadores ubicados en el archivo `entrenadores.csv`, seguido de sus **programones asociados**. Además, se deberá dar la opción de escoger al entrenador que te representará en el *DCCampeonato Programón*. Dado que es el primer menú que se muestra en pantalla, **no es necesario implementar la opción de volver atrás**

```
          *** Menu de inicio ***
-------------------------------------------------
[1] Juan Vaughn: Goldeen, Growlithe
[2] Ricardo Walters: Wailmer, Bellsprout, Lotad
[3] Kaylee Robinson: Moltres, Luvdisc
[4] Amy Solomon: Bulbasaur
[5] Salir
```

Una vez cumplido lo anterior, se dirigirá el usuario al Menú Entrenador.

3.2. Menú Entrenador

En este menú se deberán encontrar todas las acciones ha realizar en el *Campeonato programón*. Esta sección debe tener seis opciones principales: **Menú de entrenamiento**, **Simulación ronda del campeonato**, **Mostrar estado del campeonato**, **Menú crear objeto**, **Menú utilizar objeto** y **Ver estado del entrenador**.

```
  *** Menú Entrenador ***
---------------------------
[1] Entrenamiento
[2] Simular ronda
[3] Resumen campeonato
[4] Crear objetos
[5] Utilizar objeto
[6] Estado entrenador
[7] Volver
[8] Salir
```

3.2.1. Menú de entrenamiento

Al seleccionar esta opción, se deberá desplegar todos los nombres de los programones que tiene tu entrenador. Al momento de seleccionar al programón que se desee entrenar, se le descontará al entrenador una cantidad `ENERGIA_ENTRENAMIENTO` de energía. Al realizar esto, la experiencia del programón elegido va a aumentar en un entero aleatorio entre `MIN_AUMENTO_EXPERIENCIA` y `MAX_AUMENTO_EXPERIENCIA`[2]. Si el programón llega a tener una experiencia de 100, entonces aumenta su nivel en 1, su experiencia vuelve a 0 y se le suma lo restante de lo que había ganado. Además, los atributos de vida, ataque, defensa y velocidad del programón cambian de acuerdo a lo mencionado en la sección de Programón. En caso de no contar con energía suficiente no se lleva a cabo el entrenamiento y se deberá notificar en consola.

```
 *** Menú de entrenamiento ***
--------------------------------
   [1] Blastoise
   [2] Charizard
   [3] Squirtle
   [4] Ivysaur
   [5] Volver
   [6] Salir
```

Luego de mostrar en consola **como mínimo los cambios** en la energía del entrenador y la experiencia del programón, se debe **volver al Menú Entrenador**.

3.2.2. Simulación ronda del campeonato

Para ganar el *DCCampeonato Programón*, **el jugador debe simular y ganar 4 rondas**[3]. **Al elegir esta opción, simularás sólo una ronda**. Primero debes mostrar una lista de los programones que tu entrenador tiene disponible, para seleccionar con cuál quieres luchar, de la siguiente manera:

```
   *** Elige tu luchador ***
--------------------------------
   [1] Blastoise
   [2] Charizard
   [3] Squirtle
   [4] Ivysaur
   [5] Volver
   [6] Salir
```

El resto de los entrenadores peleará con uno de sus programones elegido aleatoriamente. Luego, se debe **agrupar aleatoriamente en pares** a los entrenadores que quedan en la ronda, entre los cuales sucederán las luchas. Para cada lucha, se debe calcular quién gana comparando los resultados obtenidos para cada programón utilizando la **fórmula** especificada en Programón, ya que quien obtenga el **mayor valor** utilizando dicha fórmula será el ganador. Al final de cada ronda, se **restaura al máximo la energía** de todos los entrenadores asociados. Se debe imprimir en la consola el **número de ronda**, los **pares de entrenadores que se enfrentaron**, los **programones que utilizo cada uno** y el **ganador de cada lucha**. Puedes incluir más información que consideres relevante.

[2] Asume que `MIN_AUMENTO_EXPERIENCIA` es menor a `MAX_AUMENTO_EXPERIENCIA`

[3] Puedes pensar esto como octavos de final y asumir que siempre habrán 16 entrenadores compitiendo, incluyendo el que se utiliza

```
                                Ronda 1
--------------------------------------------------------------------------------
Juan Vaughn usando al programón Goldeen, se enfrenta a Ricardo Walters
usando al programón Wailme.
Ricardo Walters ha ganado la batalla.
Kaylee Robinson usando al programón Moltres, se enfrenta a Amy Solomon
usando al Bulbasaur.
Kaylee Robinson ha ganado la batalla.
```

Luego de mostrar lo anterior, en caso de quedar más rondas, se volverá al Menú Entrenador. En otro caso, se avisa el resultado, se reinicia el programa y se vuelve al Menú de Inicio.

3.2.3. Mostrar estado del campeonato

Al seleccionar esta opción, se deberá desplegar en consola toda la información relevante del campeonato hasta el momento. Esta deberá incluir como mínimo: los **entrenadores participantes**, la **ronda actual** y los **entrenadores que siguen participando**. Puedes incluir más información que consideres relevante.

```
                            Resumen Campeonato
--------------------------------------------------------------------------------
Participantes: Juan Vaughn, Ricardo Walters, Kaylee Robinson, Amy Solomon
Ronda actual: 2
Entrenadores que siguen en la competencia: Ricardo Walters, Kaylee Robinson
```

3.2.4. Menú crear objeto

En este menú se deben desplegar los 3 tipos de objetos existentes especificados en la sección Objetos. Cada objeto tiene asociado una cantidad de energía que se debe gastar por intentar crearlo y una probabilidad de que este se logre crear, según su tipo [4]. Todos estos datos se encuentran en la sección Objetos. Luego de haber elegido el tipo de objeto y si la probabilidad fue exitosa, a tu entrenador se le asigna aleatoriamente un objeto del tipo seleccionado de los que se encuentran en el archivo `objetos.csv`.

```
  *** Menú Objetos ***
--------------------------
[1] Baya
[2] Poción
[3] Caramelo
[4] Volver
[5] Salir
```

3.2.5. Menú utilizar objeto

En este menú se deben desplegar todos los objetos disponibles que tenga tu entrenador.

```
  *** Objetos disponibles ***
--------------------------
[1] Raro
[2] Aranja
```

[4] Para ver si la probabilidad fue exitosa o no, puedes hacer uso del método random.random y comparar el valor con la probabilidad de éxito

```
[3] Meloc
[4] Antiparalizante
[5] Mente
[6] Volver
[7] Salir
```

Al seleccionar uno serás dirigido a un menú donde se muestren los programones de tu entrenador (tal como en el Menú de entrenamiento) y seleccionarás el programón sobre el cual quieras utilizar el objeto. Se debe mostrar en consola los efectos producidos por el objeto sobre el programón.

```
Programón beneficiado: Blastoise
Objeto utilizado: Raro (Tipo caramelo)
Aumento vida: 9
La vida subió de 51 a 60
Aumento ataque: 6
El ataque subió de 47 a 53
Aumento defensa: 10
La defensa subió de 100 a 110
```

3.2.6. Ver estado del entrenador

Al seleccionar esta opción, se deberá desplegar en consola toda la información relevante de tu entrenador. Esta deberá incluir como mínimo el **nombre del entrenador**, **energía**, **objetos** y la información de los programones de tu entrenador, incluyendo el **nombre**, **tipo**, **nivel** y **vida** de cada programón. Puedes incluir más información si lo consideras relevante.

```
            *** Estado entrenador ***
-----------------------------------------------
Nombre: Juan Vaughn
Energía: 32
Objetos: aranja, persai, vigor
-----------------------------------------------
                   Programones
-----------------------------------------------
   Nombre    |    Tipo   |   Nivel  |   Vida
-----------------------------------------------
 Blastoise   |    Agua   |    47    |    60
 Charizard   |   Fuego   |    76    |    201
  Squirtle   |    Agua   |    82    |    243

[1] Volver
[2] Salir
```

Cabe recordar que todos los menús presentados en las imágenes anteriores son de **referencia**, por lo cual puedes realizar todas las modificaciones que prefieras, mientras cumplas con lo **mínimo que pide el enunciado**.

4. Entidades

En esta sección se detallarán las entidades que necesitarás para simular *DCCampeonato*. Deberás utilizar conceptos claves de **Programación Orientada a Objetos** como herencia, clases abstractas, polimor-

fismo, *properties* y relaciones, que pueden ser de agregación o composición. Ten en cuenta, que cada uno de estos elementos debe ser incluido en el programa al menos una vez, y deberás descubrir dónde implementarlos según lo propuesto por el enunciado.

Las entidades principales de *DCCampeonato* son **Liga Programón, Programón, Entrenador y Objeto**. Debes incluir como mínimo las características nombradas a continuación, pero siéntete libre de añadir nuevos atributos y acciones si lo estimas necesario.

A continuación, se mostrarán las entidades de ***DCCampeonato Programón***.

4.1. Liga Programón

Esta entidad corresponde al mapa general en que se desarrollará el *DCCampeonato*, más específicamente la opción de batalla. Aquí se tendrán los entrenadores que compiten, la ronda actual, los perdedores y el ganador de la liga.

- **Entrenadores**: Corresponde a todos los entrenadores que compiten en el *DCCampeonato*.
- **Perdedores**: Corresponde a todos los entrenadores que han perdido en el *DCCampeonato*.
- **Ronda actual**: Es un número que indica la ronda que se está llevando a cabo.
- **Campeón**: Corresponde al entrenador, que resulta ganador del *DCCampeonato*.

Por otro lado, a partir de esta entidad se realizan diferentes acciones:

- **Resumen Campeonato**: Muestra en consola los participantes, la ronda actual y los entrenadores que siguen en competencia. Esto se muestra en Mostrar estado del campeonato.
- **Simular ronda**: Se encarga de simular la ronda y mostrar en consola el resumen de esta. Esto está especificado en Simulación ronda del campeonato.

4.2. Programón

Los programones son criaturas que viven en el mundo programón (valga la redundancia) y que diversos entrenadores los atrapan para lograr ♪ser siempre el mejor, mejor que nadie más♪. Dentro de la simulación, los programones tendrán diversas características, algunas definidas en el archivo `programones.csv` y otras de acuerdo a este enunciado. A continuación, se presentarán las características del programón:

- **Nombre**: Corresponde al nombre de la criatura, que se encuentra en el archivo `programones.csv`.
- **Nivel**: Corresponde a un número, entre `1` y `100` que afectará el puntaje de victoria del programón.
- **Nivel de megaevolución (Bonus)**: Corresponde a un número entre `1` y `100` que se usa durante la implementación del bonus (No es necesario agregarlo si no se realiza). Es el nivel en que el programón mega evolucionará y se encuentra en el archivo `evoluciones.csv`.
- **Experiencia**: Corresponde a un número, entre `0` y `100`, al comenzar el programa este valor comienza en `0`. Este atributo sirve para aumentar el nivel durante el entrenamiento. Si al sumar un valor de experiencia, el programón fuera a tener una experiencia mayor o igual a `100`, entonces se aumentará el nivel del programón en `1`, su experiencia vuelve a `0` y se le sumará lo sobrante de lo que había ganado. En otras palabras, por cada 100 unidades de experiencia se debe subir un nivel consumiendo dicha experiencia. Por cada nivel que se suma, se deberá sumar a los atributos de *Vida, Ataque, Defensa y Velocidad* un valor entre `MIN_AUMENTO_ENTRENAMIENTO` y `MAX_AUMENTO_ENTRENAMIENTO` aleatoriamente.[5]. Si el nivel llega a `100`, la experiencia quedará en `0` y el entrenamiento ya no hará efecto.

[5]Considera que `MIN_AUMENTO_ENTRENAMIENTO` es menor a `MAX_AUMENTO_ENTRENAMIENTO`

- **Tipo**: Corresponde al nombre del tipo del programón y cómo esto afectará su rendimiento durante la batalla.
- **Vida**: Corresponde a un número, entre `1` y `255` que corresponde a los puntos de salud del programón. Se puede ver afectada por los objetos.
- **Ataque**: Corresponde a un número entre `5` y `190` que puede afectar en el calculo de la probabilidad de ganar. Se puede ver afectado por los objetos.
- **Defensa**: Corresponde a un número entre `5` y `250`. Este se ocupa para calcular la probabilidad de ganar. Se puede ver afectado por los objetos.
- **Velocidad**: corresponde a un número entre `5` y `200`. Este se ocupa para calcular la probabilidad de ganar.

Además, el programón se va a ver involucrado en las siguientes mecánicas del juego:

- **Entrenamiento**: El programón va a entrenar sus capacidades y va a poder aumentar su experiencia aleatoriamente entre `MIN_AUMENTO_EXPERIENCIA` y `MAX_AUMENTO_EXPERIENCIA` puntos.
- **Luchar**: En esta parte, se deberá calcular el *Puntaje de Victoria*. Este se comparará con el programón del rival al momento de realizar la Simulación ronda del campeonato y el que tenga el mayor puntaje gana.

4.2.1. Tipos Elementales

Los programones sólo pueden poseer tres tipos: Planta, Fuego o Agua. Cada uno de ellos tiene una ventaja o desventaja sobre los otros. Esto se enuncia a continuación:

- **Tipo Planta**: Son especies que habitan en lugares apacibles y tranquilos, rodeados de naturaleza. Por ser de este tipo, al ganar una batalla, su vida aumentará en `AUMENTAR_VIDA_PLANTA`. Es débil contra el tipo fuego y fuerte contra el tipo agua.
- **Tipo Fuego**: Estos pokémon habitan en lugares cálidos o cerca de volcanes o fuentes de magmatismo. Por otra parte, sus ataques pueden generar quemaduras. Cada vez que gane una batalla, su ataque aumentará en `AUMENTAR_ATAQUE_FUEGO`. Es débil contra el tipo agua y fuerte contra el tipo planta.
- **Tipo Agua**: Corresponde al tipo elemental con más especies dentro del mundo programón. La mayoría de ellos vive cerca de fuentes de agua. Cada vez que gane una batalla, su velocidad aumentará en `AUMENTAR_VELOCIDAD_AGUA`. Es débil contra el tipo planta y fuerte contra el tipo fuego.

Para que un programón gane, se va a usar el **Puntaje de victoria**, que corresponde a un número. El programón que posea la variable más alta va a ganar la ronda. Por otra parte, si el valor de este parámetro entre los dos programones es igual, el que gane la batalla será escogido aleatoriamente.

$\texttt{max}(0, (Vida * \texttt{0.2} + Nivel * \texttt{0.3} + Ataque * \texttt{0.15} + Defensa * \texttt{0.15} + Velocidad * \texttt{0.2} + \text{Ventaja de tipo} * \texttt{40}))$

- **Ventaja de tipo**: Corresponde a una variable que depende de la ventaja o desventaja del tipo del programón sobre el que está luchando. Si se tiene ventaja, esta corresponderá a `1`, si es del mismo tipo será `0`, y si se tiene desventaja, la variable tendrá un valor de `-1`

4.3. Entrenador

Los entrenadores en el *DCCampeonato* son los encargados de hacer que los programones logren su máximo potencial y enfrenten de mejor manera las batallas. Todo lo necesario para modelarlos se encuentra en `entrenadores.csv`.

- **Nombre**: Corresponde al nombre del entrenador.
- **Energía**: Corresponde a un número entre `0` y `100` que representa la energía del entrenador, será ocupada para crear objetos.
- **Programones**: corresponde a los programones que posee el entrenador.
- **Objetos**: corresponde a los objetos que posee el entrenador.

Las acciones que puede realizar el entrenador son las siguientes:

- **Estado entrenador**: Muestra en consola el nombre, energía, objetos y programones del entrenador junto a información importante de cada uno.
- **Crear objetos**: Con esta acción el entrenador buscar crear los objetos, para esto gasta `ENERGIA_CREAR_OBJETO` dependiendo del objeto. La creación o no de los objetos depende de `PROB_EXITO_OBJETO`

4.4. Objetos

Los objetos en *DCCampeonato* juegan un rol influyente en las batallas, ya que estos te entregan un aumento en la vida, el ataque y/o defensa del programón . Los objetos pueden ser de tipo **Bayas, Pociones o Caramelos**. Por último, los atributos de esta entidad se encuentran en `objetos.csv` y son:

- **Nombre**: Representa el nombre del objeto.
- **Tipo**: Corresponde al tipo de objeto que se le esta dando al programón que puede ser **Bayas, Pociones o Caramelos**.
- **Costo**: Corresponde a un número que representa la energía que ocupa el entrenador para intentar crear el objeto.
- **Probabilidad de éxito**: Corresponde a un número que representa la probabilidad que tiene el entrenador de crear exitosamente el objeto.

Además, deberás aplicar su beneficio al programón mediante la siguiente acción.

- **Aplicar objeto**: Esta acción aplica los beneficios del objeto al programón escogido. Estos beneficios **dependerán** del **tipo** de objeto.

4.4.1. Tipos de Objetos

Además, existen distintos tipos de objetos, cada una contribuyendo con distintas cualidades al programon.

- **Baya**: Las bayas son un tipo de objeto muy natural, y por esta razón generalmente entregan muchos nutrientes a los programones que afectan subiendo la vida de estos. El aumento de la vida deberá ser un valor al azar entre `1` y `10`. La vida nunca debe superar los 255 puntos. Crear este objeto gasta `GASTO_ENERGIA_BAYA` de energía y tiene una probabilidad de éxito de `PROB_EXITO_BAYA`.
- **Poción**: Son objetos que crean los entrenadores con distintos líquidos, tienen sabores muy diversos y que alegran a los programones, lo que genera que aumenten su ataque. El aumento del ataque deberá ser un valor al azar entre `1` y `7`. El ataque nunca debe superar los `190` puntos. Crear este objeto gasta `GASTO_ENERGIA_POCION` de energía y tiene una probabilidad de éxito de `PROB_EXITO_POCION`.

- **Caramelo**: Es la mezcla de las bayas y las pociones con un ingrediente especial. Estos generan mucha alegría en los programones, lo que sumado a los muchos nutrientes que tienen, genera un aumento en la vida, en el ataque y la defensa. Los efectos que se aplican son tanto los de **Baya** como los de **Poción** y también habrá un aumento en la defensa de `AUMENTO_DEFENSA`. Crear este objeto gasta `GASTO_ENERGIA_CARAMELO` de energía y tiene una probabilidad de éxito de `PROB_EXITO_CARAMELO`

5. Archivos

Los siguientes archivos contienen la base de datos de los programón, entrenadores y los tipos de objetos de *DCCampeonato Programón*. Todos estos archivos los necesitarás para tu programa. Podrás notar que cada uno de los siguientes archivos viene con un encabezado (*header*) en la primera línea que indica a cuál columna corresponde cada uno de los elementos de las siguientes filas, separadas por comas (`","`). Puedes asumir que el orden de las columnas no cambiará y que cada archivo es consistente con sus datos, por lo que cada entrenador tendrá cierta cantidad de programones pertenecientes al archivo `programones.csv`, además de que cada objeto de cada entrenador también estará presente en el archivo `objetos.csv`.

5.1. `programones.csv`

Este archivo contiene todos los atributos de los diferentes programones, junto con sus características que son descritas en la siguiente tabla:

Nombre	Tipo de Dato	Descripción
nombre	`str`	Indica el nombre del programon.
tipo	`str`	Corresponde al tipo de elemento que tiene el programon.
nivel	`int`	Nivel actual del programon con un rango de 1 y 100.
vida	`int`	Corresponde a la salud actual del programon, con un rango de 1 a 255.
ataque	`int`	Corresponde al ataque del programon y va entre 5 y 190
defensa	`int`	Corresponde a la defensa del programon, con un rango de 5 a 250.
velocidad	`int`	Corresponde a la velocidad que posee el programon, va entre 5 y 200.

Un ejemplo del archivo `programon.csv` es el siguiente:

```
nombre,tipo,nivel,vida,ataque,defensa,velocidad
Blastoise,agua,47,60,110,96
Charizard,fuego,76,201,122,129,44
Squirtle,agua,82,243,91,131,27
Ivysaur,planta,29,35,68,156,200
```

5.2. `entrenadores.csv`

Este archivo contiene todos los atributos de cada entrenador y una lista de programones que deben ser manejadas para ser asignadas a cada entrenador. El archivo siempre contendrá 16 entrenadores, por lo que siempre podrás agruparlos para llevar a cabo las luchas. A continuación están las descripciones de todas las características:

Nombre	Tipo de Dato	Descripción
nombre	`str`	Indica el nombre del entrenador.
programones	`list`	Indica todos lo programón que el entrenador posee, separados por un `";"` entre cada uno.
energía	`int`	Indica la energía del entrenador, entre 0 y 100.
objetos	`list`	Corresponde a todos los objetos que posee el entrenador, separados con un `";"`.

Un ejemplo del archivo `entrenadores.csv` es el siguiente:

```
nombre,programones,energia,objetos
Juan Vaughn,Lickitung;Cubone;Voltorb;Gengar,32,aranja;perasi;vigor
Ricardo Walters,Omanyte;Articuno;Dragonair,76,raro;pocion
Kaylee Robinson,Totodile;Furret;Aipom,45,antiquemar;zidra;impetu
```

5.3. `objetos.csv`

Este archivo contiene todos los objetos, por lo que en la siguiente tabla se muestra su contenido y descripción:

Nombre	Tipo de Dato	Descripción
nombre	`str`	Indica el nombre del objeto.
tipo	`str`	Indica el tipo del objeto.

Un ejemplo del archivo `objetos.csv` es el siguiente:

```
nombre,tipo
raro,caramelo
aranja,baya
meloc,baya
antiparalizante,pocion
mente,caramelo
musculo,caramelo
```

5.4. `evoluciones.csv`

Este archivo contiene las evoluciones de los programones pertenecientes al archivo `programon.csv`, además del nivel necesario para evolucionar.[6] Deberás usarlo **sólo si realizas el bonus**. En la siguiente tabla se muestra su contenido y descripción:

Nombre	Tipo de Dato	Descripción
nombre	`str`	Indica el nombre del programon.
nivel	`int`	Indica el nivel necesario a alcanzar para evolucionar.
evolucion	`str`	Indica el nombre de la evolucion.

[6] Este archivo se deberá utilizar exclusivamente si se realiza el Bonus

Un ejemplo del archivo `evolucion.csv` es el siguiente:

```
nombre,nivel,evolucion
Blastoise,3,MegaBlastoise
Charizard,4,LeyendCharizard
Ivysaur,3,MegaIvysaur
```

5.5. `parametros.py`

Para esta tarea, deberás crear un archivo `parametros.py` y **completarlo con todos los parámetros mencionados a lo largo del enunciado**, los cuales encontrarás en `ESTE_FORMATO` y en ese color. Además, debes agregar cualquier valor constante en tu tarea, junto con los *paths* que utilices.

Recuerda que los parámetros deben ser descriptivos y reconocibles:

```
CIEN = 100 # mal parámetro
DEUDA_TOTAL = 3000 # buen parámetro
PROBABILIDAD_EVENTO = 0.2 # buen parámetro
```

Si necesitas agregar algún parámetro que varíe de acuerdo a otros parámetros, una correcta parametrización sería la siguiente:

```
PI = 3.14
RADIO_ESFERA = 3
VOLUMEN_ESFERA = (4/3) * PI * (RADIO_ESFERA ** 3)
```

Dentro del archivo `parametros.py` deberás hacer uso de todos los parámetros almacenados y deberás importarlos correctamente. Si se almacena cualquier información no correspondiente a parámetros, esto tendrá un descuento en tu nota. Por último, no está permito que un parámetro se encuentre ***harcodeado***[7], ya que es una mala práctica y su uso conlleva un descuento.

Para esta tarea, el archivo `parametros.py` no se debe ignorar y debes subirlo a tu repositorio. En caso contrario, tu tarea no se podrá corregir.

6. *Bonus*

En esta tarea habrá una serie de *bonus* que podrás obtener. Cabe recalcar que necesitas cumplir los siguientes requerimientos para poder obtener *bonus*:

1. La nota en tu tarea (sin bonus) debe ser **igual o superior a 4.0**[8].
2. El bonus debe estar implementado **en su totalidad**, es decir, **no se dará puntaje intermedio**.

Finalmente, la cantidad máxima de décimas de *bonus* que se podrá obtener serán 5 décimas. Deberás indicar en tu `README` si implementaste alguno de los bonus, y cuáles fueron implementados.

6.1. Mega Evolución (3 décimas)

Para optar a este bonus se debe implementar una nueva acción durante el entrenamiento. Para esto, cuando un programón llegue o supere al *Nivel de mega evolución*, indicado en el archivo `evoluciones.csv`, este va

[7] *Hard-coding* es la mala práctica de incrustar datos directamente en el código fuente del programa, en vez de obtener los datos de una fuente externa.

[8] Esta nota es sin considerar posibles descuentos.

a convertirse en su evolución. En la práctica, todos los programones pueden mega evolucionar y su nombre deber ser cambiado, según lo indique el archivo antes mencionado. Esto ocurre solo una vez, se mantiene durante toda la simulación y sus características se verán aumentadas en los siguientes parámetros:

- **Vida**: Se verá aumentado respecto al parámetro `MEGA_VIDA`.
- **Ataque**: Se verá aumentado respecto al parámetro `MEGA_ATAQUE`.
- **Defensa**: Se verá aumentado respecto al parámetro `MEGA_DEFENSA`.
- **Velocidad**: Se verá aumentado respecto al parámetro `MEGA_VELOCIDAD`.

Si el programón ya tiene el nivel necesario para mega evolucionar al momento del inicio de la simulación, automáticamente deberá mega evolucionar y este cambio deberá ser mostrado en consola. Cabe recalcar que no hay que sobreescribir el archivo `entrenadores.csv` con la mega evolución. Esto sólo toma lugar durante la ejecución de la simulación.

6.2. CSV dinámicos (2 décimas)

Para optar a este bonus tu programa debe ser capaz de leer correctamente los archivos donde el orden de las columnas no esté fijo. En otras palabras, todos los archivos `.csv` pueden tener variación en sus columnas y el programa debe leer el archivo sin problema. Por ejemplo, el archivo `objetos.csv` puede ser el siguiente

```
nombre,tipo
raro,caramelo
aranja,baya
meloc,baya
```

o tambien puede ser como este otro archivo:

```
tipo,nombre
caramelo,raro
baya,aranja,
baya,meloc,
```

7. Diagrama de clase

En conjunto con el programa, deberás realizar un Diagrama de clase modelando las entidades necesarias del *DCCampeonato Programón*. Este diagrama se entregará en dos ocasiones:

La primera en el Avance de tarea, y la segunda en la entrega final. Para esta última, deberás entregar la **versión final** de tu diagrama que **represente fielmente** la modelación de clases implementada en tu programa.

En ambos casos, el diagrama deberá:

- Entregarse en **formato PDF o de imagen**[9].
- Contener todas las clases junto con sus atributos y métodos. También deberás identificar cuáles clases serán abstractas y cuáles no.
- Contener todas las relaciones existentes entre las clases (agregación, composición y herencia).

[9] Cualquier otro formato no será considerado como una entrega válida y no tendrá décimas de avance o puntaje en la tarea.

- No es necesario indicar la cardinalidad ni la visibilidad (público o privado) de los métodos o atributos.

Para realizar el diagrama de clases te recomendamos utilizar **draw.io**, **lucidchart** o aplicaciones similares.

Es conveniente adjuntar a tu diagrama un documento con una explicación general de tu modelación. Esto es con el fin de ayudar en la corrección del ayudante a comprender tu razonamiento.

Tanto el diagrama (en formato PDF o de imagen) como la explicación de su modelación (en formato **Markdown**) deben ubicarse en la misma carpeta de entrega de la tarea.

8. Avance de tarea

Para el Avance de tarea de tu tarea deberás realizar una versión preliminar del Diagrama de clase que refleje cómo planeas modelar tu programa. A partir de los avances entregados, se te brindará un *feedback* general de lo entregado y además, te permitirá optar por **hasta 2 décimas** adicionales en la nota final de tu tarea.

9. `.gitignore`

Para esta tarea **deberás utilizar un** `.gitignore` para ignorar los archivos indicados, este deberá estar dentro de tu carpeta `Tareas/T1/`. Puedes encontrar un ejemplo de `.gitignore` en el siguiente ***link***.

Para esta tarea, los archivos a ignorar corresponden a las bases de datos entregadas para la simulación. Es decir, deberás ignorar:

- `Enunciado.pdf`
- `programones.csv`
- `entrenadores.csv`
- `objetos.csv`
- `evoluciones.csv`

Se espera que no se suban archivos autogenerados por las interfaces de desarrollo o los entornos virtuales de Python, como por ejemplo: la carpeta `__pycache__`, la carpeta `.vscode`, entre otros.

Para este punto es importante que hagan un correcto uso del archivo `.gitignore`, es decir, los archivos no **deben** subirse al repositorio debido al archivo `.gitignore` y no debido a otros medios.

10. Entregas atrasadas

Posterior a la fecha de entrega de la tarea se abrirá un formulario de Google Form, en caso de que desees que se corrija un *commit* posterior al recolectado, deberás señalar el nuevo *commit* en el *form*.

El plazo para rellenar el *form* será de 24 horas, en caso de que no lo contestes en dicho plazo, se procederá a corregir el *commit* recolectado.

11. Importante: Corrección de la tarea

Para todas las tareas de este semestre deberás redactar un archivo `README.md`, un archivo de texto escrito en Markdown, que tiene por objetivo explicar su tarea y facilitar su corrección para el ayudante. Markdown es un lenguaje de marcado (como LaTeX o HTML) que permite crear un texto organizado y simple de leer. Pueden encontrar un pequeño tutorial del lenguaje en este ***link***.

Un buen `README.md` debe **facilitar la corrección de la tarea.** Una forma de lograr esto es explicar de forma breve y directa el **idioma** en qué programaste (puedes usar inglés o español) y **qué cosas fueron implementadas, y qué cosas no**, usualmente **siguiendo la pauta de evaluación**. Esto permite que el ayudante dedique más tiempo a revisar las partes de tu tarea que efectivamente lograste implementar, lo cual permite entregar un *feedback* más certero. Para facilitar la escritura del `README`, se entregarán algunos ejemplos de **plantillas** (*templates*) a rellenar con la información adecuada.

Como forma de motivarte a redactar buenos `README`s, todas las tareas tendrán **décimas de des-descuento** si el ayudante considera que tu `README` fue especialmente útil para la corrección. Estás décimas anulan décimas de descuento que les hayan sido asignadas hasta un máximo de cinco.

Finalmente, si durante la realización de tu tarea se te presenta algún problema o situación que pueda afectar tu rendimiento, no dudes en contactar a la ayudante jefa de Bienestar al siguiente correo: **bienestar.iic2233@ing.puc.cl**.

12. Restricciones y alcances

- Esta tarea es **estrictamente individual**, y está regida por el **Código de honor de Ingeniería**.
- Tu programa debe ser desarrollado en Python 3.10.
- Tu programa debe estar compuesto por uno o más archivos de extensión `.py`.
- Si no se encuentra especificado en el enunciado, supón que el uso de cualquier librería Python está prohibido. Pregunta en la *issue* especial del **foro** si es que es posible utilizar alguna librería en particular.
- Debes adjuntar un archivo `README.md` **conciso y claro**, donde describas los alcances de tu programa, cómo correrlo, las librerías usadas, los supuestos hechos, y las referencias a código externo. **Tendrás hasta 2 horas después del plazo de entrega** de la tarea para subir el `README` a tu repositorio.
- Tu tarea podría sufrir los descuentos descritos en la **guía de descuentos**.
- Entregas con atraso de más de 24 horas tendrán calificación mínima (1,0).
- Cualquier aspecto no especificado queda a tu criterio, siempre que no pase por sobre otro.

Las tareas que no cumplan con las restricciones del enunciado obtendrán la calificación mínima (1,0).