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

IIC2233 Programación Avanzada (2024-1)

Tarea 2

Entrega

- **Tarea y** `README.md`
  - **Fecha y hora oficial (sin atraso):** martes 23 de abril de 2024, 20:00
  - **Fecha y hora máxima (2 días de atraso):** jueves 25 de abril de 2023, 20:00.
  - **Lugar:** Repositorio personal de GitHub — Carpeta: `Tareas/T2/`.
    El código debe estar en la rama (*branch*) por defecto del repositorio: `main`.
  - **Pauta de corrección**: **en este enlace**.
  - **Bases generales de tareas (descuentos)**: **en este enlace**.
- **Ejecución de tarea**: La tarea será ejecutada **únicamente** desde la terminal del computador. Además, durante el proceso de corrección, se cambiará el nombre de la carpeta "`T2/`" por otro nombre y se ubicará la terminal dentro de dicha carpeta antes de ejecutar la tarea. **Los *paths* relativos utilizados en la tarea deben ser coherentes con esta instrucción.**

Objetivos

- Aplicar conceptos de programación orientada a objetos (POO) para modelar y resolver un problema complejo.
- Utilizar *properties*, clases abstractas y polimorfismo como herramientas de modelación.
- Procesar *input* del usuario de forma robusta, manejando potenciales errores de formato.
- Trabajar con archivos de texto para leer y procesar datos.
- Familiarizarse con el proceso de entrega de tareas y uso de buenas prácticas de programación.

Índice

1. *DCCombatientes*

Luego de haber ayudado a **Gatochico** a desarrollar el metro de la *DCCiudad* te pones a recorrer la ciudad, dándote cuenta que el metro no era el único problema. Hablando con los gatos chicos, llegas a la conclusión de que **Gatochico** no es una buena alcaldesa, pues ha saboteado la calidad de vida de sus gatos chicos. Afortunadamente, existe una resistencia, liderada por Crismiau Ruz, y formada por **Gatos Combatientes**, gatos chicos que se mueven por el bajo mundo tratando de aumentar sus números para causar una revuelta y lograr su independencia.

Al enterarse de tus magníficas capacidades de programación, te imploran crear un programa capaz de cambiar el destino de su ciudad, simulando la batalla que tendrán contra **Gatochico** para terminar con su gobierno tirano. ¡Ayuda a **Gatos Combatientes** a recuperar su calidad de vida!

Figura 1: Logo de *DCCombatientes*.

2. Flujo del programa

En esta tarea el jugador debe diseñar un ejército de *DCCombatientes* capaces de derrotar a los Ejércitos de **Gatochico**. Para esto, se debe comenzar por seleccionar una dificultad, la cual **debe** ser entregada al programa como **parámetro** por consola. Iniciado el programa, contarás con una cantidad de dinero igual a `ORO_INICIAL`[1], con el que puedes comprar gatos Combatiente e Ítemes para hacer más poderoso a tu Ejército, inicialmente vacío. Finalmente, cuando termines de armar tu Ejército gatuno, ¡podrás luchar contra los combatientes de **Gatochico**!

La única manera de interactuar con el programa es a través de la consola, mediante menús anidados. Primero accederás al Menú de Inicio, que contiene cuatro opciones: **Tienda**, **Ejercito**, **Combatir** y **Salir**.

Dentro de Menú de Tienda podrás encontrar la única manera de agregar nuevos miembros a tu ejército, comprando los distintos Tipos de combatientes, como **Gatos Guerreros**, **Magos** y **Caballeros**. Ade-

[1] A lo largo del enunciado encontrarás varias palabras escritas en `ESTE_FORMATO`. Estas palabras corresponden a parámetros y puedes encontrar los detalles sobre ellos en la sección `parametros.py`.

más, podrás comprar objetos de evolución de los **Gatos Combatientes**, como **Pergaminos**, **Lanzas** y **Armaduras**, los cuales permitirán que uno de tus **Gatos Combatientes** reciba características de un tipo distinto. Los detalles de cada combatiente y sus evoluciones se explican con mayor detalle en la sección Tipos de combatientes. Por último, la tienda tendrá una opción para curar la vida del ejercito, cobrando por realizar tal acción.

Por otro lado, al seleccionar la opción **Combatir**, se iniciará una ronda de Enfrentamientos contra **Gatochico**. Los gatos de cada equipo combatirán uno a uno en cada enfrentamiento, usando sus ataques contra su rival hasta que alguno de los dos no pueda continuar. Avanzarás a la siguiente ronda si consigues eliminar a todos los gatos enemigos. Ganarás el juego si logras completar 3 rondas de combate. Al contrario, si todos tus gatos son eliminados, perderás el juego y deberás volver a comenzar desde la primera ronda. En ambos casos, al finalizar el combate, volverás al Menú de Inicio, donde –en caso de una victoria– podrás continuar la partida actual, o –en caso de sufrir una derrota– podrás volver a comenzar desde la primera ronda, donde al igual que en un inicio, contarás con un ejército vacío y el dinero inicial.

Podrás revisar el estado actual de tu equipo seleccionando la opción **Ejército**, que mostrará todos tus **Gatos Combatientes** y sus respectivos *stats*. Finalmente, podrás terminar el programa seleccionando la opción **Salir**.

3. Menús

Dentro de esta increíble simulación *DCCombatientes* las interacciones del usuario se realizarán a través de menús. Tanto la interacción como el despliegue de los menús deberán ser mediante la consola. Un aspecto importante de los menús es que deben ser a prueba de **todo tipo de errores** de usuario, esto quiere decir que en caso de que el usuario ingrese un *input* inválido como respuesta, se deberá informar al usuario de que el *input* no es válido y se volver a desplegar las opciones del menú. A continuación se explicarán cada uno de los menús mínimos.

3.1. Menú de Inicio

Al iniciar el programa se deberá desplegar el Menú de Inicio. Dentro de este menú se deberá poner un encabezado que diga **Menú de Inicio**, seguido de un texto que indique el dinero disponible y el número de ronda actual. Luego de lo anterior, se desplegarán las opciones de **Tienda**, **Ejército**, **Combatir** y **Salir del programa**.

Al seleccionar la opción **Tienda**, se deberá desplegar un segundo menú, que se encuentra detallado en Menú de Tienda.

Por otro lado, al seleccionar **Ejército**, se deberá hacer `print` a esta clase, haciendo que el ejército se presente y mostrando en pantalla los detalles de cada unidad como se explica en la sección **Ejército**.

Al seleccionar la opción **Combatir**, se deberá iniciar la simulación del combate según se explica en Combates.

Finalmente, al elegir la opción **Salir del programa** se debe **detener la ejecución del programa**, imprimiendo un mensaje de salida para el usuario y cerrando el programa.

```
*** Menú de inicio ***

Dinero disponible: 5
Ronda actual: 1

[1] Tienda
[2] Ejercito
[3] Combatir

[0] Salir del programa

Indique su opción:
```

Figura 2: Ejemplo de Menú de Inicio

3.2. Menú de Tienda

En el Menú Tienda, se deberá mostrar primero un encabezado que diga **Tienda**, junto con la cantidad de dinero disponible. Luego, se deben desplegar todas las opciones de compra disponibles, junto con sus respectivos precios. La última opción de este menú será el volver al Menú de Inicio.

En la tienda se podrán comprar los 3 tipos básicos de **Gatos Combatientes** (Mago, Guerrero y Caballero), cada uno de los Ítemes (Armadura, Pergamino y Lanza), y la opción **Curar Ejército**. Los detalles sobre los Combatientes e Itemes se explican en las sección Entidades y los precios para cada una de estas opciones se corresponderán, respectivamente, con los parámetros `PRECIO_MAG`, `PRECIO_GUE`, `PRECIO_CAB`, `PRECIO_ARMADURA`, `PRECIO_PERGAMINO`, `PRECIO_LANZA` y `PRECIO_CURA`.

```
*** Tienda ***

Dinero disponible: 5

       Producto     Precio
[1] Gato Mago         10
[2] Gato Guerrero     10
[3] Gato Caballero    10
[4] Ítem Armadura      5
[5] Ítem Pergamino     5
[6] Ítem Lanza         5
[7] Curar Ejército     7

[0] Volver al Menú de inicio

Indique su opción:
```

Figura 3: Ejemplo de Menú de Tienda

Sin importar qué opción seleccione el usuario, o si la transacción fue realizada con éxito o falló, luego de cada interacción con la tienda se deberá regresar al Menú de Tienda, exceptuando la opción de volver al Menú de Inicio.

En caso de que el usuario seleccione uno de los **Gatos Combatientes** y cuente con el dinero suficiente, el monto deberá ser descontado del dinero total, se deberá elegir, de manera aleatoria, a un gato del tipo seleccionado desde el archivo `unidades.txt`, para luego agregar el combatiente al ejército. Completada la transacción, se deberá imprimir un mensaje indicando que la compra ha sido un éxito. Cabe señalar que esta es la única manera de agregar nuevos gatos a tu ejército.

Por otro lado, si el usuario selecciona uno de los Ítemes y cuenta con el dinero para comprarlo, se deberá comprobar si tiene alguna unidad que pueda aceptar el **Ítem** en su ejército. Si hay algún gato compatible con el objeto, se deberá desplegar el menú de Selección de gato. En caso de que el usuario no cuente con ninguna unidad que pueda aceptar el Ítem, se deberá imprimir un mensaje de error y **no** se cobrará el dinero, concluyendo el intento de compra.

Si se selecciona la opción de **Curar Ejército** y se cuenta con el dinero suficiente, cada miembro del equipo deberá restaurar su vida en `CURAR_VIDA` puntos, se deberá descontar el dinero correspondiente. Finalmente, se deberá notificar al usuario que la compra se ha realizado con éxito.

Finalmente, en el caso de que el usuario intente realizar cualquier compra con un precio mayor a su dinero disponible, se deberá mostrar un mensaje notificando el fallo de la compra, junto con la razón de este.

3.2.1. Selección de gato

En este menú se deberá desplegar un encabezado que diga **Selecciona un gato**, seguido de la lista de **Gatos Combatientes** compatibles que se encuentran en el Ejército. Al seleccionar un **Gato** que pueda adquirir el **Ítem**, éste lo obtendrá, se descontará el dinero correspondiente y se imprimirá un mensaje indicando que la compra ha sido un éxito, para luego regresar al Menú de Tienda.

```
*** Selecciona un gato ***

        Clase        Nombre
[1] Gato Guerrero   Cataconi
[2] Gato Guerrero   Aaossa
[3] Gato Mago       Gecoli

Indique su opción:
```

Figura 4: Ejemplo de Menú de Selección de Gato para el Ítem Armadura

4. Entidades

En esta sección se detallarán las entidades que necesitarás para simular *DCCombatientes*. Deberás utilizar conceptos claves como *herencia, clases abstractas, polimorfismo* y *properties*, pertenecientes a **Programación Orientada a Objetos**, para modelar las entidades que se encuentran a continuación. Ten en cuenta que cada uno de estos elementos debe ser incluido en la tarea al menos una vez, por lo que deberás descubrir dónde implementarlos **correctamente** según lo propuesto en el enunciado.

Las tres entidades principales de *DCCombatientes* son **Ejercito**, **Combatiente** e **Ítem**. Debes incluir como mínimo las características nombradas a continuación, pero siéntete libre de añadir nuevos atributos y métodos si lo estimas necesario.

4.1. Ejército

El Ejército es tu centro de operaciones en *DCCombatientes*. Aquí es donde puedes comandar a tus Combatiente y manejar las diferentes acciones a realizar dentro del campo de batalla. Estos interactúan con los **Ejércitos enemigos** por medio de Combates. Para lograr tu objetivo, el ejército debe sobrevivir contra las distintas oleadas de enemigos, por lo que la estrategia es clave para alcanzar la victoria. A continuación, se presentan las características de un **Ejército**:

- **Combatientes**: Contiene todos los combatientes que posees el ejército.
- **Oro**: Contiene el total en `int` de oro disponible a usar en la Menú de Tienda.

Además, debe poder realizar las siguientes acciones:

- **Combatir**: Recibe un Ejército enemigo, para luego comenzar y monitorear el combate, en base a lo descrito en Combates. Debe ser este método el que maneje los enfrentamientos entre combatientes.
- **Presentarse**: Imprime un mensaje que muestra el estado actual de cada combatiente del ejército. Para esto se **debe** sobreescribir el método `__str__` de la clase y se debe hacer uso de las presentaciones de los peleadores. El siguiente es un ejemplo de presentación:

```
*** Este es tu Ejército Actual ***

¡Hola! Soy Gecoli, un Gato Mago con 15 / 40 de vida, 20 de ataque y 5 de defensa.
¡Hola! Soy Mpiavf, un Caballero Arcano con 80 / 80 de vida, 60 de ataque y 15 de defensa.
¡Hola! Soy NatC18, un Mago de Batalla con 60 / 60 de vida, 80 de ataque y 10 de defensa.
¡Hola! Soy CrisTop, un Gato Paladín con 100 / 100 de vida, 40 de ataque y 20 de defensa.

Te quedan 4 combatientes. ¡Éxito, Guerrero!
```

Figura 5: Ejemplo de presentación de un Ejército

Por último, el Ejército debe ser capaz de imprimirse de tal manera que presente a cada uno de los combatientes que posee, junto a todos sus atributos, así como también debe ser capaz de añadir Combatiente a tus unidades disponibles.

4.2. Combatiente

Los combatientes son la unidad principal en *DCCombatientes* y los que ~~sacrifican su vida~~ combaten contra el temible ejército de **Gatochico**.

Los combatientes poseen diversas estadísticas y habilidades en función de su clase, que dictaminará cómo se desenvolverán en el campo de batalla. A continuación, se presentarán las diferentes características de los combatientes:

- **Nombre**: Corresponde a un `str` que representa al nombre del peleador.
- **Vida Máxima**: Corresponde a un `int` entre `0` y `100` que representa cantidad máxima de puntos de vida que puede tener el peleador.
- **Vida**: Corresponde a un `int` que representa la cantidad de vida actual del peleador. No puede tener un valor menor a `0` o mayor a la vida máxima del combatiente.
- **Poder**: Corresponde a un `int` entre `1` y `10` que representa la cantidad de poder del peleador al atacar.
- **Defensa**: Corresponde a un `int` entre `1` y `20` que representa la resistencia del peleador al ser atacado.
- **Agilidad**: Corresponde a un `int` entre `1` y `10` que representa la destreza de un peleador.
- **Resistencia**: Corresponde a un `int` entre `1` y `10` que representa la resistencia de un peleador a lo largo del combate.
- **Ataque**: Corresponde a un `int` que representa el daño que puede producir el combatiente. El ataque de un combatiente dependerá de los *stats* del peleador, junto con su condición actual y se calculará como:

$$ataque = \texttt{round}\left((poder + agilidad + resistencia) \times \frac{2 \times vida}{vida\_máxima}\right)$$

Además, los peleadores pueden realizar las siguientes acciones:

- **Atacar**: Permite atacar a otro peleador. El daño a producir **siempre será un** `int` que dependerá del tipo de peleador, el valor de su **Ataque** y la **Defensa** de su enemigo. Se detalla el ataque de cada tipo de combatiente en Tipos de combatientes.
- **Curarse**: Permite curar el daño recibido por un peleador. Debe recibir un `int` y sumarlo a la vida actual del combatiente, manteniendo su valor dentro de los límites permitidos.
- **Evolucionar**: Permite que un **Gato Combatiente** evolucione dependiendo de su clase y el Ítemes que se utilice en ellos. Los detalles de la evolución se encuentran en Tipos de combatientes.
- **Presentarse**: Imprime en consola un *string* que presente el nombre del combatiente, su tipo, vida máxima, vida actual, ataque y defensa. Este método **debe implementarse** sobreescribiendo el método `__str__` de la clase. El siguiente es un ejemplo de presentación:

```
¡Hola! Soy Gecoli, un Gato Mago con 15 / 60 de vida, 65 de ataque y 5 de defensa.
```

4.2.1. Tipos de combatientes

Existen 6 diferentes tipos de Combatientes, cada uno con características especiales y peculiares formas de combatir. Al momento de **Atacar**, el combatiente podrá activar sus poderes, cambiando el daño que realiza e incluso afectando los *stats* de su enemigo. Luego de cada ataque, el combatiente se cansará lentamente, afectando algunos de sus *stats* permanentemente. Sin importar el tipo de ataque que se realice, **el daño final debe ser un** `int` **mayor o igual a 1**. Se describen las características de cada combatiente a continuación:

- **Guerrero**: Experto en usar la fuerza bruta, no utiliza ningún poder especial y siempre ataca con la totalidad de su fuerza. Debido a la energía gastada en el combate cuerpo a cuerpo, la **Agilidad** del **Guerrero** disminuye en un `CANSANCIO` % después de cada ataque. Cuando el **Guerrero** ataque, la vida de su oponente se reducirá en:

$$\texttt{round}\,(ataque - defensa\_oponente)$$

- **Caballero**: Gracias a su excelente **Agilidad**, al momento de **Atacar** los caballeros tienen un `PROB_CAB` % de probabilidad de sorprender al rival, reduciendo el **Poder** de su enemigo en un `RED_CAB` %, para luego atacar con un `ATQ_CAB` % su ataque. Si el **Caballero** activa su poder, la vida de su oponente se reducirá en:

$$\texttt{round}\left(ataque \times \frac{\texttt{ATQ\_CAB}}{100} - defensa\_oponente\right)$$

  Si no se activa su poder, su daño se calculará igual que el de un **Guerrero**. **Independiente de la activación de sus poderes**, luego de cada ataque, la **Resistencia** del caballero disminuye en un `CANSANCIO` %.

- **Mago**: Debido a sus grandes conocimientos de lo arcano, los magos tienen la habilidad de reducir la **Defensa** del enemigo con sus hechizos. Al momento de **Atacar**, estos tienen un `PROB_MAG` % de probabilidad de activar su poder, ignorando un `RED_MAG` % de la **Defensa** del enemigo, para luego hacer un daño igual al `ATQ_MAG` % de su ataque, calculado como:

$$\texttt{round}\left(ataque \times \frac{\texttt{ATQ\_MAG}}{100} - defensa\_oponente \times \frac{100 - \texttt{RED\_MAG}}{100}\right)$$

Si no se activa su poder, su daño se calculará igual que el de un **Guerrero**. **En cualquiera de los casos**, después de cada ataque el **Mago** se fatiga, disminuyendo su **Resistencia** y su **Agilidad** en un `CANSANCIO` % cada una.

- **Paladín**: Los paladines son seres de luz que se rigen por el valor de la lucha. Es la combinación entre un **Guerrero** y un **Caballero**. Debido a su vasta experiencia en el combate, tienen un `PROB_PAL` % de probabilidad de activar sus poderes de **Caballero**. Además, luego de cada ataque aumentará su **Resistencia** en un `AUM_PAL` %.
- **Mago de Batalla**: Son **Guerreros** con características de **Mago** que utilizan sus conocimientos arcanos para encantar su armamento. Activarán sus poderes de **Mago** con `PROB_MDB` % de probabilidad. Luego de cada ataque, su **Agilidad** disminuye en un `CANSANCIO` %, pero su defensa aumenta en un `DEF_MDB` %.
- **Caballero Arcano**: **Caballero** y **Mago** con gran poder y agilidad. Esta unidad puede alternar entre distintos estilos de combate, por lo que antes de atacar tiene `PROB_CAR` % de probabilidades de optar por **Atacar** cuerpo a cuerpo como un **Caballero** y `100-PROB_CAR` % de probabilidad **Atacar** por medio de hechizos como **Mago**. Luego de cada ataque, el **Caballero Arcano** aumenta su **Poder** y **Agilidad** en un `AUM_CAR` %, y disminuye su **Resistencia** en un `CANSANCIO` %.

4.3. Ítemes

Los ítemes en *DCCombatientes* juegan un papel fundamental para mejorar tu Ejército y asegurar la victoria. Estos podrán ser adquiridos por medio de la Menú de Tienda a cambio de oro, y sirven para poder hacer evolucionar a tus **Gatos Combatientes**, mejorando sus estadísticas. A continuación, te presentamos el armamento:

- **Pergamino**: Pergaminos sagrados creados por los mejores Monjes Cat-tólicos. Permite evolucionar a un **Guerrero** o **Caballero** en un **Mago de Batalla** o **Caballero Arcano**, respectivamente. No se puede usar en combatientes del tipo **Mago**.
- **Lanza**: Lanza empuñada por el legendario Carlo Mewgno. Permite evolucionar a un **Mago** o **Caballero** en un **Mago de Batalla** o **Paladín**, respectivamente. No se puede usar en combatientes del tipo **Guerrero**.
- **Armadura**: Armadura hecha de ~~caparazones de tortugas llamadas Pepa~~ *DCCobalto*. Permite evolucionar a un **Mago** o **Guerrero** en un **Caballero Arcano** o **Paladín**, respectivamente. No se puede usar en combatientes del tipo **Caballero**.

5. Combates

Un combate entre dos ejércitos consiste en tres rondas, donde cada ronda está compuesta por varios enfrentamientos entre dos gatos, uno de cada ejército. Entre las rondas del combate, el jugador podrá hacer cambios y mejoras a su ejército. El ejército del jugador corresponderá al que haya armado desde el Menú de Inicio antes de que inicie el combate, mientras que el ejército de **Gatochico** deberá cargarse desde los Archivos dependiendo de la dificultad.

En cada ronda, el primer gato de cada ejército se enfrenta contra el otro, un **Ataque** a la vez según lo indicado en Enfrentamientos. Los combatientes continuarán luchando hasta que uno salga victorioso o ambos sean derrotados en el intento. Aquellos **Gatos Combatientes** que pierdan el combate morirán, por lo que no podrán seguir luchando y tampoco pueden ser recuperados.

Después de que termine el enfrentamiento, si uno de los gatos sobrevive, este continuará luchando contra el gato siguiente del **Ejército** rival, manteniendo sus *stats* actuales, es decir, sin recuperarse. Esta secuencia

se repite hasta que uno de los equipos se quede sin **Gatos Combatientes**. Si consigues ganar, avanzarás a la siguiente ronda, obteniendo `ORO_GANADO` y volviendo al Menú de Inicio para invertir este dinero en tu **Ejército**. En el caso contrario, perderás el juego y deberás volver al Menú de Inicio regresando a la ronda 1, con un **Ejército** vacío y una cantidad de dinero igual a `ORO_INICIAL`.

5.1. Enfrentamientos

Los enfrentamientos ocurren en un **1 vs 1** entre gatos de equipos enemigos. En ellos, ambos gatos usarán su método de **Ataque** para calcular el daño realizado al rival. Una vez que ambos **Gatos Combatientes** ataquen, se debe disminuir la vida los combatientes **al mismo tiempo**, es decir, se debe aplicar el total del daño a ambos gatos, sin importar si fueron derrotados o no por el ataque actual. Si es que ambos sobreviven, se debe repetir el proceso anterior hasta que alguno de los gatos –o ambos– sea derrotado. El enfrentamiento termina cuando al menos uno de los **Gatos Combatientes** es derrotado en la lucha.

5.2. Ilustración combate

Figura 6: Ejemplo de un combate completo.

En la figura 6 se puede ver un enfrentamiento entre dos ejércitos. De izquierda a derecha tenemos al equipo azul formado por: Gato Caballero, Gato Mago, Gato Guerrero, Gato Mago y al equipo verde, formado por: Gato Mago, Gato Guerrero, Gato Caballero, Gato Guerrero.

En el primer enfrentamiento, el **Gato Mago** verde elimina al primer **Gato Mago** azul, por lo que debe continuar peleando en el segundo enfrentamiento. Luego, el **Gato Mago** verde logra eliminar al **Gato Caballero** azul, sin embargo es eliminado al mismo tiempo; por lo anterior, para el tercer enfrentamiento se seleccionan los siguientes **Gatos Combatientes** de cada equipo. En el tercer enfrentamiento gana nuevamente el **Gato Caballero** del equipo verde, mientras que en el cuarto prevalece el último combatiente del equipo azul, un **Gato Guerrero** que elimina al **Gato Caballero** verde. En la quinta, y última ronda, el **Gato Guerrero** verde logra derrotar al último adversario del equipo azul, dándole la victoria a su equipo.

En caso de que el usuario fuera el equipo azul, se debe retornar al inicio y reiniciará el juego. Al contrario, si el ejército del jugador era el verde, al volver al Menú de Inicio se pasaría a la siguiente ronda con dos gatos en su ejercito, un **Gato Caballero** intacto y un **Gato Guerrero** que recibió daño peleando.

6. Archivos

Para el desarrollo de *DCCombatientes* será necesaria la lectura y carga de archivos. Estos contendrán información sumamente relevante para la ejecución correcta de tu programa.

6.1. Dificultad

En *DCCombatientes* vas a combatir en escenarios pre-establecidos, con 3 posibles niveles de dificultad, fácil, intermedio y difícil. Cada una de las dificultades tendrá un archivo de texto asociado ubicados en la carpeta `data/`, `facil.txt`, `intermedio.txt` y `dificil.txt`. Al momento de ejecutar el programa se deberá recibir el nivel de dificultad por consola. Por ejemplo, al ejecutar "`python3 main.py intermedio`" se está solicitando iniciar el juego con dificultad intermedia.

Puedes asumir que los 3 archivos de dificultad siempre estarán en la carpeta `data/`, pero **no puedes** asumir que la cantidad o los valores de los parámetros entregados por consola serán correctos. En otras palabras, deberás verificar que se esté recibiendo solamente un parámetro y que corresponda a una de las dificultades válidas, ejecutando el programa si es el caso e imprimiendo un mensaje de error si no lo es.

Cada uno de estos archivos contiene los ejércitos a los que te enfrentarías según la dificultad escogida. El archivo tendrá 3 líneas, una por cada ronda, donde se codificarán las características de los combatientes de la ronda, separados por el caracter '`;`'. Un gato tendrá sus características separadas por el caracter '`,`' con la siguiente codficación y orden: `nombre`, `str`; `tipo`, ya sea *Mago, Caballero, Guerrero, Caballero Arcano, Paladín* o *Mago de Batalla*, representados por los `str` `'MAG'`, `'CAB'`, `'GUE'`, `'CAR'`, `'PAL'`, `'MDB'` respectivamente; vida máxima, `int`; defensa, `int`; poder, `int`; agilidad, `int`; resistencia, `int`.

Por ejemplo, en la figura 7, el primer combatiente de la segunda ronda será Elezzio, un **Mago** con vida máxima `55`, defensa `9`, poder `10`, agilidad `6` y resistencia `5`.

```
1 Esteban,CAB,60,10,5,8,5;Elezzio,MAG,55,9,10,6,5;Gery,GUE,70,15,7,5,7
2 Elezzio,MAG,55,9,10,6,5;Esteban,CAB,60,10,5,8,5;Gery,GUE,70,15,7,5,7
3 Gery,GUE,70,15,7,5,7;Esteban,CAB,60,10,5,8,5;Elezzio,MAG,55,9,10,6,5
```

Figura 7: Ejemplo de archivo `facil.txt`

NO puedes asumir que el archivo entregado estará correcto, por lo que, al momento de cargar el archivo, deberás asegurarte de que tenga el formato correcto y de que los *stats* y tipos de los gatos sean válidos. En caso de que el formato del archivo o los valores para alguno de los gatos sean inválidos, se deberá informar al usuario del error, terminando la ejecución del programa.

6.2. Unidades Disponibles

Al momento de comprar combatientes, el jugador deberá elegir el tipo de unidad que quiere y un gato de este tipo será seleccionado del archivo `unidades.txt`, presente en la carpeta `data/`.

Cada línea de este archivo tendrá la información de un gato combatiente, codificada con el mismo formato usado en Dificultad. **NO** puedes asumir que el archivo tendrá un gato de cada tipo básico o que el archivo no contendrá gatos de los tipos que no pueden ser comprados, por lo que deberás asegurarte de que el archivo es correcto al momento de cargarlo, imprimiendo un mensaje de error y terminando el programa si no es el caso.

A continuación se encuentran ejemplos de archivos válidos e inválidos. Notar que el archivo inválido contiene gatos que no pueden ser comprados y además cada uno de los gatos tiene *stats* que se encuentran fuera de los límites permitidos.

```
Esteban,CAB,60,10,5,8,5
Elezzio,MAG,55,9,10,6,5
Gery,GUE,70,15,7,5,7
```

(a) Archivo válido

```
Esteban,CAB,260,10,5,8,5
Elezzio,MAG,55,59,10,6,5
Gery,PAL,70,15,7,5,-6
```

(b) Archivo inválido.

Figura 8: Ejemplo de distintos archivos `unidades.txt`

6.3. `parametros.py`

Para esta tarea, se requiere la creación de un archivo `parametros.py` donde deberás **completar todos los parámetros mencionados a lo largo del enunciado**. Dichos parámetros se presentarán en `ESTE_FORMATO` y en ese color. Además, es fundamental incluir cualquier valor constante necesario en tu tarea, así como cualquier tipo de *path* utilizado.

Un parámetro siempre tiene que estar nombrado de acuerdo a la función que cumplen, por lo tanto, asegúrate de que sean descriptivos y reconocibles. Un ejemplo de parametrizaciones es la siguiente:

```python
CINCO = 5 # mal parámetro
DINERO_MAXIMO = 3000 # buen parámetro
PROBABILIDAD_TORMENTA = 0.2 # buen parámetro
```

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

```python
PI = 3.14
RADIO_CIRCUNFERENCIA = 3
AREA_CIRCUNFERENCIA = PI * (RADIO_CIRCUNFERENCIA ** 2)
```

Dentro del archivo `parametros.py`, es obligatorio que hagas uso de todos los parámetros almacenados y los importes correctamente. Cualquier información no relacionada con parámetros almacenada en este archivo resultará en una penalización en tu nota. Recuerda que no se permite el ***hard-coding***[2], ya que esta práctica se considera incorrecta y su uso conllevará una reducción en tu calificación.

7. `.gitignore`

Para esta tarea **deberás utilizar un** `.gitignore` para ignorar los archivos indicados, este deberá estar dentro de tu carpeta `Tareas/T2/`.

Los elementos que no debes subir y **debes ignorar mediante el archivo** `.gitignore` para esta tarea son:

- El enunciado.
- La carpeta `data/`

Recuerda **no ignorar archivos vitales de tu tarea como los que tú creas o modificas, o tu tarea no podrá ser revisada**.

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

[2] *Hard-coding* es la práctica de ingresar valores directamente en el código fuente del programa en lugar de parametrizar desde fuentes externas.

8. Importante: Corrección de la tarea

En el **siguiente enlace** se encuentra la distribución de puntajes. En esta señalará con color **amarillo** cada ítem que será evaluado a nivel funcional y de código, es decir, aparte de que funcione, se revisará que el código esté bien confeccionado. Todo aquel que no esté pintado de amarillo será evaluado si y sólo si se puede probar con la ejecución de su tarea.

Importante: Todo ítem corregido por el cuerpo docente será evaluado únicamente de forma ternaria: cumple totalmente el ítem, cumple parcialmente o no cumple con lo mínimo esperado. Finalmente, todos los descuentos serán asignados manualmente por el cuerpo docente respetando lo expuesto en **el documento de bases generales**.

Para terminar, si durante la realización de tu tarea se te presenta algún problema o situación que pueda afectar tu rendimiento, no dudes en contactar al ayudante de Bienestar de tu sección. El correo está en el **siguiente enlace**.

9. 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 3.11.X con X mayor o igual a 7.
- Tu programa debe estar compuesto por uno o más archivos de extensión `.py` que estén correctamente ordenados por carpeta. **No se revisará archivos en otra extensión como `.ipynb`**.
- Toda el código entregado debe estar contenido en la carpeta y rama (*branch*) indicadas al inicio del enunciado. Ante cualquier problema relacionado a esto, es decir, una carpte distinta a `T2` o una rama distinta a `main`, se recomienda preguntar en las ***issues* del foro**.
- 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 **único archivo markdown**, llamado `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. El no incluir este archivo, incluir un readme vacío o el subir más de un archivo `.md`, conllevará un **descuento** en tu nota.
- Esta tarea se debe desarrollar **exclusivamente** con los contenidos liberados al momento de publicar el enunciado. No se permitirá utilizar contenidos que se vean posterior a la publicación de esta evaluación.
- Se encuentra estrictamente prohibido citar código que haya sido publicado **después de la liberación del enunciado**. En otras palabras, solo se permite citar contenido que ya exista previo a la publicación del enunciado. Además, se encuentra estrictamente prohibido el uso de herramientas generadoras de código para el apoyo de la evaluación.
- Cualquier aspecto no especificado queda a tu criterio, siempre que no pase por sobre otro que sí sea especificado por enunciado.

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