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

IIC2233 Programación Avanzada (2021-2)

Tarea 1

Entrega

- **Avance de tarea**
  - **Fecha y hora:** miércoles 29 de septiembre de 2021, 20:00
  - **Lugar:** Repositorio personal de GitHub — Carpeta: `Tareas/T1/`
- **Tarea**
  - **Fecha y hora:** sábado 9 de octubre de 2021, 20:00
  - **Lugar:** Repositorio personal de GitHub — Carpeta: `Tareas/T1/`
- `README.md`
  - **Fecha y hora:** sábado 9 de octubre de 2021, 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. *DCCapitolio*

Después de un levantamiento fallido contra Juan Carlos de la Llera, el decano de la Facultad de Ingeniería, y tiempos apocalípticos donde solo queda en el planeta Tierra la Comunidad UC, el decano divide el territorio de ingeniería en 12 distritos. El territorio se comprende del *DCCampus*, que es la central del gobierno del decano, y los doce distritos que están bajo su control. Como castigo para evitar otros futuros levantamientos, se creó un evento llamado *DCCapitolio*, en donde anualmente los doce distritos deben enviar a un tributos (dos personas elegidas al azar) para que luchen a muerte en una arena hasta que solamente quede uno.

Figura 1: Logo de *DCCapitolio*

2. Flujo del programa

Tu misión es crear un programa que permita simular el *DCCapitolio*, una competición realizada en el *DCCampus* en la que cada uno de los 12 participantes (escogidos al azar desde los distritos), llamados tributos, pelearán a muerte en una arena televisada hasta que solo quede uno vivo. El campeonato comenzará con la elección de un tributo por parte del usuario, seguido de una simulación durante varias horas en el cual el usuario deberá tomar decisiones que impactarán en su rendimiento dentro de la arena.

La ejecución e interacción del juego será exclusivamente mediante los Menús impresos en la consola. El objetivo del usuario es ganar *DCCapitolio*, ya sea creando una excelente estrategia de supervivencia, o bien, derrotando a toda persona que se interponga en tu camino. La arena consta de diferentes ambientes, los cuales cambiarán en cada Simulación de hora. Cada ambiente posee sus propias características y eventos. Durante cada Simulación de hora, el jugador podrá tomar decisiones que influirán a lo largo de la partida. Entre ellas, el jugador podrá atacar a otros tributos, realizar acción heroica para impresionar a los patrocinadores, pedir y utilizar objetos, o simplemente rendirse. Cada tributo posee diferentes características, las cuales influirán en el desempeño a lo largo del torneo.

Al iniciar el programa se abrirá el Menú de Inicio, donde el usuario puede empezar una nueva partida o salir. En caso de empezar una partida, se deberá seleccionar un tributo que se usará en *DCCapitolio*, los 11 tributos restantes se convertirán en oponentes. Luego, el programa debe mostrar el Menú Principal, desde el cual se podrá simular una hora, ver el estado actual del tributo, utilizar objetos, el resumen del

DCCapitolio con el estado de los otros tributos o rendirse.

Cada Simulación de hora comenzará con una acción por parte del tributo escogido. Luego, se realizarán encuentros aleatorios entre los tributos, los cuales pueden incluso acabar con la vida de tus oponentes. Finalmente, ocurrirá un evento especial que dependerá de la arena seleccionada y afectará a todos los tributos, los cuales deberán recuperarse para seguir sobreviviendo en la arena.

Esta simulación continuará hasta que tu tributo resulte muerto, se rinda o gane el *DCCapitolio*. Una vez finalizada la simulación, el usuario deberá volver al Menú de Inicio.

3. Menús

Para esta tarea, la simulación del *DCCapitolio* 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 ejecutar el programa se deberá desplegar un Menú de inicio, en el cual se den las opciones de **iniciar partida** y **salir del programa**. Dado que es el primer menú que se muestra en pantalla, no es necesario implementar la opción de **volver atrás**.

Si se selecciona **iniciar partida**, se deberán mostrar todos los tributos de `tributos.csv` junto con sus distritos, y dar la opción de escoger uno para utilizar en el *DCCapitolio*. Por otro lado, los demás tributos serán parte también del *DCCapitolio*, con los cuales deberás enfrentarte durante este. Luego, se deberá elegir una de las arenas disponibles en `arenas.csv`, considerando los respectivos datos de la opción escogida.

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

```
*** Menú de Inicio ***
----------------------
[1] Iniciar partida
[2] Salir
```

3.2. Menú Principal

En este menú se encontrará todas las opciones para poder poder salir victorioso del *DCCapitolio*. Este constará de cinco opciones principales: **Simulación de hora**, **Mostrar estado del tributo**, **Utilizar objeto** y **Resumen *DCCapitolio***.

```
*** Menú Principal ***
----------------------
[1] Simulación hora
[2] Mostrar estado del tributo
[3] Utilizar objeto
[4] Resumen DCCapitolio
[5] Volver
[6] Salir
```

3.2.1. Simulación de hora

Al seleccionar esta opción el programa deberá simular una hora transcurrida en el *DCCapitolio*, en el ambiente correspondiente a la hora.

En primer lugar, se podrá elegir entre cuatro opciones, **Acción heroica**, **Atacar a un tributo**, **Pedir objeto a patrocinadores** o **Hacerse bolita** [1].

1. **Acción heroica**: al elegir esta opción, el tributo gastará energía igual a `ENERGIA_ACCION_HEROICA` para ganar una cantidad de popularidad igual a `POPULARIDAD_ACCION_HEROICA`, la cual se explica en la sección de Entidades. Se deberá notificar en consola la cantidad de popularidad que ganó y la cantidad de energía restante. Solo se podrá realizar esta acción si posee la energía necesaria.

2. **Atacar a un tributo**: se mostrarán todos los tributos con vida y se deberá elegir uno de ellos al cual se atacará (esta acción se describe en Entidades). El tributo gastará una cantidad de energía igual a `ENERGIA_ATACAR` para poder atacar a otro tributo, quitándole vida a este. En caso de que el tributo atacado muera, el tributo ganará una cantidad de popularidad igual a `POPULARIDAD_ATACAR`. Solo se podrá realizar esta acción si posee la energía necesaria. Se deberá notificar en consola el resultado de atacar a un tributo y la cantidad de energía restante.

3. **Pedir objeto a patrocinadores**: esta opción permitirá obtener un objeto aleatorio de `objetos.csv` a cambio de una cantidad de popularidad igual a `POPULARIDAD_PEDIR` establecida para esta opción. Se deberá notificar en consola el resultado de pedir un objeto.

4. **Hacerse bolita**: permitirá al tributo recuperar una cantidad de energía igual a `ENERGIA_BOLITA`. No habrá restricciones para elegir esta opción.

En el caso de que no se pueda llevar a cabo la opción elegida, se deberán volver a mostrar la opciones y dar de nuevo la posibilidad de escoger una de estas cuatro.

Una vez realizada la opción escogida, se deberá simular una cantidad de **encuentros** entre todos los tributos dependiendo de la dificultad de la arena y la cantidad de tributos con vida. En cada encuentro, un **tributo enemigo aleatorio** atacará a **otro tributo aleatorio**. Este último es elegido de entre todos los tributos con vida (incluido el tributo utilizado) y distintos al tributo que está atacando. Por lo tanto, en estos encuentros solo **atacaran** tributos **enemigos**, y los atacados serán todos los tributos que se encuentren vivos, incluyendo el tributo del usuario. El tributo enemigo que ataca, no gastara energía ni ganara popularidad al momento de realizar esta acción, sólo producirá un daño en el tributo atacado. Los tributos enemigos, **sólo podrán realizar la acción de atacar**. Cabe mencionar que puede darse el caso de que un tributo sea atacado mas de una vez. Todos los encuentros realizados, deberán ser mostrados en consola junto al resultado de cada uno junto con la energía restante de cada tributo implicado.

Finalmente, existirá una `PROBABILIDAD_EVENTO` adjunto a la arena seleccionada. El evento que ocurrirá será uno aleatorio dentro de los eventos disponibles del ambiente simulado en la hora, especificado en Ambiente, y se deberá aplicar su efecto en el *DCCapitolio*.

Al final de cada hora, debe ser posible visualizar un resumen de todo lo que ocurrió en esa hora. En este se deberá mostrar como mínimos: el resumen de la opción escogida al principio de la simulación de la hora actual, los encuentros ocurridos junto a su resultado, los tributos derrotados en la hora, los tributos que siguen con vida y el resumen del evento del ambiente (si ocurrió o no). Luego, se deberá volver al menú principal.

En caso de que la vida del tributo escogido llegue a 0, se deberá notificar en consola y volver al Menú de Inicio. En caso de que el tributo escogido sea el único con vida, se notificará en consola la victoria y se volverá al Menú de Inicio.

[1] Hacerse bolita se refiere a protegerse y recuperar energía

3.2.2. Mostrar estado del tributo

Al seleccionar esta opción, se deberá desplegar en consola toda la información relevante del estado del tributo. Esta deberá incluir como mínimo: el **nombre del tributo**, **distrito**, su **edad**, **vida**, **energía**, **agilidad**, **fuerza**, **ingenio**, **popularidad**, los **objetos del tributo** y el **peso**. Puedes incluir más información que consideres relevante.

```
                              Estado tributo
--------------------------------------------------------------------------------
Nombre: DCCatniss Everdeen
Distrito: DCC
Edad: 23
Vida: 51
Energía: 43
Agilidad: 13
Fuerza: 6
Ingenio: 5
Popularidad: 15
Objetos: arco y flecha, handroll, panino fritto wurstel patatine
Peso: 11
```

3.2.3. Utilizar objeto

Se dará la opción de utilizar uno de los objetos que posee el tributo, aplicando su efecto en este, notificando el resultado en consola. En caso de **no** poseer objetos se deberá notificar en consola y volver al Menú Principal.

3.2.4. Resumen *DCCapitolio*

Al seleccionar esta opción, se deberá desplegar en consola toda la información relevante del estado del *DCCapitolio*, mostrando como mínimo: **la dificultad de la Arena**, los **tributos que siguen vivos con la cantidad de vida** y el **ambiente en la próxima hora** Puedes incluir más información que consideres relevante.

```
                            Estado DCCapitolio
--------------------------------------------------------------------------------
Dificultad: avanzado
Tributos vivos:
    Gatochico: 55
    matiasmasjuan: 33
    DCCollao: 60
Próximo ambiente: montaña
```

4. Entidades

En esta sección se detallarán las entidades que necesitarás para simular *DCCapitolio*. Deberás utilizar conceptos claves de **Programación Orientada a Objetos** como herencia, clases abstractas, polimorfismo, *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 *DCCapitolio* son **Tributo**, **Ambiente**, **Objeto** y **Arena**. Debes incluir como mínimo las características nombradas a continuación, pero siéntete libre de añadir nuevos atributos y métodos si lo estimas necesario.

4.1. Tributo

Los tributos son los personajes de *DCCapitolio*. Interactúan en la **Arena** y pueden usar distintos **Objetos** durante la simulación. El objetivo de los tributos es ganar el *DCCapitolio* que se consigue al ser el último participante vivo en la arena. Para lograr su objetivo los tributos deberán sobrevivir a los eventos que se producen al final de cada hora como también a los ataques de los otros tributos.

En *DCCapitolio* existen los patrocinadores, estas personas serán fundamentales para poder seguir con vida en el juego. Pues, los tributos pueden pedir objetos a los patrocinadores por medio de la popularidad. La popularidad en *DCCapitolio* son puntos que pueden ser canjeados por objetos. Para conseguir puntos de popularidad, el tributo deberá atacar a otros tributos o realizar acciones heroicas siempre y cuando tenga la **energía** suficiente para realizar la acción. De lo contrario deberá descansar o consumir algún objeto que le regenere energía. Además, ten en cuenta que cada tributo tendrá a su disposición una mochila que va a empezar vacía por lo que tendrás que ganar mucha popularidad para poder pedir un objeto a los patrocinadores. A continuación, se presentarán las características de los tributos:

- **Nombre**: corresponde a un `str` que representa el nombre del tributo.
- **Distrito**: es un `str` que corresponde al distrito que representa el tributo.
- **Edad**: corresponde a un `int` que representa los años que tiene el tributo.
- **Vida**: corresponde a un `int` entre `0` y `100` que representa la cantidad de vida que tiene el tributo.
- **Energía**: corresponde a un `int` entre `0` y `100` que representa la cantidad de energía del tributo.
- **Esta Vivo**: es un `bool` que indica si el tributo se encuentra vivo.
- **Agilidad**: corresponde a un `int` que representa la destreza que tiene el tributo.
- **Fuerza**: corresponde a un `int` que representa la vigorosidad de un tributo.
- **Ingenio**: corresponde a un `int` que representa la inteligencia del tributo.
- **Popularidad**: corresponde a un `int` que representa los puntos de popularidad del tributo. Estos puntos se pueden canjear para obtener objetos.
- **Mochila**: almacena los objetos que posee el tributo.
- **Peso**: corresponde a un `int` que representa la suma de los pesos de los objetos que está llevando el tributo. La cantidad de objetos que puede llevar el tributo no tiene limite pero impacta en el desempeño durante la batalla.

Además, los tributos pueden realizar las siguientes acciones:

- **Atacar**: Este método permite atacar a otro tributo. El daño producido por el tributo se calcula según la siguiente formula:

$$min\left(90, max\left(5, \frac{60 * Fuerza + 40 * Agilidad + 40 * Ingenio - 30 * Peso}{Edad}\right)\right)$$

  En donde:

  - *Agilidad*: Es el atributo **Agilidad** del Tributo. Se encuentra en `tributos.csv`.

- *Ingenio*: Es el atributo **Ingenio** del Tributo. Se encuentra en `tributos.csv`.
- *Edad*: Es el atributo **Edad** del Tributo. Se encuentra en `tributos.csv`.
- *Fuerza*: Es el atributo **Fuerza** del Tributo. Se encuentra en `tributos.csv`.
- *Peso*: Representa la suma de los pesos de todos los objetos que está llevando el tributo. Hace referencia a la característica **Peso**.

- **Utilizar Objeto**: Está acción permite utilizar objetos que tiene el tributo en la mochila. Los objetos pueden ser de tres tipos: **Consumible**, **Arma** y **Especial**. Los beneficios de estos objetos los puedes ver en la sección Tipos de Objeto. Por lo que, esta acción se ejecuta sólo si posee algún objeto en la mochila y una vez, utilizado el objeto se debe sacar de la mochila.
- **Pedir Objeto**: Esta acción permite a los tributos pedirle a los patrocinadores objetos. Para pedir un objeto, el tributo tiene que canjear los puntos de popularidad para poder obtener un objeto. El objeto recibido tiene que ser aleatorio de la lista posibles de objetos lo que significa que puede ser tanto un **Arma**, **Consumible** o **Especial**. Además, va a tener un costo de `COSTO_OBJETO`, que debe ser descontado de la popularidad del tributo.
- **Acción Heroica**: Está acción permite ganar puntos de popularidad que puedes canjear para obtener objetos de parte de los patrocinadores. La cantidad de puntos que puedes ganar es igual a `POPULARIDAD_ACCION_HEROICA`. Además, el tributo al realizar está acción gastará una cantidad de energía igual a `ENERGIA_ACCION_HEROICA`.

4.2. Ambiente

Un ambiente define las características de la Arena como también los eventos que ocurrirán en *DCCapitolio*. Existen tres ambientes, los cuales van a ir cambiando en cada hora en la Arena, los tres tipos de ambientes son **Playa**, **Montaña** y **Bosque**. Además, en cada hora de la simulación, los ambientes tienen que estar rotando. La rotación de los ambientes tiene que ser cíclico y tener un orden. El orden en que van a rotar los ambientes queda a criterio del estudiante, lo importante es que una vez definido el orden este se tiene que respetar en todo el programa. Por ejemplo, si el orden elegido es **Montaña** - **Playa** - **Bosque**, al inicio de la simulación la Arena tiene que tener el ambiente **Montaña**. Después, para la siguiente hora simulada, el ambiente rota a **Playa** y así, sucesivamente. A continuación, se presentan las características de un Ambiente.

- **Nombre**: corresponde a un `str` que representa el nombre de un ambiente.
- **Evento**: es un conjunto de todos los eventos que pueden ocurrir en el Ambiente. Estos eventos dependen del tipo de Ambiente, que puede ser **Playa**, **Montaña** y **Bosque**. Lo pueden encontrar en `ambientes.csv`.

Además, deberá realizar la siguiente acción:

- **Calcular Daño**: En cada hora, se producirá un daño que tiene que ser calculado por medio de una fórmula. Esta fórmula la puedes encontrar en Tipos de Ambiente.

4.2.1. Tipos de Ambiente

Además, existen distintos tipos de ambiente, cada uno con diferentes parámetros y eventos especiales. Estos son playa, montaña y bosque.

- **Playa**: Este tipo de ambiente se caracteriza por su clima cálido y vientos fuertes. Estas características están definidos en `VELOCIDAD_VIENTOS_PLAYA` y `HUMEDAD_PLAYA`. Además, los eventos asociados a este tipo de ambiente son: Tsunami, Huracán y Marea Roja.

- **Montaña**: El ambiente montaña, se caracteriza por su clima frío. Estas características están definidas en `NUBOSIDAD_MONTANA`. y `PRECIPITACIONES_MONTANA`. Además, posee los siguientes eventos: Avalancha, Tormenta de Nieve y Erupción Volcánica.
- **Bosque**: Este ambiente se caracteriza por su clima templado. Estas características están definidos en `PRECIPITACIONES_BOSQUE` y `VELOCIDAD_VIENTOS_BOSQUE`. Además, posee los siguientes eventos: Incendio, Tormenta Eléctrica y Gas Venenoso.

A partir de lo anterior, los eventos pueden afectar a los tributos de distinta manera en cada hora simulada en la Arena. Para ello, se define la siguiente fórmula para calcular el daño provocado por el evento. Tener en cuenta que el daño se le tiene que restar a todos los tributos en cada hora simulada en *DCCapitolio*.

$$max\left(5, \frac{0.4 * humedad + 0.2 * vientos + 0.1 * precipitaciones + 0.3 * nubosidad + \ daño\_evento}{5}\right)$$

En donde:

- *daño_evento*: corresponde al daño específico de cada evento, que está detallado en el archivo `ambientes.csv`
- *humedad*: corresponde a `HUMEDAD_PLAYA`.
- *vientos*: corresponde a `VELOCIDAD_VIENTOS_PLAYA` o `VELOCIDAD_VIENTOS_BOSQUE`.
- *precipitaciones*: corresponde a `PRECIPITACIONES_BOSQUE` o `PRECIPITACIONES_MONTANA`.
- *nubosidad*: corresponde a `NUBOSIDAD_MONTANA`.

Se debe tener en cuenta que en caso de que el ambiente no tenga alguna característica presente en la fórmula, se tiene que considerar estas características igual a 0. Por ejemplo, si el ambiente de la arena es **Playa**, la fórmula para calcular sería la siguiente:

$$max\left(5, \frac{0.4 * \texttt{HUMEDAD\_PLAYA} + 0.2 * \texttt{VELOCIDAD\_VIENTOS\_PLAYA} + 0.1 * 0 + 0.3 * 0 + \ daño\_evento}{5}\right)$$

4.3. Objeto

Los objetos de *DCCapitolio* juegan un papel fundamental en la supervivencia de cada tributo. Estos podrán ser adquiridos mediante el menú principal, donde el usuario podrá pedir obsequios a los patrocinadores de *DCCapitolio*. Los objetos pueden ser del tipo arma, consumible o especial y los atributos de entidad son:

- **Nombre**: corresponde a un `str` que representa el nombre de un objeto.
- **Tipo**: corresponde a un `str` que representa el tipo del objeto, que puede ser **Consumible**, **Arma** o **Especial**.
- **Peso**: corresponde a un `int` que representa la capacidad del objeto. Se espera que el peso del objeto afecte al peso del tributo.

Además deberá realizar la siguiente acción:

- **Entregar beneficio**: Esta acción recibe la entidad tributo a la cual va a aplicar los beneficios según los parámetros y formulas del tipo de objeto, estos son arma, comida o un objeto especial.

4.3.1. Tipos de Objeto

- **Consumible**: Es un objeto que permite aumentar la energía del tributo. La cantidad de energía es igual a `AUMENTAR_ENERGIA`.
- **Arma**: Es un tipo de objeto que permite aumentar la fuerza del tributo. El aumento está definido a partir de la siguiente ecuación:

$$Fuerza * (\texttt{PONDERADOR\_AUMENTAR\_FUERZA} * Riesgo + 1)$$

  En donde:

  - *Riesgo*: Es una característica de la Arena, que puede tomar valores entre `0` y `1`, considerando `0` como riesgo mínimo y `1` como riesgo máximo. Lo puedes encontrar en `arenas.csv`.
  - Fuerza: Es el atributo de la entidad Tributo. Lo puedes encontrar en `tributos.csv`.

- **Especial**: En *DCCapitolio* existen objetos especiales que poseen los mismos beneficios que un Objeto Consumible y un Objeto Arma. Además, un objeto especial cuenta con beneficios adicionales dados por `AUMENTAR_AGILIDAD` y `AUMENTAR_INGENIO`.

4.4. Arena

Es el mapa general que podrás elegir al momento de comenzar una partida de *DCCapitolio*. Cada arena tiene sus propias características, las que afectarán tanto la supervivencia de los tributos como también las condiciones del programa. Dentro de la arena van a existir ambientes los cuales van a ir cambiando a medida que pasa cada hora de la simulación, siguiendo un orden cíclico. A continuación, se enunciarán las características de la entidad Arena.

- **Riesgo**: es un `float` entre `0` y `1` que representa el riesgo de la arena.
- **Dificultad**: corresponde a un `str` que puede ser `'principiante'`, `'intermedio'` y `'avanzado'`.
- **Jugador**: tributo elegido por el jugador para participar en la arena.
- **Tributos**: todos los participantes de *DCCapitolio* que el jugador no controla.
- **Ambientes**: conjunto de todos los ambientes disponibles en la arena.

La Arena puede realizar las siguiente acción:

- **Ejecutar Evento**: Cada arena es susceptible a un evento al final de una hora de simulación. El evento que se genere tiene una probabilidad de ocurrencia de `PROBABILIDAD_EVENTO` en cada hora. El evento que podrá ocurrir será uno de los tres eventos posibles para cada ambiente, con la misma probabilidad. El daño que provoca el evento depende del tipo de ambiente en el cual se encuentre la arena en esa hora.
- **Encuentros**: Realiza los encuentros de batalla entre los tributos menos el tributo controlado por el jugador. Dentro de este encuentro un tributo aleatorio deberá atacar a otro tributo aleatorio que se encuentre con vida y distinto de sí mismo. Como el número de encuentros durante una hora puede variar según la dificultad del mapa un tributo puede luchar varias veces en la misma hora. El número de encuentros que tiene que ocurrir durante cada hora se calcula como:

$$N° \text{ de encuentros} = Riesgo * (\text{Tributos restantes}) \ // \ \texttt{2}$$

  donde:

  - **Riesgo**: es un `float` entre `0` y `1` que representa el riesgo de la arena. Lo puedes encontrar en `arenas.csv`

- **Tributos Restantes**: es un `int` que representa la cantidad total de tributos que quedan con vida en la arena.

5. Archivos

Los siguientes archivos contienen la base de datos de los tributos, arenas, ambientes y objetos que 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 (`","`).

5.1. `tributos.csv`

Este archivo contiene todos los tributos que participarán de un juego de *DCCapitolio*, junto con sus características que son descritas en la siguiente tabla:

Nombre	Descripción
Nombre	Indica el nombre del tributo
Distrito	Indica el distrito al que pertenece el tributo
Edad	Indica la edad del tributo
Vida	Indica la vida del tributo.
Energía	Indica la energía del tributo.
Agilidad	Indica la agilidad del tributo.
Fuerza	Indica la fuerza del tributo.
Ingenio	Indica el ingenio del tributo.
Popularidad	Indica la popularidad del tributo.

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

```
nombre,distrito,edad,vida,energía,agilidad,fuerza,ingenio,popularidad
DCCatniss Everdeen,DCC,23,79,77,9,3,5,6
gatochico,Construcción,29,81,64,6,9,3,2
igbasly,Estructural,18,100,35,8,10,9,7
DCCollao,Ambiental,27,63,90,2,10,5,6
```

5.2. `arenas.csv`

Este archivo contiene todas las opciones de arenas que el usuario podrá seleccionar para comenzar una partida de *DCCapitolio*, junto con sus características que son descritas en la siguiente tabla:

Nombre	Descripción
Nombre	Indica el nombre del mapa
Dificultad	Indica la dificultad del mapa y puede ser una de las siguientes opciones: `'principiante'`, `'intermedio'` o `'avanzado'`
Riesgo	Indica del 0 al 1 el riesgo del mapa, siendo 0 el riesgo mínimo y 1 el máximo.

Un ejemplo del archivo arenas.csv es el siguiente:

```
nombre,dificultad,riesgo
Patio de la Virgen,principiante,0.3
Los Meones,avanzado,1
Escaleras CS,intermedio,0.6
```

5.3. `ambientes.csv`

Como se explicitó en la sección Tipos de Ambiente, existen tres tipos de ambiente: Bosque, Playa y Montaña. Cada ambiente posee distintas características que se traducen a distintos tipos de eventos. Debido a las características únicas de cada ambiente, existen distintos tipos de eventos que pueden ocurrir a la hora se simular una hora. El archivo `ambientes.csv` detalla los tres distintos eventos que pueden ocurrir en cada ambiente.

Nombre	Descripción
Nombre	Indica el nombre del ambiente. Corresponde a una de las siguientes opciones: bosque, playa o montaña.
Evento	Indica el nombre de cada evento que podría ocurrir en el ambiente.
Daño	Indica el daño que provoca cada evento.

A continuación se muestra un ejemplo del archivo `ambientes.csv`. Cada línea contiene el nombre del ambiente, seguido por los tres eventos y sus respectivos daños (separados por `";"`):

```
nombre,evento_1;daño_1,evento_2;daño_2,evento_3;daño_3
bosque,incendio;5,tormenta eléctrica;4,gas venenoso;8
playa,tsunami;9,marea roja;3,huracán;10
montaña,avalancha;6,tormenta de nieve;5,erupción volcánica;10
```

5.4. `objetos.csv`

Este archivo contiene la información de todos los objetos que existirán dentro del juego. Cada objeto tiene un nombre, tipo de objeto (consumible, arma o especial), y su peso.

Nombre	Descripción
Nombre	Indica el nombre del objeto.
Tipo	Indica el tipo de objeto. Puede ser consumible o arma.
Peso	Indica el peso del objeto.

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

```
nombre,tipo,peso
arco y flecha,arma,3
puñal,arma,8
catapulta,especial,6
handroll,consumible,3
```

5.5. `parametros.py`

Para esta tarea, deberás crear un archivo `parametros.py` y completarlo con todos los parámetros mencionados a lo largo del enunciados, los cuales encontrarás en `ESTE_FORMATO`. 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
ENERGIA_ACCION_HEROICA = 15 # buen parámetro
POPULARIDAD_ATACAR = 4 # buen parámetro
```

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

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

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***[2], 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**[3].
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 3 décimas. Deberás indicar en tu `README` si implementaste alguno de los bonus, y cuáles fueron implementados.

Acá van los bonus del enunciado. Por lo general cada bonus se anota como una subsección. Por ejemplo:

[2] *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.

[3] Esta nota es sin considerar posibles descuentos.

6.1. Guardar Partida (3 décimas)

Para optar a este bonus se debe implementar la opción de guardar la partida actual en la simulación en un archivo `partidas.txt` dentro de la misma carpeta donde se encuentra la tarea. La opción de **guardar la partida** debe aparecer dentro del Menú Principal y al presionarlo deberá preguntar al usuario el nombre con el cual se quiere guardar la partida y después volver al Menú de Inicio. Finalmente, desde el menú de inicio se debe agregar la opción de **continuar partida** la cual al ser seleccionada debe mostrar todas las partidas guardadas con su respectivo nombre y fecha de creación.

El formato que se debe utilizar para guardar la partida dentro del archivo `partidas.txt` queda a su criterio, sin embargo, se debe asegurar de que toda la información del tributo seleccionado, los adversarios, objetos y datos de la partida se almacenen correctamente de modo tal que se pueda restaurar la partida guardada.

7. Avance de tarea

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

Una versión preliminar que refleje cómo planeas modelar tu programa, que corresponderá al **avance** de esta tarea. 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.

Luego de esto, junto a la entrega final, deberás entregar una **versión final** de tu diagrama que **represente fielmente** la modelación del problema usada en tu programa.

En ambos casos, el diagrama deberá:

- Entregarse en **formato PDF o de imagen**[4].
- 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).
- 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. `.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:

- `tributos.csv`

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

- `arenas.csv`
- `ambientes.csv`
- `objetos.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__`.

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.

9. 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.

10. Importante: Corrección de la tarea

Para esta tarea, el carácter funcional del programa será el pilar de la corrección, es decir, **sólo se corrigen tareas que se puedan ejecutar**. Por lo tanto, se recomienda hacer periódicamente pruebas de ejecución de su tarea y *push* en sus repositorios.

Cuando se publique la distribución de puntajes, se señalará con color **amarillo** cada ítem que será evaluado a nivel de código, todo aquel que no esté pintado de amarillo significa que será evaluado si y sólo si se puede probar con la ejecución de su tarea.

En tu archivo `README.md` deberás señalar el archivo y la línea donde se encuentran definidas las funciones o clases relacionados a esos ítems.

Finalmente, 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 jefe de Bienestar al siguiente correo: **bienestar.iic2233@ing.puc.cl**.

11. 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 3.8.
- 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).