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

IIC2233 Programación Avanzada (2024-2)

Tarea 2

Entrega

- **Tarea y** `README.md`
  - **Fecha y hora oficial (sin atraso):** viernes 13 de septiembre de 2024, 20:00
  - **Fecha y hora máxima (2 días de atraso):** domingo 15 de septiembre de 2024, 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.
  - **Formulario entrega atrasada:** 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. *DCCampesino*

Luego de pasar días trabajando al lado de Hernán en el desarrollo de los cultivos, todo parecía estar saliendo de maravilla. Los predios, que antes estaban en plena expansión, ahora florecían bajo un extremo cuidado. Sin embargo, en una tarde tranquila, se escucha a lo lejos un sonido perturbador. Al mirar por los binoculares, te impactas al ver humo emergiendo del centro de operaciones de **Los Zombies**.

Entras en pánico al darte cuenta que el humo era el presagio de una horda de enojados zombies acercándose con la intención de terminar con todo el jardín. Frente a tal amenaza todo se veía perdido, pero, afortunadamente Hernán estaba ocupado en el laboratorio creando una fórmula secreta que tenía como objetivo evolucionar el cultivo de plantas. Esta fórmula secreta es capaz de transformar las plantas en plantas mutantes capaces de defender y obtener recursos para mejorar tus defensas.

Mientras los zombies se acercan y las plantas se fortalecen, se necesita de tu ayuda para poder mantener las plantas en buenas condiciones. La batalla por la supervivencia ha comenzado, y el destino de los cultivos y tu propia seguridad dependen de ti.

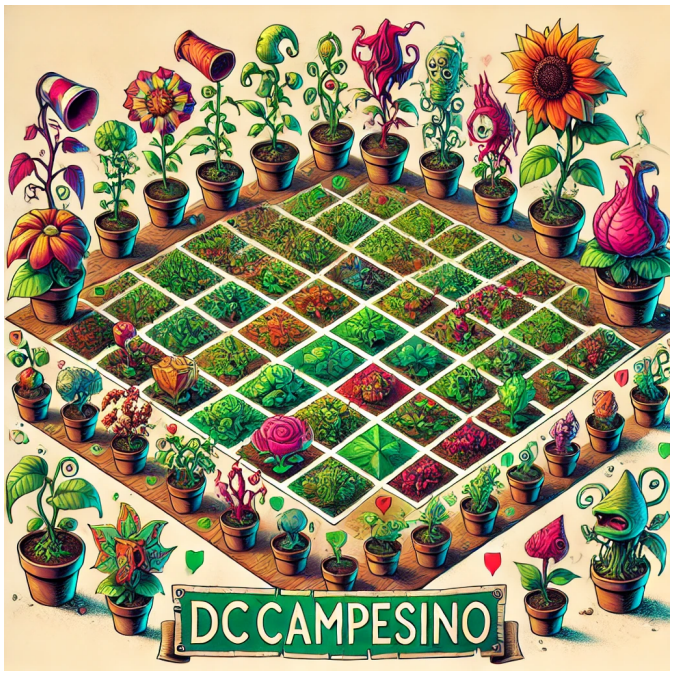

Figura 1: Logo de *DCCampesino*.

2. Flujo del programa

En esta tarea, tu objetivo es convertirte en el *DCCampesino* más eficiente mientras enfrentas los desafíos del clima y a tus archienemigos, **Los Zombies**! Para iniciar tu misión, primero debes seleccionar tu jardín inicial, que se representará como un tablero de dimensiones variables, sobre el cual podrás colocar plantas para producir **soles**. Además, debes especificar la dificultad del juego, la cual debes entregar como un **argumento** en la consola. Es obligatorio seleccionar un jardín y una dificultad.

Una vez dentro, accederás al Menú de Inicio con una cantidad inicial de `SOLES_INICIO`[1], que contarán como tu puntaje total al final de la partida, y con una temperatura inicial de valor `TEMP_INICIAL` °C.

[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`.

Entre cada día tendrás la opción de colocar **plantas** en alguna casilla del Jardín a través de su menú. Es crucial que aproveches las características y sinergias de cada **planta**. Por ejemplo, puedes situar un **Potencilantro** junto a un **Solaterillo** para aumentar la cantidad de soles que produce.

Después de organizar tu Jardín, puedes seleccionar la opción **Pasar día**, que simula el transcurso de un día, lo que te otorga los soles producidos por tus plantas, junto con un valor aleatorio entre `MIN_SOLES` y `MAX_SOLES`. Además, cada vez que pasa un día, pueden ocurrir **Eventos** como **Heladas** u oleadas de **Los Zombies**. Todos estos eventos están detallados en la sección Eventos. Tendrás un plazo de `DURACION` días para generar la mayor cantidad de soles posible.

Al completar el día número `DURACION` deberás imprimir un mensaje notificando que es el final del juego, luego deberás imprimir cada planta presente en el jardín para acabar imprimiendo el estado final de tu jardín. Luego de esto la ejecución deberá detenerse marcando el final de tu aventura como *DCCampesino*.

3. Menús

Dentro de esta increíble labor, tú como *DCCampesino* recibirás las interacciones del usuario que se realizarán a través de menús. Tanto la interacción como el despliegue de estos menús deberán ser mediante la consola.

Para esto, se ejecutará tu programa mediante un archivo `main.py`, el cual deberá aceptar argumentos de línea de comando para poder indicar el nombre del jardín y la dificultad del juego[2], siguiendo el formato "`python3.11 main.py nombre_jardin nombre_dificultad`"[3]. Para trabajar con argumentos de línea de comando, te recomendamos investigar sobre el funcionamiento de `sys.argv` de Python en internet.

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 debe volver a desplegar las opciones del menú.

Además, **todos**[4] los menús deben tener la opción de volver al Menú de Inicio. A continuación se explicarán y dan ejemplo de cada uno de los menús mínimos a implementar. Sin embargo, son libres de escoger la distribución estética siempre y cuando cuenten con todos los requerimientos **mínimos** pedidos.

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 los **soles** disponibles. La explicación de cada opción será señalado detalladamente a lo largo de esta sección. También se debe señalar la temperatura actual del **Jardín** y el número de día actual. Luego de lo anterior, se desplegarán las opciones **Menú Jardín**, **Laboratorio**, **Pasar Día** y **Salir del programa**.

Aspectos a considerar:

- Los soles disponibles al comenzar el programa corresponden a `SOLES_INICIO` y no puedes tener soles negativos.
- La temperatura indicada corresponde a la del jardín.
- El Día actual indica el número de oleada que va a acontecer al seleccionar la opción **Pasar Día**, que es un `int` entre 0 y `DURACION`.

[2] Dificultad puede ser: facil, intermedio o dificil.
[3] Por ejemplo: "`python3.11 main.py dificil`".
[4] Menos el Menú de Inicio.

Opciones del Menú de Inicio:

- Al seleccionar la opción **Menú Jardín**, se deberá desplegar un segundo menú, que se encuentra detallado en Menú Jardín.
- Al seleccionar la opción **Laboratorio**, se deberá desplegar un segundo menú, que se encuentra detallado en Laboratorio.
- Al seleccionar la opción **Pasar Día**, se deberá comenzar con la simulación, que se encuentra detallada en Simular día donde se podrán activar algún evento explicado en Eventos

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. Un posible ejemplo de un mensaje de salida sería:

```
Saliste de DCCampesino, supongo que vas a seguir tu camino...
Gracias por ayudar a Hernán a combatir los zombies!!
```

Figura 2: Ejemplo de mensaje de salida

A continuación, se presentan un ejemplo del menú de inicio:

```
--------------------------
      Menú de inicio
--------------------------

Soles disponibles: 1000
Temperatura: 15 °C
Día actual: 3

[1] Menu Jardín
[2] Laboratorio
[3] Pasar Día

[0] Salir del programa

Opción:
```

Figura 3: Ejemplo de Menú de Inicio

3.2. Menú Jardín

En el menú de Jardín, se mostrará el título que diga **Jardín**, seguido del tablero que representa el Jardín del usuario cargado por consola, y el despliegue de las opciones **Intercambiar**, **Cultivar**, y **Regar** que se explicarán a continuación.

```
--------------------------------
             Jardín
--------------------------------
      ["X","X","X","X","X"]
      ["X","S","X","X","X"]
      ["X","X","X","X","A"]


[1] Intercambiar
[2] Cultivar
[3] Regar

[0] Volver al Menú de inicio

Indique su opción:
```

Figura 4: Ejemplo de Menú Jardín

Opciones del Jardín

- **Intercambiar:** Esta opción permite cambiar la posición entre dos ubicaciones del tablero. Los cambios realizados aquí deben ser tanto visibles en el tablero del **Jardín** como funcionales para los eventos que transcurran en la simulación de un día. Deberás ingresar dos posiciones válidas, separadas por un punto y coma (“**;**”). Cada posición debe estar compuesta por dos números, que representan la fila y la columna respectivamente, separados por una coma (“**,**”).

  Por ejemplo, un input de `3,2;0,4` indicará que lo que esté en la fila 3, columna 2 se intercambiara con lo que esté en la fila 0, columna 4.

  El intercambio se puede realizar de las siguientes maneras: **planta-planta**, **vacío-planta** o **vacío-vacío**, y en caso de colocar la misma posición dos veces se debe mantener igual el **Jardín**. A continuación se presenta un ejemplo de cada tipo de intercambio:

```
["X", "X", "X", "X"]          ["X", "X", "X", "X"]
["X", "S", "X", "X"] ------>  ["X", "P", "X", "X"]
["X", "X", "P", "X"]          ["X", "X", "S", "X"]
["X", "X", "X", "X"]          ["X", "X", "X", "X"]
```

Figura 5: Ejemplo de intercambio planta-planta para un input `1,1;2,2`

```
["C", "X", "X", "X"]          ["X", "X", "X", "X"]
["X", "X", "X", "X"] ------>  ["X", "X", "X", "X"]
["X", "X", "X", "X"]          ["X", "X", "X", "X"]
["X", "X", "X", "X"]          ["X", "X", "C", "X"]
```

Figura 6: Ejemplo de intercambio vacío-planta para un input `0,0;3,2`

Figura 7: Ejemplo de intercambio vacío-vacío para un input `0,0;0,0`

Finalmente, si el *input* es incorrecto, por ejemplo, si contiene carácteres no numéricos o números fuera del rango permitido, recibirás un mensaje en consola indicándote que la respuesta entregada no es correcta, y se devolverá al Menú Jardín.

- **Cultivar:** Esta opción permite cultivar las plantas disponibles en el **inventario de Plantas** de tu Jardín.

  Se desplegarán las plantas que tengas disponibles de cada tipo para que puedas escoger cual quieres plantar. Luego, si tienes al menos una planta de dicha especie podrás plantarla en tu jardín, indicando la posición con dos números separado por comas (“**,**”) que corresponden a la fila y columna en donde se ubicará la planta, respectivamente. En caso de seleccionar una posición donde **ya había una planta**, esta se debe **reemplazar** por la nueva.

  Finalmente, se debe indicar por consola si la acción fue exitosa o no. En caso de fallar, se debe especificar la causa del error (no poseer la planta o posición no válida) y se devolverá a Menú Jardín.

```
Plantas Disponibles:
[1] Solaretillo: 0
[2] Defensauce: 7
[3] Potencilantro: 10
[4] Aresauce: 0
[5] Cilantrillo: 2
[6] Fensaulantro: 1

[0] Volver al Menú Jardín

Escoge el número de la planta que quieres usar: 5
Escoge la posición donde quieres cultivar:  2,3
```

Figura 8: Ejemplo de menú de cultivar inicial

- **Regar:** Esta acción restablece la salud de las Plantas cultivadas de tu Jardín haciendo uso del método **Regar plantas** de cada una. Una vez finalizado, deberás imprimir la salud de las plantas antes y después del riego de cada una.

```
============================

Se han regado tus plantas

============================
Un Solaretillo en (3,2) ha subido su salud de 10 a 20.
Un Aresauce en (0,0) ha subido su salud de 10 a 12.
Un Aresauce en (0,1) ha subido su salud de 2 a 8.
```

Figura 9: Ejemplo de resultado de riego de plantas.

3.3. Laboratorio

Al mezclar cierto tipo de plantas, estas pueden fusionarse y generar una planta que contenga las características de ambas. Esta fusión no cuesta **Soles**, es gratis. [5]

En el laboratorio podrás mutar las plantas principales que se encuentren en tu inventario del **Jardín**. Se debe colocar la guía de combinaciones posibles, las plantas que tienes en el **Inventario Planta** y las opciones de las mutaciones: **Aresauce**, **Cilantrillo** y **Fensaulantro**.

Si cuentas con al menos una planta de cada tipo que deseas mutar se debe mostrar un mensaje que indique que se realizó la mutación correctamente, eliminar las plantas usadas del **Inventario Planta** y se añadirá una Planta Mutada, cuyas características se escogerán del archivo plantas.txt. En caso contrario, deberás indicar que no contabas con las plantas necesarias y permanecerás en el **Laboratorio** esperando otra indicación.

Las mutaciones existentes son:

- Solaretillo + Defensauce = Aresauce
- Solaretillo + Potencilantro = Cilantrillo
- Defensauce + Potencilantro = Fensaulantro

El detalle de las plantas y qué hacen se encuentra en Tipos de plantas.

```
============================
         Laboratorio
============================

             |Guía|

Solaretillo + Defensauce    = Aresauce
Solaretillo + Potencilantro = Cilantrillo
Defensauce  + Potencilantro = Fensaulantro

            | Tienes |

Solaretillo    Defensauce    Potencilantro
    99             7              4
          | Mutaciones |

  Aresauce    Cilantrillo   Fensaulantro
    [1]           [2]           [3]

[0] Volver al Menú principal

Indique su mutación a crear:
```

Figura 10: Ejemplo de Laboratorio

[5] Realizar una mutación no cuesta soles porque Hernán pudo hablar con **EL Árbol Madre** donde tuvieron una charla profunda sobre anime que desencadenó una amistad de por ~~raíz~~ vida

4. Entidades

En esta sección se detallarán las entidades que necesitarás para simular *DCCampesino*. Deberás utilizar conceptos claves como *herencia, clases abstractas, polimorfismo*, *properties*, etc., 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. Además, cualquier clase que programes deberá ser utilizada en el flujo de tu programa para obtener el puntaje completo en su modelación.

Las dos entidades principales de *DCCampesino* son el **Jardín** y las **Plantas**. Debes incluir como mínimo las características nombradas a continuación, pero siéntete libre de añadir nuevos atributos y métodos si lo estimas necesario.

4.1. Jardín

El Jardín es tu centro de operaciones en *DCCampesino*. Este contiene un tablero de *n*m* casillas en donde puedes poner tus plantas. Las plantas de tu jardín pueden ser atacadas aleatoriamente en cada oleada de zombies, o ser afectadas por eventos, por lo cual deberás organizarlas bien para sobrevivir.

A continuación, se presentan las características de un **Jardín**:

- **Tablero**: Matriz que contiene todas las plantas ya cultivadas del jardín.
- **Inventario Plantas**: Contiene las Plantas que llegan, explicado en Llegada de plantas, que aún no han sido posicionadas en el tablero. Inicialmente está vacío.
- **Soles**: Contiene el total en `int` de soles generados por las plantas. Inicialmente corresponde a `SOLES_INICIO`.
- **Temperatura**: Corresponde a un `int` entre `-5` y `25` que representa la temperatura actual del jardín en °C. Inicialmente corresponde a `TEMP_INICIAL`.

Además, debe poder realizar las siguientes acciones:

- **Cultivar planta**: Recibe una de las plantas del inventario y unas coordenadas del tablero. Agrega la planta en dicha posición. Si ya había una planta en ese lugar, esta última es reemplazada y se borra definitivamente. Este proceso se lleva a cabo en el menú Menú Jardín.
- **Regar plantas**: Hace que todas las plantas del tablero se rieguen, es decir, que recuperen salud. Para esto deberás definir dos parámetros enteros, `RIEGO_1` y `RIEGO_2`, que representan la salud recuperada, de los cuales se debe elegir aleatoriamente uno de los 2 cada vez que se riegue una planta. Este proceso se lleva a cabo en el menú Menú Jardín.
- **Mutar**: Este método permite combinar dos plantas básicas del inventario de plantas para obtener una planta mutante, de acuerdo a lo descrito en Tipos de plantas. Este proceso se lleva a cabo en el menú Laboratorio.
- **Presentarse**: Imprime la matriz del tablero que muestra la distribución actual de tus plantas en el jardín. Además, se listan las plantas del inventario. Para esto se **debe** sobreescribir el método `__str__` de la clase. Esto se logrará asignando una letra a cada planta (la letra de cada planta se puede ver en Archivos), mientras que la `X` representa un espacio vacío. El siguiente es un ejemplo de presentación:

```
*** Este es tu Jardín Actual ***
Temperatura: 10 °C

["S", "S", "D", "A"]
["C", "P", "X", "F"]
["C", "S", "X", "X"]
["X", "S", "X", "X"]

Te quedan:
3 solaretillo
1 defensauce
1 potencilantro
1 aresauce
2 cilantrillo
1 fensaulantro

Inventario Plantas:
["S", "S", "D", "A", "F", "P"]
```

Figura 11: Ejemplo de presentación de un Jardín

4.2. Plantas

Las plantas son la unidad principal en *DCCampesino* y las que defienden tu casa de las hordas de **Los Zombies**.

Las plantas poseen diversas estadísticas y habilidades en función de su clase, que dictaminará cómo se desenvolverán en el jardín. A continuación, se presentarán las diferentes características básicas de las plantas:

- **Tipo**: Corresponde a un `str` que representa el tipo de la planta.
- **Vida Máxima**: Corresponde a un `int` entre `0` y `100` que representa cantidad máxima de puntos de vida que puede tener la planta.
- **Vida**: Corresponde a un `int` que representa la cantidad de vida actual de la planta. No puede tener un valor menor a `0` o mayor a la vida máxima de la planta.
- **Resistencia**: Corresponde a un `int` entre `0` y `40` que representa la resistencia de la planta a los ataques de zombies.
- **Resistencia térmica**: Corresponde a un `int` entre `-5` y `25` que representa la temperatura mínima que puede resistir la planta.
- **Congelación**: Es un `bool` que indica si la planta está congelada o no. Una planta se congela si su resistencia térmica es mayor a la temperatura del Jardín.
- **Altura**: Corresponde a un `int` entre `0` y `30` que representa la altura en centímetros de la planta.

Además, todas las plantas pueden realizar las siguientes acciones:

- **Regarse**: Permite curar el daño recibido por una planta. Debe recibir un `int` y sumarlo a la vida actual de la planta, manteniendo su valor dentro de los límites permitidos.

4.2.1. Tipos de plantas

Existen 6 diferentes tipos de Plantas, cada una con características especiales. Al momento de **Simular día**, la planta será afectada de forma distinta dependiendo de su vida y resistencia. Incluso puede cambiar las estadísticas de la horda de zombies actual dependiendo de su tipo. Se describen las características de cada planta a continuación:

Plantas básicas

- **Solaretillo**: Planta con poca vida y resistencia. Su función es generar soles. Tiene una característica adicional llamada **potencial** que se inicializa, por cada planta, como un número aleatorio entre `POTENCIAL_SOLARETILLO_MIN` y `POTENCIAL_SOLARETILLO_MAX`. Luego, la producción de soles final está dada por:

$$\texttt{round}\left(\texttt{CONSTANTE\_SOLES} \times potencial \times \frac{temperatura}{25} \times \frac{altura}{30}\right)$$

  Donde `CONSTANTE_SOLES` es una constante para el cálculo de la producción. Además, la producción nunca debe ser menor a **0**.

- **Defensauce**: Planta con mucha vida y resistencia. Su función es recibir daño mientras protege a las demás plantas. Tiene una característica adicional llamada `armadura`, un `int` que cuenta como vida adicional y **no se puede recuperar**. Es decir, si esta planta es dañada, primero se daña su armadura hasta que llegue a cero, y en el siguiente ataque se dañará su vida. Si el daño recibido es mayor al valor de armadura restante, esto no dañará la vida **en ese mismo ataque**. Regar esta planta solo curará la vida, pero no la armadura.

- **Potencilantro**: Planta con un poco más de vida y resistencia que las plantas productoras de soles. Su función es aumentar el potencial de los Solaretillos plantados en el jardín en un `AUMENTO_NUTRIENTE` %. Por lo que la producción de estos últimos pasa a ser:

$$\texttt{round}\left(\texttt{CONSTANTE\_SOLES} \times potencial \times \frac{temperatura}{25} \times \frac{altura}{30} \times (1 + \frac{aumento}{100})\right)$$

  El área potenciada es un cuadrado, es decir las ocho posibles casillas inmediatamente colindantes, a menos que la planta se encuentre en un borde o esquina del tablero, en cuyo caso potenciará menos casillas.

Plantas mutantes

- **Aresauce**: Esta planta es la combinación entre **Solaretillo** y **Defensauce**. Además de generar soles y poseer una `armadura`, reduce la capacidad de **Los Zombies** de robar soles en un `ANTI_ROBO` %, siendo este un `float` entre 0 y 1.

- **Cilantrillo**: Esta es una planta con características tanto de un **Solaretillo** como de un **Potencilantro**. Es decir, genera soles y potencia en un cuadro de 8 casillas (y se potencia a sí mismo también). Tanto su atributo de potencial como el de aumento deben ser mayores que el de sus clases padre, respectivamente en `AUM_POT_CIL` % y `AUM_AUM_CIL` %, siendo ambos un `float` entre 0 y 1.

- **Fensaulantro**: Esta planta es la combinación entre **Defensauce** y **Potencilantro**. Además de comportarse como las dos plantas anteriores, teniendo `armadura` y potenciación en 8 casillas, reduce el nivel de ataque de las oleadas de **Los Zombies** en un `RED_ATQ` %, siendo este un `float` entre 0 y 1.

Las plantas mutantes, que resultan de la combinación de otras dos, pueden ser creadas en el Laboratorio.

5. Simular día

El paso de un día es el suceso principal de la simulación, siendo el momento donde se deberá calcular la generación de soles según lo transcurrido, y por lo tanto marcará el progreso y puntuación del usuario.

Al inicio de cada día se deberá medir la temperatura del jardín y luego decidir si sucede o no un evento, aplicando un efecto a nuestro jardín en caso de ser necesario. Posteriormente se deberán calcular los soles generados por el jardín durante el día y luego ver si llegan plantas nuevas, para finalmente presentar el estado del jardín. El funcionamiento de cada una de estas partes será explicado a continuación.

5.1. Temperatura

Al comienzo del día, se deberá obtener y modificar la temperatura actual del Jardín en ese día. Esta temperatura corresponde a un valor aleatorio entre `TEMP_MIN_JARDIN` °C y `TEMP_MAX_JARDIN` °C.

5.2. Elección de evento

Se deberá decidir si sucederá un evento o no durante el día. Cada evento tiene una probabilidad de ocurrir en específico que dependerá del tipo de evento y la dificultad en que se esté simulado. Los valores serán entregados en el archivo eventos.txt.

Debes tener en cuenta que durante un día **no pueden suceder los 2 eventos al mismo tiempo**, por lo que se deberá intentar activar un evento detrás de otro hasta conseguir que se active exitosamente uno o que ningún evento sea activado. El orden de intento de activación debe ser: Oleada de Zombies y Helada. Los tipos de eventos y sus efectos son explicados a mayor profundidad en la sección Eventos.

A modo de ejemplo, suponga que tenemos una probabilidad de **40 %** para la activación del evento **Oleada de Zombies** y una de **30 %** para el evento **Helada**. Se deberá empezar intentando una activación de Oleada de Zombies con probabilidad 40 %. En caso de no activarse, se deberá intentar una activación de Helada con probabilidad 30 %, en caso de fallar, es un día sin eventos.

Tras tomar la decisión de activar algún evento o no, se deberá imprimir un mensaje informando el **evento activado** y el **efecto que tiene** sobre las plantas. En caso de no activarse ningún evento también debe informarse al usuario de esto, con el mensaje "Día tranquilo" sin una descripción.

```
============================================================
                Se ha activado el evento: Helada
============================================================
  Las plantas con baja temperatura verán reducida su producción
============================================================
```

Figura 12: Ejemplo de Evento al Simular día.

5.3. Calcular soles

Durante esta etapa de la simulación, se deberá calcular el total de soles generado por el jardín durante este día. Para ello, se deberá considerar si hay un evento activo y la producción de soles de cada planta del jardín. Durante este periodo, cada planta deberá imprimir un mensaje con el tipo de planta al que pertenece, posteriormente deberá imprimir su ubicación en el jardín y posteriormente deberá colocar la producción de soles que realizó durante el día.

Además, se deben mostrar los soles que se obtienen del **cielo** que corresponde a un valor aleatorio entre `MIN_SOLES` y `MAX_SOLES`.

Finalmente se deberá imprimir el total de soles obtenidos durante el día.

```
====================================

Se ha iniciado la producción de soles

====================================
Un Solaretillo en (3,2) ha producido 50 soles.
Un Aresauce en (1,3) ha producido 20 soles.
Un Cilantrillo (3,3) ha producido 60 soles
====================================

Se han obtenido 10 soles del cielo

En total se han recolectado 140 soles durante el día
```

Figura 13: Ejemplo de cálculo de soles

5.4. Llegada de plantas

Los fuertes vientos pueden provocar que lleguen plantas nuevas a tu inventario. La cantidad de plantas que lleguen es un numero aleatorio entre `NUM_MIN_PLANTA` y `NUM_MAX_PLANTA` y solo pueden llegar plantas básicas: **Solaretillo**, **Defensauce** y **Potencilantro**. Por cada planta nueva que llegue, su tipo deberá ser determinado de forma aleatoria entre las tres opciones. Además, las características de la planta deben ser tomadas del archivo plantas.txt dependiendo del tipo de planta que llegó. Finalmente, las plantas son guardadas en el inventario de Jardín y se deberá imprimir un mensaje con la cantidad de plantas que llegaron.

```
*** Han llegado 2 plantas a tu inventario ***
```

Figura 14: Ejemplo de Llegada de plantas

5.5. Presentación

Finalmente se deberá imprimir el estado final del jardín, esto quiere decir que se deberá mostrar la temperatura, la posición que ocupa cada planta en él e imprimir el número de plantas congeladas actualmente:

```
*** Este es tu Jardín Actual ***
Temperatura: 10 °C

["*S*", "S", "D", "A"]
["C", "P", "X", "F"]
["C", "*S*", "X", "X"]
["X", "*S*", "X", "X"]

Te quedan en inventario:
3 solaretillo
1 defensauce
1 potencilantro
1 aresauce
2 cilantrillo
1 fensaulantro
```

Figura 15: Ejemplo de presentación de un Jardín

6. Eventos

Dentro de la simulación de *DCCampesino*, se deberá simular el paso del tiempo mediante la sucesión de distintos días. Durante cada día puede suceder como **máximo** un **Evento**, los cuales tendrán distintos efectos en el comportamiento de tus plantas. La probabilidad de que ocurra cada evento varía según la dificultad del juego y está especificada en eventos.txt.

- **Heladas:** Durante una helada, la temperatura del jardín cambiará aleatoriamente a un valor que puede estar entre `TEMP_MIN_HELADA` °C y `TEMP_MAX_HELADA` °C. Todas las plantas con resistencia térmica superior a esta temperatura se congelarán. Si una planta está congelada no puede producir **soles** hasta que su resistencia térmica sea menor a la temperatura del jardín. Las plantas que no se congelen seguirán produciendo su cantidad normal de soles, más un bono de `SOLES_EXTRA_HELADA` durante el evento. Si una planta congelada no tiene como función generar soles, esta no se verá afectada.

  Para representar que una planta está congelada en el Jardín su letra pasa a estar encapsulada por ~~copos de nieve~~ asteriscos. Por ejemplo, en el siguiente jardín hay dos Solaretillos congelados, pero como el Defensauce tenía una resistencia térmica con un valor menor a las temperaturas se mantiene normal.

```
*** Este es tu Jardín Actual ***

["X", "X", "X"]
["X", "*S*", "X"]
["X", "D", "*S*"]

Te quedan en inventario:
2 solaretillo
1 defensauce
```

Figura 16: Ejemplo de Jardín con plantas congeladas

- **Oleadas de Zombies:** Cuando ocurre este evento, un grupo de `ZOMBIES_DIFICULTAD`[6] **zombies** atacará tu jardín, cada uno eligiendo una casilla diferente. Si una casilla está vacía, se imprimirá: "¡Un zombie intentó atacar la posición (x, y) pero estaba vacía!". Si hay una planta en la casilla, esta recibirá **daño** según la fórmula:

$$\texttt{max}(1, \texttt{round}\left(ataque \times \frac{40 - resistencia}{40}\right))$$

  y se debe imprimir: "`La planta TIPO_PLANTA en (x, y) ha perdido X puntos de vida`". Si la planta muere, el mensaje debe ser: "`La planta TIPO_PLANTA en (x, y) ha perdido X puntos de vida, lo que acabó con su vida`".

  Finalmente, por cada ataque que haya logrado hacer daño de parte de **Los Zombies**, se te robarán `SOLES_ROBADOS` de tu balance actual. Si tu saldo de soles es menor a la cantidad robada, tu balance se reducirá a cero. Esto se reflejará en consola con el mensaje: "`Los zombies lograron atacar X casillas, robando Y soles`". Si no se produce ningún ataque exitoso, el mensaje será: "`¡Ninguna de tus plantas fue atacada! No has sufrido robos de soles`".

[6] Siendo `DIFICULTAD` la dificultad actualmente elegida.

```
==================================

Durante la noche pasó una oleada de 3 zombies!

==================================
¡Un zombie intentó atacar (1, 2) pero estaba vacía!
La planta defensauce en (3, 0) ha perdido 12 puntos de vida.
La planta potencilantro en (3, 1) ha perdido 21 puntos de vida, lo que acabó con su vida.
Los zombies lograron atacar 2 casillas, robando 8 soles.
```

Figura 17: Ejemplo de anuncio de que ocurrió un evento

7. Archivos

Para el desarrollo de *DCCampesino* será necesaria la lectura y carga de archivos. Estos contendrán información relevante para la ejecución correcta de tu programa. El contenido y explicación de cada archivo será presentada a continuación.

Dentro de los archivos presentados a continuación se utilizarán siglas para referirse a los distintos tipos de plantas presentes en la simulación. La siguiente tabla representa la relación de cada sigla con su correspondiente planta.

Siglas	**Tipo**
X	Casilla vacía
S	Solaretillo
D	Defensauce
P	Potencilantro
A	Aresauce
C	Cilantrillo
F	Fensaulantro

Cuadro 1: Siglas de plantas

7.1. eventos.txt

Dentro de la simulación de *DCCampesino* existen distintos eventos que ocurren de forma aleatoria cada día. La probabilidad de que suceda cada evento vendrá dictada por el archivo eventos.txt el cual contendrá una linea por cada evento. Cada linea contendrá 4 elementos separados por un "**;**". El primer elemento corresponderá a las siglas de un evento, seguido de la probabilidad de ocurrencia del evento en las dificultades fácil, medio y difícil, en ese orden.

En la siguiente tabla se puede ver la relación entre las siglas y su correspondiente evento.

Siglas	**Tipo**
Z	Oleada de Zombies
H	Helada

Cuadro 2: Siglas de eventos

Por ejemplo en la primera fila de la siguiente figura se debe interpretar que el evento Oleadas de Zombies, tiene una probabilidad de ocurrencia de 10 % en dificultad fácil, un 20 % en dificultad media y un 30 % en dificultad difícil.

```
Z;0.1;0.2;0.3
H;0.5;0.3;0.9
```

Figura 18: Ejemplo de contenido **eventos.txt**

7.2. plantas.txt

Este archivo contiene las estadísticas iniciales que cada tipo de planta debe tener al ser iniciada junto al tablero o al llegar al inventario.

Cada linea del archivo corresponderá a un tipo de planta distinta con sus atributos, cada elemento estará separado por un "**;**". El primer elemento será el tipo de planta al cual corresponde la linea del archivo seguido de su valor de vida máxima, vida, resistencia, resistencia térmica, congelación y altura en ese orden.

Variable	**Tipo**
Tipo De Planta	`string de largo 1`
Vida Maxima	`int entre 0 y 100`
Vida	`int entre 0 y Vida Maxima`
Resistencia	`int entre 0 y 40`
Resistencia térmica	`int entre -5 y 25`
Congelación	`bool`
Altura	`int entre 0 y 30`

Cuadro 3: Valores válidos de cada atributo en **plantas.txt**

Debes tener en cuenta que el archivo contendrá un total de 12 líneas, mientras que los tipos de planta son tan solo 6. Esto se debe a que existen 6 líneas erróneas, esto quiere decir que presentan valores para los atributos que no pertenecen a los rangos anteriormente presentados y por tanto son lineas que deben ser descartadas. Es importante tener en cuenta que pese a ser lineas erróneas, se respetará el tipo de dato entregado en Valores válidos de cada atributo en **plantas.txt**. Por ejemplo, una resistencia errónea será siempre un **int** pero fuera del rango entre 0 y 10.

A continuación se presenta un ejemplo de archivo válido de `plantas.txt`, en este caso se puede ver en la línea 1 que los Solaterillos tienen una vida máxima de 30, una resistencia de 5, una temperatura de 10 y una altura de 20.

Por otro lado, se puede ver que en la segunda línea se presenta una planta Solaterillo, pero que contiene un valor de resistencia de 43, lo cual no está dentro del rango válido. Por lo que esta línea es errónea y debe ser descartada.

```
S;30;15;5;10;0;20
S;20;15;43;10;0;20
D;80;60;8;18;1;15
D;80;30;10;-20;0;10
P;20;15;3;10;1;20
P;-3;10;3;10;0;20
A;60;50;7;18;0;17
A;60;34;8;-12;0;18
C;25;20;4;10;0;20
C;25;20;4;10;0;20
F;60;50;6;18;0;17
F;67;60;6;18;0;40
```

Figura 19: Ejemplo de contenido **plantas.txt**

7.3. `jardines.txt`

En este archivo se encontrarán los jardines iniciales con los cuales se puede iniciar una partida. Debes tener en cuenta que los jardines pueden tener dimensiones no cuadradas y que pueden contener plantas ya cultivadas dentro. En caso de contener plantas iniciales, estas tendrán las estadísticas iniciales entregadas en el archivo plantas.txt para su tipo de planta en específico.

Cada linea del archivo contendrá la información de un tablero inicial. El formato de cada linea será el siguiente: primero encontrarás una palabra que representa el nombre del tablero. Luego, encontrarás el carácter "**!**"que representa la separación entre el nombre del tablero y su contenido.

En el contenido del tablero, cada "**fila**" vendrá separada por un "**;**" mientras que cada casilla de esta fila vendrá separada por una "**,**".

```
5X5_vacio!X,X,X,X,X;X,X,X,X,X;X,X,X,X,X;X,X,X,X,X;X,X,X,X,X
4x3_defensivo!D,X,D;X,X,X;X,X,X;D,X,D
1x1_vacio!X
potencia!S,X,S;X,C,X;X,X,X;X,C,X;S,X,S
```

Figura 20: Ejemplo de contenido **jardines.txt**

En el ejemplo anterior, se puede ver que el jardin `4x3_defensivo` debe ser interpretado como un jardín con 4 filas y 3 columnas. Además contiene 4 **Defensauces** ubicados en cada esquina. Lo que visualmente se traduce en:

```
D,X,D
X,X,X
X,X,X
D,X,D
```

Figura 21: Visualización de **4x3_defensivo**

7.4. `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:

```
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:

```
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***[7], ya que esta práctica se considera incorrecta y su uso conllevará una reducción en tu calificación.

8. `.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.

9. 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. Finalmente, el equipo docente se reserva el derecho de quitar

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

u otorgar puntaje nulo, en caso de considerar de que los objetivos de la tarea no se están cumpliendo adecuadamente para un ítem.

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**.

10. 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)**.**