Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2233 — Programación Avanzada 2021-1

20 de mayo de 2021
Actividad Formativa

Actividad Formativa 5

Decoradores

Entrega

- **Lugar:** En su repositorio privado de GitHub, en la **carpeta** `Actividades/AF5/`
- **Hora del *push*:** 12:00, sábado 22 de mayo

Importante: Antes de comenzar, comprueba que Git este funcionando correctamente en tu repositorio privado. Para esto, **sube los archivos base de la actividad de inmediato** (*add*, *commit*, *push*). Se espera que en esta actividad (así como en las demás actividades y tareas) utilices Git a lo largo de **todo tu desarrollo** como una herramienta, no sólo como un método de entrega. Es por esto que recomendamos enfáticamente que vayas subiendo tus cambios constantemente (***push***), ya que **problemas de último minuto** relacionados con la entrega y Git **no serán considerados**.

Introducción

A raíz de la cuarentena te has puesto a indagar en nuevos hobbies para pasar tu tiempo libre, probaste con varios deportes y algunas otras actividades recreativas pero finalmente te diste cuenta que lo tuyo era la cocina. Justo cuando decides comenzar con tu nuevo pasatiempo, ves por telegram que el DCC hará un concurso de comida online llamado **HELL's DCCocinando**. La idea consiste en cocinar ciertos platos en base a un desafío y quienes logren completar la tarea pasan a la siguiente etapa y así hasta que solo quede un ganador. Ya que justo estás cursando Programación Avanzada y además, junto a tu familia te has visto las 19 temporadas de Hell's Kitchen (cualquier parecido con el nombre del concurso del DCC es mera coincidencia), te sientes confiado en que puedes lograr el desafío y te animas a inscribirte. En este caso, decidiste usar tus conocimientos para programar un DCCocinero que cocinará por tí.

Sin embargo, te das cuenta que para esta instancia de **HELL's DCCocinando**, hay muchos casos especiales para los cuales tu DCCocinero no está capacitado: las recetas de tortas vienen encriptadas, y no incluyen el relleno entre cada topping. Usaste tu único permiso para llevarle una pizza a tus ayudantes de Programación, y no puedes ir al supermercado a comprar los ingredientes que faltan. Entre los ingredientes que ya tenías en casa, ¡algunos se han infectado con Progravirus! Te das cuenta que es el momento perfecto para que pongas a prueba tu conocimiento de **decoradores** e implementes las modificiaciones necesarias a tu `DCCocinero`.

HELL'S DCCocina

El flujo principal de esta actividad será simular la preparación de varias tortas a partir de los ingredientes que tengas disponibles en tu casa y de las recetas encriptadas de cada torta correspondiente. La clase

`DCCocina` es la que controlará el flujo principal del código. Esta recibe todas las recetas y se las va entregando al `DCChef`, quien será el encargado de preparar las tortas de la clase `Torta`, para finalmente escribir la receta final de cada torta cocinada. Las tortas a cocinar serán elegidas por el usuario desde la consola.

La idea principal de esta actividad es que puedas implementar 6 decoradores para poder cumplir exitosamente el desafío del concurso **HELL'S DCCocinando**. Cada decorador recibe un método de alguna clase que ya está creada y que no debes modificar, la idea es que el decorador modifique indirectamente esta función y entregue la función decorada para que se comporte como se solicita. Tienes las siguientes indicaciones:

- Tienes que desencriptar todas las recetas usando la función `desencriptar`, la cual se encuentra implementada.
- Deberás contabilizar cuántos ingredientes tienes para poder cocinar las tortas que se te piden, pero **OJO!** que por el Progravirus hay algunos ingredientes que ahora pueden estar infectados y tienes que eliminarlos antes de hacer el conteo general.
- Antes de comenzar a preparar una torta vas a tener que fijarte si tienes al menos un ingrediente en la despensa. En caso contrario, no se cocinará nada más, finalizando la actividad.
- Cuando estés leyendo una receta y se necesite de un ingrediente específico, pero ya no te queden cantidades en tu despensa, tendrás que improvisar para poder terminar de cocinar la torta. Deberás revisar la lista de preferencias de reemplazo (mediante la función `encontrar_preferencia`, que ya viene implementada) para escoger el ingrediente adecuado.
- Es tu misión que entre cada ingrediente que se le vaya agregando a la torta, haya una capa de relleno a modo de separación. Este relleno puede ser bizcocho, panqueque o milhoja, y se elige por el usuario en consola cada vez que se simule **HELL'S DCCocina**. Tal como los ingredientes, estos se pueden acabar, en cuyo caso debes terminar la torta luego de agregar una última capa de relleno, y luego terminar la simulación.
- Finalmente, debes escribir las recetas nuevas que generaste al improvisar los ingredientes en un archivo separado, dejando guardada esta nueva torta para la posteridad. Deberás encriptar estas recetas usando la función `encriptar`, para que los enemigos del DCC no puedan utilizarlas.

Archivos

Para los archivos `.csv`, la primera línea es el *header* que etiqueta las columnas.

- `recetas.csv`: Cada línea tiene una receta, donde el primer valor separado por una coma es el nombre de la torta y el resto son los ingredientes encriptados:

  `nombre_torta,ingrediente_encriptado_1,ingrediente_encriptado_2....`

  El orden de los ingredientes en el archivo está según el orden en que se agregan a la torta.

- `ingredientes_totales.csv`: Este archivo contiene por cada línea el nombre de un ingrediente disponible en la despensa y la cantidad total de este separados por una coma:

  `ingrediente,cantidad`

- `recetas_nuevas.csv`: Este archivo tiene solo el *header*. Acá se deben escribir las recetas nuevas que se generen en la simulación, en el mismo formato que `recetas.csv`.

- `preferencias.csv`: Este archivo contiene el orden de todos los ingredientes según la preferencia para su posible sustitución en caso de necesitarlo, separados por comas:

  `ingrediente_original,ingrediente_reemplazo`

- `main.py`: Este archivo se ejecuta para simular **HELL'S DCCocina**. **No tienes que modificar este archivo.**

- `clases.py`: Este archivo contiene a las clases implementadas `DCChef`, `Torta` y `DCCocina`. **Sólo debes modificar este archivo agregando decoradores a métodos.**

- `funciones.py`: Este archivo tiene funciones útiles para el funcionamiento del programa. **No tienes que modificar este archivo**, pero debes usar las funciones que están acá para implementar los decoradores. Contiene las siguientes funciones:

  - `encontrar_preferencia(ingrediente : str) →str`: recibe un ingrediente y retorna un reemplazo óptimo, determinado por las preferencias en `preferencias.csv`. Debes usar esta función.

  - `desencriptar(palabra:str) →str`: recibe una palabra encriptada y la retorna desencriptada. Debes usar esta función.

  - `encriptar(palabra:str) →str`: recibe una palabra desencriptada y la retorna encriptada. Debes usar esta función.

  - `log(anuncio:str, tipo_anuncio:str)`: recibe un `anuncio` que contiene información de lo que está ocurriendo y según el `tipo_anuncio` indicado (general, de relleno, ingredientes, u otros) imprime el anuncio en consola con un formato específico. Luego duerme el programa `LOG_SLEEP_S` segundos. Puedes usar esta función para realizar anuncios ordenados de lo que ocurre en los decoradores.

  - `recibir_input(lista_opciones: list) →str`: Esta función recibe las posibles opciones a desplegar en el menú en consola. Pide un *input* al usuario, chequea que sea válido y retorna la elección de la lista de opciones. Si el *input* ingresado no es válido, se sigue preguntando hasta que lo sea. **No debes usar esta función.**

  - `preguntar_rellenos() →str`: permite que el usuario ingrese el relleno que quiere usar para la simulación, y lo retorna. **No debes usar esta función.**

- `decoradores_cocina.py`: En este archivo se encuentran decoradores que ayudan en la cocina: `desencriptar_receta`, `encriptar_receta`, e `ingredientes_infectados`. **Debes modificar este archivo.**

- `decoradores_cocinero.py`: En este archivo se encuentran decoradores que ayudan al cocinero: `improvisar_toppings`, `capa_relleno`, y `revisar_ingredientes`. **Debes modificar este archivo.**
- `parametros.py`: Tal como en las tareas, este archivo tiene parámetros necesarios para el funcionamiento del programa. **No tienes que modificar este archivo.**

Clases implementadas

A continuación se describen las 3 clases con las que vas a tener que trabajar para esta actividad, que se encuentran en el archivo `clases.py`. **Todas estas clases ya vienen implementadas y no debes modificar la definición de sus atributos y métodos.** Sin embargo, **debes decorar los métodos que lo necesiten**.

Clase Torta

Esta clase hereda de `list`, y se encarga de almacenar los ingredientes que la componen. **NO debes modificarla**. Tiene los siguientes atributos:

- `self.finalizada:bool`: indica si la torta está terminada (`True`) o no (`False`).
- `self.name:str`: nombre de la torta.

Además tiene el siguiente método:

- `def __str__(self) →str` : retorna el atributo `self.name` de la torta.

Clase DCChef

Esta clase se encarga de preparar cada torta según una receta ya desencriptada que recibirá, por lo tanto tiene todos los métodos necesarios para cocinar la torta de inicio a fin. **NO debes modificar esta clase**. Tiene los siguientes atributos:

- `self.tipo_relleno: str`: Este es un atributo global de la clase, que tiene el tipo de relleno que se va a utilizar a lo largo de la simulación (escogido por el usuario).
- `self.ingredientes_disponibles: dict`: Diccionario donde las llaves corresponden a nombres de ingredientes, y el valor a la cantidad restante. Su formato es el siguiente:

  ```
  {"mermelada": 5, "manjar": 15, ...}
  ```

  Este atributo se pide como parámetro al momento de crear la instancia DCChef.
- `self.relleno_restante : int`: Es la cantidad de relleno que queda.

Además cuenta con los siguientes métodos:

- `agregar_topping(nombre_ingrediente:str, torta: Torta)`: método que agrega una capa del ingrediente a la torta (ya sea manjar, mermelada, crema, etc).
- `cocinar_torta(name: str, receta: list) →Torta`: método que recibe el nombre de una torta y su receta desencriptada. La receta corresponde a una lista con los ingredientes necesarios para preparar la torta, en el orden que se deben agregar. Crea una `Torta` vacía y va agregando cada ingrediente usando el método `agregar_topping` en orden. Cuando se hayan agregado todos los ingredientes (o sus reemplazos, en caso de que no queden), la torta se finaliza y se retorna el objeto.

Clase DCCocina

Esta clase se encarga de simular el programa: carga y contiene las recetas y tiene también a un DCChef. **NO debes modificar esta clase**. Tiene los siguientes atributos:

- `self.chef`: `DCChef`: instancia de DCChef encargado de cocinar las tortas.
- `self.recetas`: `dict`: diccionario de recetas de la siguiente forma:

```
{ "nombre" : ["ingrediente_1", "ingrediente_2", ... , "ingrediente_n"] }
```

- `self.tortas`: `list`: lista que contiene los objetos de tortas que se han completado.

Y tiene los siguientes métodos:

- `def leer_recetas(self, archivo:str)` →`dict`: este método lee el archivo de recetas y lo retorna en un diccionario de la siguiente forma:

```
{ "nombre_receta" : ["ingrediente_1", "ingrediente_2", ... , "ingrediente_n"] }
```

- `def revisar_despensa(self, archivo : str)` →`dict`: este método lee el archivo `ingredientes_totales.csv` y los retorna en un diccionario de la siguiente forma:

```
{ "ingrediente" : cantidad }
```

- `def escribir_receta(self, receta:Torta)`: este método guarda en `recetas_nuevas.csv` la receta recibida. La Torta es una lista de forma:

```
["ingrediente_1", "ingrediente_2", ... , "ingrediente_n"]
```

  El atributo `self.name` de `Torta` guarda el nombre de la torta. Las recetas son guardadas en el mismo formato que el archivo `recetas.csv`.

- `def simular_cocina(self)`: lleva a cabo la simulación de la cocina: lee los archivos de ingredientes y recetas, instancia al `DCChef` y maneja los *inputs* para cocinar tortas y luego almacenarlas.
- `def print_seleccion(self)`: Este método imprime el nombre de las tortas y un número que representa el input que se debe introducir para elegir la torta a cocinar (este método se llama antes de cocinar una torta).

Parte I: Decoradores de Cocina

En esta parte, deberás implementar los decoradores que tienen que ver con el encriptado y desencriptado de recetas, y la revisión de ingredientes infectados.

Los decoradores que tienes que completar en esta parte están en el archivo `decoradores_cocina.py` y son los siguientes:

`desencriptar_receta(metodo_original: function)` →`dict`:

Este decorador debe hacer que el método `leer_recetas` de la clase `DCCocina` retorne un diccionario de recetas *desencriptadas*. Para hacer esto, se debe usar la función `desencriptar` del archivo `funciones.py`.

`encriptar_receta(metodo_original: function) →str:`

Este decorador debe hacer que el método `escribir_recetas` de la clase `DCCocina` encripte las recetas **antes** de escribirlas en el archivo `nuevas_recetas.csv`. Para hacer esto, se debe usar la funcion `encriptar` del archivo `funciones.py`.

`ingredientes_infectados(probabilidad_infectado: float) →decorador:`

Este decorador se preocupa de revisar que los ingredientes no estén infectados antes de que sean contabilizados por `DCCocina` en su método `revisar_despensa`. Este decorador recibe un argumento que corresponde a la probabilidad de encontrar un ingrediente infectado y es de valor `PROBABILIDAD_INFECCION` (importado desde `parametros.py`). Por cada tipo de ingrediente, se debe calcular la probabilidad y compararla con el argumento dado para revisar si hay una infección. En ese caso, se debe reducir la cantidad total de ese ingrediente en 1, antes de ser agregado al total de ingredientes. Si se encuentra un ingrediente infectado, se debe imprimir un mensaje en la consola que lo indique.

Parte II: Decoradores del Cocinero

En esta parte debes implementar las funciones que tienen que ver con la revisión y uso de ingredientes cuando se está armando una torta. Las funciones que tienes que completar en esta parte son las siguientes:

`capa_relleno(tipo_relleno: str) →decorador:`

Este decorador se encarga de agregar una capa de relleno antes de que se le agregue un ingrediente a la torta, en el método `agregar_topping` de la clase `DCChef`. El decorador debe recibir como argumento el tipo de relleno usado, que se encuentra almacenado en el **atributo de clase** `tipo_relleno` de `DCChef`. Debe revisar que quede relleno **antes** de que se agregue un ingrediente nuevo; en caso contrario, se termina la torta sin agregar otra capa de relleno. Se debe imprimir un mensaje en cada caso: si queda relleno o no.

`improvisar_toppings(funcion_original: function):`

Este decorador se preocupa de verificar que el ingrediente que se va a agregar a una torta en el método `agregar_topping` de la clase `DCChef` esté en *stock*. Si no lo está, se encarga de reemplazarlo por uno similar, usando la función `encontrar_preferencia` que se encuentra en el archivo `funciones.py`. Es posible que tampoco quede suficiente de esta preferencia, en cuyo caso debes ocupar este nuevo ingrediente para encontrar otra preferencia. Se debe imprimir en consola el ingrediente que falta y el ingrediente por el que se va a reemplazar.

`revisar_ingredientes(funcion_original: function):`

Este decorador se preocupa de revisar que hayan suficientes ingredientes antes de que el `DCChef` comience a preparar una torta con su método `cocinar_torta` (tienen que haber al menos el número de ingredientes que tiene la receta). De no haber suficientes ingredientes, se debe levantar un `ValueError`, e imprimir en consola lo ocurrido.

Notas

- Es importante que entiendas que hace cada método para poder crear el decorador de manera efectiva. Todos los decoradores van en métodos de `clases.py`, pero no han sido asignados.
- Al asignar los decoradores, recuerda que algunos reciben argumentos y otros no.

- Recuerda que el primer argumento de un método, `self`, es una instancia de la clase. Puedes ocupar este argumento para acceder a los atributos de esa instancia desde los decoradores. Del mismo modo, si llaman al metodo desde los decoradores, no olvides incluir a la instancia como primer argumento.
- Pueden usar la función `log` para hacer `print()` más ordenados.

Objetivos

- Implementar distintos decoradores para añadir funcionalidades a métodos.
- Resolver diversos problemas usando decoradores.