Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2233 — Programación Avanzada 2022-1

21 de Abril de 2022
Actividad Sumativa

Actividad Sumativa 2

Threading

Entrega

- **Lugar:** En su repositorio privado de GitHub, en la **carpeta** `Actividades/AS2/`
- **Hora del *push*:** 16:40

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

¿Aburrido de las filas para calentar en los microondas? ¿Aburrido de las filas para comprar comida? ¡Llegó tu salvación! Con el retorno a las clases presenciales, el profesor Nicolás Elliott decidió aprovechar su curso del instituto "Le Cordon Bleu" para abrir su propio restaurant y convertirse en **DCChef D'Elliot**. Sin embargo, al contratar al personal se dió cuenta que había un gran flujo de personas en el restaurant, por lo que te solicitó a ti, alumno estrella de programación avanzada, que lo ayudes a simular el flujo en el restaurant con Threads.

Flujo del programa

El programa consiste en una simulación de un restaurant en que los clientes hacen pedidos, los meseros los llevan a la cocina, los cocineros los cocinan y finalmente los meseros llevan de vuelta los platos listos a la mesa. El flujo del restaurant del Chef D'Elliot comienza al ejecutar el archivo `main.py`, donde se cargan los archivos de datos y se instancian las entidades `Cocina`, `Cocinero` y `Mesero` y se inicia el thread de la clase `Cocina`, la cual a su vez inicia múltiples threads de la clase `Cocinero` y `Mesero`. Así, a medida que se va leyendo el archivo de pedidos, los meseros se encargan de tomar los pedidos y agregarlos a la cola de pedidos de la cocina, tras lo cual un cocinero los cocina. Luego, se agrega el plato listo a la cola de pedidos listos de la cocina y un mesero se encarga de llevarlo a la mesa respectiva. Finalmente, una vez que hayan acabado los pedidos, el restaurant se cierra y termina el programa.

Archivos

Archivos de datos

- `ingredientes.csv`: En este archivo se encuentran las recetas de todos los platos que prepara el restaurant. Cada línea contiene información de un plato, junto con los ingredientes que necesita separados por comas y la cantidad que requiere de estos, separado por punto y coma (";") de la siguiente forma:

  `nombre_plato,ingrediente_1;cantidad_1,ingrediente_2;cantidad_2,...`

- `pedidos.csv`: En este archivo se encuentra la información de todos los pedidos que pedirá cada mesa del restaurant. Cada línea contiene la mesa que realizó el pedido, seguida de todos los platos que pidió separados por comas, de la siguiente forma:

  `id_mesa,nombre_plato_1,nombre_plato_2,...`

Archivos de código

- `lectura_archivo.py`: Contiene las funciones `cargar_pedidos`, `cargar_ingredientes` y `generar_bodega` para cargar los datos del programa. No debes modificarlo
- `parametros.py`: Contiene los parámetros necesarios para la ejecución del programa . No debes modificarlo
- `main.py`: Contiene la simulación del programa. No debes modificarlo
- `cocina.py`: Contiene la clase `Cocina`. Debes modificarlo
- `entidades.py`: Contiene las clases `Persona`, `Cocinero` y `Mesero`. Debes modificarlo

Parte 1: Modelación de Entidades

En esta parte deberás completar las clases `Persona`, `Cocinero` y `Mesero` del archivo `entidades.py`, las cuales representan a los encargados de preparar y llevar los platos del restaurante.

- `class Persona`: Corresponde a una clase **abstracta** que representa a las personas que trabajan junto a DCChef D'Elliot. De esta clase heredan las subclases `Cocinero` y `Mesero`. Debes hacer que esta clase herede de la clase `Thread`. Además, debes definir los siguientes locks como atributos de clase: `lock_bodega`, `lock_cola_pedidos`, `lock_cola_pedidos_listos`. Debes modificarlo
  - `def __init__(self, nombre: str)`: Inicializador de la clase que recibe un `str` correspondiente al nombre de la persona. Además, la clase posee los siguientes atributos: No debes modificarlo

    - `self.nombre`: `str` que corresponde al nombre de la persona.
    - `self.disponible`: `bool` que representa si la persona está disponible o si está ocupada.
    - `self.trabajando`: `bool` que indica si la persona está dentro de su horario de trabajo.
    - `self.daemon`: `bool` que indica si el `Thread` es `Daemon`.
  - `def run(self)`: Método abstracto que representa la ejecución del thread. **No debes modificarlo**

- `class Cocinero`: Esta clase hereda de la clase `Persona` y representa a un cocinero encargado de preparar los platos. **Debes modificarlo**
  - `def __init__(self, nombre: str, cocina: Cocina)`: Inicializador de la clase que recibe un `str` que corresponde al nombre del cocinero y la instancia de la clase `Cocina`. Deberás definir el atributo `evento_plato_asignado` como un evento. Además, la clase posee el siguiente atributo ya implementado: **Debes modificarlo**
    - `self.lugar_trabajo`: Instancia de `Cocina` que representa al lugar en que trabaja el cocinero.
  - `def run(self)`: Representa la ejecución del thread. En este método debes hacer que mientras el cocinero esté en su horario laboral, espere a que se le asigne un plato con el evento `evento_plato_asignado`. En el momento que se le asigne un plato, el cocinero empezará a hacer los preparativos antes de cocinar, lo cual debes simularlo esperando un tiempo aleatorio entre 1 y 3 segundos[1]. Finalmente, debes llamar al método `cocinar`. **Debes modificarlo**
  - `def cocinar(self)`: Este método representa las acciones del cocinero cuando está cocinando. Primero, debes cambiar el valor del atributo `disponible` a `False` y sacar un plato de la cola de pedidos con el método `sacar_plato`. Luego, debes imprimir un mensaje que indique que cocinero está cocinando que plato y buscar los ingredientes para el plato con el método `buscar_ingredientes`. Después, para simular el tiempo de cocina debes esperar un tiempo aleatorio entre 1 y 3 segundos. Una vez que el plato esté listo, debes agregar el plato a la cola de pedidos listos con el método `agregar_plato` y resetear `evento_plato_asignado`. Finalmente, debes volver a cambiar el valor del atributo `disponible` a `True`. **Debes modificarlo**
  - `def sacar_plato(self)->plato: tuple`: En este método debes encargarte de sacar el primer plato de la cola de pedidos de la `Cocina` y retornarlo. Cada plato corresponde a una tupla de 2 elementos en que el primero es el número de mesa que ordenó el plato (`str`), y el segundo es el nombre del plato que se ordenó (`str`). **Debes asegurarte de que sólo un cocinero a la vez pueda sacar un plato de la cola de pedidos**. **Debes modificarlo**
  - `def buscar_ingredientes(self, plato: tuple, recetas: dict, bodega: dict)`: Este método se encarga de sacar de la bodega los ingredientes necesarios para el plato. Recibe 3 argumentos: el primer argumento corresponde al **plato**, que es una tupla de 2 elementos en que el primero es el número de mesa que ordenó el plato (`str`), y el segundo es el nombre del plato que se ordenó (`str`). El segundo argumento corresponde al diccionario de **recetas**, en que cada llave es el nombre de un plato (`str`) y cada valor es una lista de tuplas, en que cada tupla contiene dos elementos: el nombre de un ingrediente (`str`) y la cantidad de dicho ingrediente que lleva el plato (`str`). El tercer argumento corresponde al diccionario de la **bodega**, en que cada llave es el nombre de un ingrediente (`str`) y cada valor es la cantidad de dicho ingrediente que hay en la bodega (`int`).

[1] Para esto puedes utilizar el método `sleep` de la libreria `time`.

Para modelar este método, al inicio debes imprimir un mensaje que indique qué cocinero está buscando ingredientes en la bodega para qué plato. Luego, debes buscar los ingredientes que lleva el plato, a través del diccionario `recetas`. Finalmente, debes disminuir la cantidad de cada ingrediente que lleva el plato en la bodega.[2]. **Debes asegurarte de que sólo un cocinero a la vez pueda sacar alimentos de la bodega**. Debes modificarlo

- `def agregar_plato(self, plato: tuple)`: Este método recibe como argumento un plato y debes encargarte de agregarlo al final de la cola de pedidos listos de la `Cocina`. **Debes asegurarte de que sólo un cocinero a la vez pueda agregar un plato a la cola de pedidos listos**. Debes modificarlo

- `class Mesero`: Esta clase hereda de la clase `Persona` y representa a un mesero encargado de llevar los pedidos a la cocina y llevar los platos listos a las mesas. Debes modificarlo

  - `def __init__(self, nombre: str)`: Inicializador de la clase que recibe un `str` que corresponde al nombre del mesero. Debes definir el atributo `evento_manejar_pedido` como un evento. Debes modificarlo

  - `def run(self)`: Representa la ejecución del thread. En este método debes hacer que mientras el mesero esté en su horario laboral, si el mesero está disponible, debes activar el evento `evento_manejar_pedido`. Debes modificarlo

  - `def agregar_pedido(self, pedido: tuple, cocina: Cocina)`: Este método se encarga de agregar un pedido a la cola de pedidos de la cocina. Recibe como argumentos el **pedido**, que es una tupla con el número de mesa que hizo el pedido (`str`) y con el nombre del plato que se ordenó (`str`), y una instancia de (`Cocina`). Para modelar esto, primero debes resetear el evento `evento_manejar_pedido`. Luego, para simular la llegada de clientes debes esperar un tiempo aleatorio entre 1 y 2 segundos, para después agregar el pedido a la cola de pedidos de la `Cocina`. Finalmente debes activar el evento `evento_manejar_pedido`. **Debes asegurarte de que sólo un mesero a la vez pueda acceder a la cola de pedidos**. Debes modificarlo

  - `def entregar_pedido(self, cocina: Cocina)`: Este método representa la entrega de pedidos a las mesas por los meseros. Para simular esto, primero debes resetear el evento `evento_manejar_pedido`. Luego, debes simular la entrega del plato, esperando un tiempo aleatorio entre 1 y 3 segundos para luego ejecutar el método `pedido_entregado` con sus parámetros respectivos, en donde el pedido a entregar corresponde al primer pedido de la cola de pedidos listos de la `Cocina`. Finalmente, debes imprimir un mensaje que indique que mesero está entregando un pedido a que mesa. **Debes asegurarte de que sólo un mesero a la vez pueda acceder a la cola de pedidos listos**. Debes modificarlo

  - `def pedido_entregado(self, pedido: tuple)`: En este método debes imprimir un mensaje que indique que plato de que mesa fue entregado, y activar el evento `evento_manejar_pedido`. Debes modificarlo

Parte 2: Cocina

En esta parte deberás completar la clase `Cocina` del archivo `cocina.py`, la cual se encarga de asignar los pedidos a los cocineros y los pedidos listos a los meseros para llevar a las mesas.

[2]Puedes asumir que en la bodega existe una cantidad suficiente de ingredientes que necesitan los platos

- `class Cocina`: Esta clase corresponde a la cocina del restaurant. Contiene los pedidos en proceso y los pedidos listos. Además maneja la bodega y el recetario de los platos. Es la encargada de asignarle los pedidos a los cocineros y una vez que estén listos a los meseros. Incluye los siguientes métodos: No debes modificarlo
  - `def __init__(self, lista_cocineros: list, lista_meseros: list, bodega: dict,recetas: dict)`: Inicializador de la clase, posee los siguientes atributos ya implementados: No debes modificarlo
    - `self.cola_pedidos`: `deque` que representa la cola de pedidos por cocinar.
    - `self.cola_pedidos_listos`: `deque` que representa la cola de pedidos listos para ser entregados por el mesero.
    - `self.cocineros`: `list` que contiene a todas las instancias de la clase `Cocinero`.
    - `self.meseros`: `list` que contiene a todas las instancias de la clase `Mesero`.
    - `self.bodega`: `dict` donde la llave corresponde al nombre de un ingrediente y el valor a la cantidad disponible de este.
    - `self.recetas`: `dict` donde la llave corresponde al nombre del plato y el valor corresponde a una lista de tuplas donde el primer valor de la tupla corresponde a un ingrediente y el segundo valor a la cantidad necesaria de ese ingrediente.
    - `self.abierta`: `bool` que indica si la cocina esta abierta.
  - `def initialize_threads(self)`: En este método debes encargarte de iniciar los threads de todos los cocineros y meseros en la `Cocina`. Debes modificarlo
  - `def asignar_cocinero(self)`: Este método se encarga de asignar los platos a los cocineros, mientras la cocina este abierta, debes simular una espera de un segundo y luego verificar si hay pedidos en la cola de pedidos, en caso de que hayan, debes buscar si hay algún cocinero disponible y activar el evento `evento_plato_asignado` de este cocinero. Debes modificarlo
  - `def asignar_mesero(self)`: Este método se encarga de asignar los pedidos a los meseros. Para esto, mientras la cocina esté abierta, debes esperar un tiempo de un segundo y después verificar si hay pedidos en la cola de pedidos listos, en caso que hayan, debes buscar si hay algún mesero disponible y activar el evento `evento_manejar_pedido` de este mesero. Luego, debes llamar al método `entregar_pedido` del mesero con sus respectivos argumentos. Finalmente, cuando la cocina cierre debes llamar al método `finalizar_jornada_laboral`. Debes modificarlo
  - `def finalizar_jornada_laboral(self)`: Este método se encarga de finalizar la jornada laboral de todos los trabajandores del restaurant. No debes modificarlo

Notas

- Recuerden que se puede consultar el material de contenidos e internet, pero **NO** consultar a sus compañeros.
- Puedes verificar si un evento está activo con `Evento.is_set()`, retorna un `boolean`.

Requerimientos

- (0.50 pts) Modelar correctamente la clase `Persona`.

- (0.50 pts) Modelar correctamente la clase `Cocina`.
- (3.00 pts) Clase `Cocinero`:
  - (0.25 pts) Definir correctamente el método `def __init__()`
  - (0.25 pts) Definir correctamente el método `def run()`
  - (0.50 pts) Definir correctamente el método `def cocinar()`
  - (0.50 pts) Definir correctamente el método `def sacar_plato()`
  - (1.00 pt) Definir correctamente el método `def buscar_ingredientes()`
  - (0.50 pts) Definir correctamente el método `def agregar_plato()`
- (2.00 pts) Clase `Mesero`:
  - (0.25 pts) Definir correctamente el método `def __init__()`
  - (0.25 pts) Definir correctamente el método `def run()`
  - (0.50 pts) Definir correctamente el método `def agregar_pedido()`
  - (0.75 pts) Definir correctamente el método `def entregar_pedido()`
  - (0.25 pts) Definir correctamente el método `def pedido_entregado()`