PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE | la clase ejecutiva

Instrucciones Miniproyecto 3

Descripción

Debe construir un programa orientado a objetos que modele el funcionamiento de un restaurante. En el restaurante trabajan cocineros, los cuales preparan platos, tanto bebestibles como comestibles, con diferentes niveles de dificultad en su preparación. Los cocineros poseen niveles de habilidad que influyen en la calidad final de cada plato. Los platos son enviados a los clientes del restaurante mediante un grupo de repartidores quienes también trabajan para el restaurante y deben entregar sus pedidos según un tiempo determinado. Tanto los cocineros como los repartidores tienen una cantidad de energía que con cada acción que realizan va disminuyendo, después de la cual ya no pueden trabajar si no les queda energía positiva. Finalmente, los clientes reciben sus pedidos y efectúan una evaluación en base a la calidad de los platos recibidos. La simulación debe entregar como resultado un valor de calificación para el restaurante en base al servicio ofrecido y la evaluación de los clientes.

El trabajo a realizar

Se le entregará un archivo MP3.zip con cuatro archivos: main.py, restaurante.py, platos.py y personas.py. En estos archivos se construirán las clases y funciones que modelan la solución. El archivo **main.py es el código principal** que permite probar su solución final.

Debe completar los siguientes pasos para completar la simulación del restaurante.

1. **En el archivo platos.py** debe definir las siguientes clases:
   - 1.1. Una clase `Plato`. Debe contener los siguientes atributos:
     - `nombre:` Nombre del plato. Es un string que se recibe como parámetro de inicialización.
     - `calidad:` Calidad del plato. Es un int que se inicializa en 0.
   - 1.2. Una clase `Bebestible` que **herede** de `Plato`. Debe contener los siguientes atributos adicionales:
     - `tamano:` El tamaño del bebestible. Se inicializa eligiendo aleatoriamente entre los strings `"Pequeño"`, `"Mediano"` y `"Grande"`.
     - `dificultad:` La dificultad del plato. Si el `tamaño` es `"Pequeño"` se inicializa en 3, si el tamaño es `"Mediano"` se inicializa en 6 y finalmente, si el tamaño es `"Grande"` se inicializa en 9.
     - `calidad:` La calidad de un `bebestible`. Debe inicializarse como un número entero aleatorio entre 3 y 8.
   - 1.3. Una clase `Comestible` que **herede** de `Plato`. Debe contener los siguientes atributos adicionales:
     - `dificultad:` La dificultad del plato. Debe inicializarse en un número entero aleatorio entre 1 y 10.
     - `calidad:` La calidad de un `comestible`. Debe inicializarse como un número entero aleatorio entre 5 y 10.

`(*) Las clases Comestible y Bebestible deben tener un método __str__ que retorne un string con su nombre, tipo de plato, calidad y dificultad.`

2. En el **archivo** `personas.py` debe definir las siguientes clases:

2.1. Una clase `Persona`. Debe contener los siguientes atributos:

- `nombre`: Nombre de la persona. Es un string que se recibe como parámetro de inicialización.

2.2. Una clase `Repartidor` que **herede** de `Persona`. Debe contener los siguientes atributos adicionales:

- `tiempo_entrega`: Tiempo base que se demora en entregar un pedido en segundos. Se recibe como parámetro de inicialización que debe ser un número aleatorio entre 20 y 30.
- `energia`: Energía del repartidor. El valor se inicializa como un número aleatorio entre 75 y 100.

También debe tener (como mínimo) el siguiente método:

- `repartir(pedido)`: Recibe el argumento `pedido`, que es una lista de platos. Este método hace dos cosas. Primero, debe disminuir la `energia` del repartidor según un `factor_tamaño`. Para el `factor_tamaño`, si el `pedido` es de 2 o menos platos, este factor es 5. Si el `pedido` es de 3 o más platos, el factor es de 15.

  Además, el método debe calcular el tiempo que se demora el repartidor en entregar el pedido, el cual es equivalente al `tiempo_entrega` del repartidor ponderado por un `factor_velocidad`. Si el `pedido` es de 2 o menos platos, este factor es de 1.25, si el `pedido` es de 3 o más platos, el factor es de 0.85.

  El método debe imprimir un string del estilo "El repartidor X se ha demorado Y perdiendo Z de energía", donde X es el nombre, Y es el tiempo que se demoró, y Z es la energía perdida. El formato exacto del string da lo mismo, lo importante es que se reflejen esos 3 datos.

  Finalmente el método debe retornar el tiempo de demora del pedido.

2.3. Una clase `Cocinero` que **herede** de `Persona`. Debe contener los siguientes atributos adicionales:

- `habilidad`: Habilidad del cocinero. Se recibe como parámetro de inicialización que debe ser un número aleatorio entre 1 y 10.
- `energia`: Energía de la persona. El valor de `energia` de un cocinero se inicializa en un número aleatorio entre 50 y 80.

También debe tener (como mínimo) el siguiente método:

- `cocinar(informacion_plato)`: Recibe el argumento `informacion_plato`, que es una lista con el nombre y el tipo del plato a cocinar, donde el tipo puede ser `"Bebestible"` o `"Comestible"`. Este método hace 2 cosas.

  Primero, debe crear una instancia de la clase `Bebestible` o `Comestible` dependiendo del tipo. Luego disminuye la energía del cocinero según el plato. En caso de que se haya cocinado un `Bebestible`, el cocinero pierde 5 de `energía` si el `Bebestible` es "pequeño", 8 si es "mediano" y 10 si es "grande". En cambio, si se cocinó un `Comestible`, se pierde 15 de `energía`.

  Luego el parámetro `calidad` del plato cocinado (bebestible o comestible) debe ser multiplicado por un `factor_calidad`. Si la `dificultad` del plato es mayor a la `habilidad` del cocinero, este factor es 0.7, en caso contrario es 1.5.

  El método debe imprimir un string del estilo "El cocinero X ha cocinado Y perdiendo Z de energía", donde X es el nombre, Y es el plato que cocinó (usando su método str), y Z es la energía perdida. El formato exacto del string da lo mismo, lo importante es que se reflejen esos 3 datos.

  Finalmente, se debe retornar el plato cocinado (es decir, la instancia creada).

2.4. Una clase `Cliente` que **herede** de `Persona`. Debe contener los siguientes atributos adicionales:

- `platos_preferidos`: Lista con los nombres de los platos preferidos del cliente. Se recibe como parámetro de inicialización y pueden ser aleatoriamente entre 1 a 5 platos favoritos.

También debe tener (como mínimo) el siguiente método:

- `recibir_pedido(pedido, demora)`: Recibe el argumento `pedido`, que es una lista de objetos de la clase `Bebestible` o `Comestible`. Recibe el argumento `demora`, que es un int que indica cuánto se demoró la entrega de los platos.

  Primero se define una calificación que comienza en 10. Si la cantidad de platos en el pedido es menor a la cantidad de `platos_preferidos` del cliente o si la demora es mayor o igual a 20, la calificación es dividida a la mitad.

  Luego, por cada plato, el cliente cambiará su calificación dependiendo de la calidad del plato. Si la calidad del plato es mayor o igual a 11, a la calificación se le suma 1.5. Si la calidad es menor o igual a 8, la calificación disminuye en 3. En cualquier otro caso la calificación se mantiene.

El método debe imprimir un string del estilo "El cliente X ha recibido su pedido y le puso la calificación Y", donde X es el nombre, e Y es la calificación puesta. El formato exacto del string da lo mismo, lo importante es que se reflejen esos 3 datos.

Finalmente retorna la calificación que el cliente le ha asignado al restaurante.

3. En el **archivo restaurante.py** debe definir una clase `Restaurante` que contenga los siguientes atributos:

   - `nombre`: Nombre del restaurante de tipo string. Se recibe como parámetro de inicialización.
   - `platos`: Diccionario con todos los platos del restaurante. Cada llave es el nombre de un plato y su valor es una lista con el nombre del plato y su tipo. Se recibe como parámetro de inicialización.
   - `cocineros`: Lista con los cocineros del restaurante. Son instancias de la clase `Cocinero`. Se recibe como parámetro de inicialización.
   - `repartidores`: Lista con los repartidores del restaurante. Son instancias de la clase `Repartidor`. Se recibe como parámetro de inicialización.
   - `calificacion`: La calificación del restaurante. Se inicializa en 0.

   También debe tener (como mínimo) el siguiente método:

   - `recibir_pedidos(clientes)`: Recibe el argumento `clientes`, que es una lista con objetos de la clase `Cliente`. Modifica el valor del atributo `calificacion` del restaurante según el siguiente proceso.

     1. Por cada `cliente` de la lista `clientes` se obtienen todos sus platos favoritos.
     2. Luego por cada `plato`, este se cocina haciendo uso de algún `cocinero` y su método `cocinar(plato)` del restaurante para prepararlo y se agrega a una lista llamada `pedido`. (Solo se puede utilizar un cocinero con energía positiva para cocinar. Si no quedan cocineros que cumplan esta condición no se cocina el plato.)
     3. Una vez que los platos se han cocinado, se calcula cuánto será la demora del tiempo del `pedido` (la lista con todos los platos cocinados) con algún `repartidor` y su método

`repartir(pedido)`.(Solo se puede utilizar un repartidor con energía positiva para repartir. Si no quedan repartidores que cumplan esta condición, se le entrega al cliente en `recibir_pedido(pedido, demora)`, un `pedido` vacío (lista vacía) y una `demora` igual a 0.)

4. Se entrega el `pedido` y la `demora` al cliente con su método `recibir_pedido(pedido, demora)`.
5. Se actualiza la `calificacion` del restaurante con la calificación que el cliente retorna en el paso anterior.
6. Una vez se hayan terminado de entregar todos los pedidos, se divide la `calificacion` final por la cantidad de clientes atendidos (largo de la lista `clientes`).

4. En el **archivo main.py** completar las siguientes funciones:

- `crear_repartidores()`: No recibe parámetros. Crea 2 repartidores de la clase `Repartidor` (con sus parámetros correspondientes según lo descrito en el punto 2.2) y los agrega a una lista. Finalmente retorna la lista.
- `crear_cocineros()`: No recibe parámetros. Crea 5 cocineros de la clase `Cocinero` (con sus parámetros correspondientes según lo descrito en el punto 2.3) y los agrega a una lista. Finalmente retorna la lista.
- `crear_clientes()`: No recibe parámetros. Crea 5 clientes de la clase `Cliente` (con sus parámetros correspondientes según lo descrito en el punto 2.4) y los agrega a una lista. Finalmente retorna la lista.

  (*) Estos 3 métodos deben imprimir los nombres de los repartidores/cocineros/clientes creados. El formato exacto da lo mismo, siempre y cuando se entienda la información.

- `crear_restaurante()`: No recibe parámetros. crea una variable `cocineros` y lo iguala a la función `crear_cocineros()`. Luego crea una variable `repartidores` y lo iguala a la función `crear_repartidores()`. Además, crea un restaurante de la clase `Restaurante` (con sus parámetros correspondientes según lo descrito en el punto 3). Finalmente retorna la instancia del restaurante creado. **(Hint: el nombre del restaurante lo eligen ustedes)**

Cosas a **considerar en general**:

I. Deben manejar todos los imports de módulos externos que se necesiten en el archivo **`main.py`**, en los otros archivos ya vienen listos.

II. Para elegir aleatoriamente los platos preferidos del cliente, se tienen que usar los platos que se encuentran en la **variable** `INFO_PLATOS` en el archivo `main.py`
III. Para ingresar aleatoriamente los nombres de los cocineros, repartidores y clientes, tienen que elegir entre los nombres que se encuentran en la **variable** `NOMBRES` en el archivo `main.py`
IV. Para ingresar los platos al restaurante, tienen que usar todos los platos que se encuentran en la **variable** `INFO_PLATOS` en el archivo `main.py`
V. Es muy importante que en cada clase, aquellos atributos que se piden como parámetros de inicialización, vayan en el orden en que se mencionan. **(Hint: se pueden apoyar en los códigos de prueba para ver cuál es el orden correcto de cada clase)**

Una vez terminado su programa, **debe ejecutar el archivo main.py para probar su solución**.