Instrucciones

Construir un programa utilizando programación orientada a objetos que modele el funcionamiento de un restaurante.

En el restaurante trabajan cocineros, los cuales preparan platos (bebestibles y comestibles) con diferentes niveles de dificultad de preparación. Además, 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 los pedidos según un tiempo determinado. Tanto los cocineros como los repartidores poseen una cantidad de energía que disminuye con cada acción que realizan, 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.

Se deberá entregar como resultado un valor que corresponde a una calificación para el restaurante en base al servicio ofrecido y la evaluación de los clientes.

Archivos entregados

Para este trabajo, se entregará una carpeta *proyecto.zip* que contiene cuatro archivos:

- *main.py*: Corresponde al archivo principal que permitirá probar el funcionamiento del programa.

- *restaurante.py*: Contiene la definición inicial de la clase Restaurante, la cual deberá ser completada para modelar la simulación.

- *platos.py*: Contiene la definición inicial de la clase Plato, Bebestible y Comestible, las cuales deberán ser completadas para modelar la simulación.
- *personas.py*: Contiene la definición inicial de la clase Persona, Repartidor, Cocinero y Cliente, las cuales deberán ser completadas para modelar la simulación.

Trabajo a realizar

Para completar la simulación del restaurante se deben realizar las siguientes

tareas: **Parte 1: Modelación de platos.**

En el archivo *platos.py* se deben definir las siguientes clases:

1.1. Clase Plato:

Esta clase debe poseer los siguientes atributos:

- nombre: Es un *str* que corresponde al nombre del plato. Este se recibe como parámetro de inicialización.

- calidad: Es un *int* que representa la calidad del plato. Se inicializa en 0.

1.2. Clase Bebestible:

Esta clase debe heredar de la clase Plato. Además, esta clase debe poseer los siguientes atributos adicionales:

- tamano: Es un *str* que representa el tamaño del bebestible. Debe inicializarse eligiendo aleatoriamente entre los *strings* "Pequeño", "Mediano" y "Grande".

- dificultad: Es un *int* que representa la dificultad del plato. Si el tamaño es "Pequeño" debe inicializarse en 3, si el tamaño es "Mediano" debe inicializarse en 6 y si el tamaño es "Grande" debe inicializarse en 9.

- calidad: Es un *int* que representa la calidad del bebestible. Debe inicializarse en un número entero aleatorio entre 3 y 8.

1.3. Clase Comestible:

Esta clase debe heredar de la clase Plato. Además, esta clase debe poseer los siguientes atributos adicionales:

- dificultad: Es un *int* que representa la dificultad del plato. Debe inicializarse en un número entero aleatorio entre 1 y 10.
- calidad: Es un *int* que representa la calidad de un comestible. Debe inicializarse en un número entero aleatorio entre 5 y 10.

Si ejecutas directamente el archivo *platos.py*, podrás probar si hay errores al momento de definir las clases anteriores.

Parte 2: Modelación de personas.

En el archivo *personas.py* se deben definir las siguientes

clases: **2.1. Clase Persona:**

Esta clase debe poseer los siguientes atributos:

- nombre: Es un *str* que corresponde al nombre de la persona. Este se recibe como parámetro de inicialización.

2.2. Clase Repartidor:

Esta clase debe heredar de la clase Persona. Además, esta clase debe poseer los siguientes atributos adicionales:

- tiempo_entrega: Es un *int* que corresponde al tiempo base en segundos que se demora en entregar un pedido. Se recibe como parámetro de inicialización (será un número entre 20 y 30).

- energia: Es un *int* que representa la energía del repartidor. Este valor se inicializa como un número aleatorio entre 75 y 100.

También, debe tener (como mínimo) el siguiente método:

- repartir(pedido): Recibe como argumento una **lista** de platos. Este método debe hacer dos cosas. Primero, debe disminuir la energía 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 a 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.
  Finalmente, el método debe retornar el tiempo de demora del pedido.

2.3. Clase Cocinero:

Esta clase debe heredar de la clase Persona. Además, esta clase debe poseer los siguientes atributos adicionales:

- habilidad: Es un *int* que representa la habilidad del cocinero. Se recibe como parámetro de inicialización (será un número entre 1 y 10).

- energia: Es un *int* que representa la energía de la persona. Su valor 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 y en caso contrario es 1.5. Finalmente, se debe retornar el plato cocinado (es decir, la instancia creada).

2.4. Clase Cliente:

Esta clase debe heredar de la clase Persona. Además, esta clase debe poseer los siguientes atributos adicionales:

- platos_preferidos: Es una lista que contiene los nombres de los platos preferidos del cliente. Se recibe como parámetro de inicialización (pueden ser entre 1 y 5 platos).

También debe tener (como mínimo) el siguiente método:

- recibir_pedido(pedido, demora): Recibe como argumento el pedido, que es una lista de objetos de la clase Bebestible o Comestible, y la demora, que es un *int* que indica cuánto se demoró la entrega de los platos. Primero se debe definir 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, no pudiendo este valor descender de 0. En cualquier otro caso la calificación se mantiene.
Finalmente se debe retornar la calificación que el cliente le ha asignado al restaurante.

Si ejecutas directamente el archivo *personas.py*, podrás probar si hay errores al momento de definir las clases anteriores.

Parte 3: Modelación del restaurante.

En el archivo *restaurante.py* se debe definir la clase Restaurante, que debe contener los siguientes atributos:

- nombre: Es un *str* que corresponde al nombre del restaurante. Se recibe como parámetro de inicialización.

- platos: Es un diccionario que posee todos los platos del restaurante, donde 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: Es una lista con los cocineros del restaurante, los cuales son instancias de la clase Cocinero. Se recibe como parámetro de inicialización.

- repartidores: Es una lista con los repartidores del restaurante, los cuales son instancias de la clase Repartidor. Se recibe como parámetro de inicialización.

- calificacion: Es un *int* que corresponde a 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. Debe modificar el valor del atributo calificación del restaurante según el siguiente proceso:

    1. Por cada cliente de la lista clientes se obtienen todos sus platos

preferidos.

2. Luego, cada plato se cocina haciendo uso de algún cocinero y su método cocinar(plato) para prepararlo y se agrega a una lista llamada pedido (sólo 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) (sólo 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 calificación del restaurante sumándole la calificación que el cliente retorna en el paso anterior.

6. Una vez se hayan terminado de entregar todos los pedidos, se divide la calificación final por la cantidad de clientes atendidos (largo de la lista clientes).

Si ejecutas directamente el archivo *restaurante.py*, podrás probar si hay errores al momento de definir la clase anterior.

Parte 4: Completar código principal.

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

- crear_repartidores(): No recibe parámetros. Crea 2 repartidores de la clase Repartidor (con sus parámetros correspondientes) 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) 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) y los agrega a una lista. Finalmente

retorna la lista.

• crear_restaurante(): No recibe parámetros. Crea una variable cocineros, cuyo valor es el retorno de la función crear_cocineros(). Luego crea una variable repartidores, cuyo valor es el retorno de la función crear_repartidores(). Además, crea un restaurante de la clase Restaurante (con sus parámetros correspondientes). Finalmente retorna la instancia del restaurante creado. (El nombre del restaurante lo eliges tú)

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

• Para elegir aleatoriamente los platos preferidos del cliente, se deben usar los platos que se encuentran en la variable INFO_PLATOS en el archivo *main.py*

• Para ingresar aleatoriamente los nombres de los cocineros, repartidores y clientes, se debe elegir entre los nombres que se encuentran en la variable NOMBRES en el archivo *main.py*

• Para ingresar los platos al restaurante, debes usar todos los platos que se encuentran en la variable INFO_PLATOS en el archivo *main.py*

• Es muy importante que, en cada clase, aquellos atributos que se piden como parámetros de inicialización, vayan en el orden en que se mencionan. (Pista: se pueden apoyar en los códigos de prueba para ver cuál es el orden correcto de cada clase)

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