Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2233 — Programación Avanzada 2019-1

21 de marzo de 2019
Actividad Formativa

Actividad 02

OOP, Herencia, Clases Abstractas y Polimorfismo

Introducción

Cuenta la leyenda que un misterioso profesor llamado Kebil Nawas dejó el cuerpo docente de Programación Avanzada, y con esto ha abandonado la famosa fundación sin fines de lucro conocida como **PyKitchen**. El cuerpo docente te ha pedido a ti, alumno experimentado de `IIC2233`, realizar un programa que maneje de manera automática la organización de la cocina, ya que sin esto las **pizzatones** de ayudantes no serán posibles. A continuación se presentan las clases necesarias para modelar este programa.

Figura 1: Logo de PyKitchen Foundation

Para crear el programa recibirás cuatro archivos: `food.py`, `people.py`, `pykitchen.py` y `main.py`. En `food.py` encontrarás lo siguiente:

- `class Plato`: Contiene una pizza y un bebestible.
  - `class Pizza`: Cada pizza tiene una calidad inicial numérica, la cual puede variar dependiendo de los ingredientes y tiempo de preparación (más detalles en especificaciones). Se prepara con una base de salsa de tomate y queso más **3 ingredientes extra**, que pueden ser: pepperoni, piña, cebolla, tomate, jamón o pollo.
  - `class Bebestible`: Cada bebestible tiene una calidad inicial numérica, la cual puede variar dependiendo de si es bebida o agua (más detalles en especificaciones).

En `people.py` encontrarás las siguientes clases:

- `class Personalidad`: Contiene un método `reaccionar(plato)` para responder de acuerdo a la calidad del plato recibido.
- `class Persona`: Contiene el nombre de la persona.

En `pykitchen.py` encontrarás:

- `class PyKitchen`: Recibe una lista de chefs y de ayudantes. El método `atender()` simula 3 días de servicio. En cada uno de estos días los chefs prepararán 2 platos cada uno, que luego serán probados por los ayudantes. Al final de cada día se deben haber preparado 2 platos por chef y cada ayudante debe haber probado cada uno de los platos.

Finalmente, en `main.py` encontrarás las instancias de **PyKitchen**.

Según este modelo, deberás programar las siguientes especificaciones:

- Los chefs y ayudantes son personas.
- El chef prepara platos compuestos por una pizza y un bebestible. Cuando cocina una pizza, la base ya viene puesta y solo escoge aleatoriamente los 3 ingredientes, los cuales pueden repetirse (ver sección **Notas**). Cuando escoge el bebestible, primero elige si será agua o una gaseosa (con igual probabilidad). En caso de que sea una gaseosa, además deberá elegir entre las marcas Coca-Cola, Pepsi o Sprite (también con igual probabilidad). La clase `Chef` debe tener un método `cocinar()` que retorne un plato con pizza y bebestible.
- Cada ayudante tiene un atributo especial: su personalidad. Puede ser empatico o exigente.
- Cada ayudante tendrá un método `comer(plato)` que recibe un plato, y en el que se debe promediar la calidad de la pizza y el bebestible para obtener la calidad final del plato. Si la calidad del plato es mayor o igual a 50, entonces el ayudante se pondrá feliz, en caso contrario estará molesto. Los métodos `feliz()` y `molesto()` serán llamados dependiendo de la calidad del plato y deberán imprimir lo siguiente:
  - `feliz()`
    - Ayudante empático: `'Cosa ma wena, voy a poner puros 7'`
    - Ayudante exigente:

      `'Esta merienda está muy buena, los alumnos se merecen el 4'`
  - `molesto()`
    - Ayudante empático: `'No me gustó, pondré puros 5'`
    - Ayudante exigente: `'Que bruto, póngale 0'`
- Para verificar si el ayudante queda feliz o molesto con su plato, procure que su implementación mantenga que las clases esten encapsuladas, es decir, que no se puede revisar la personalidad del ayudante desde fuera de la clase `Personalidad`. Es trabajo de esta clase revisar la calidad del plato y llamar luego a los métodos necesarios.
- Todos los bebestibles tienen una calidad inicial aleatoria que varía uniformemente entre 50 y 150 (ver sección **Notas** para más información sobre como aplicar una distribución uniforme). Esta varía de la siguiente forma: si es gaseosa disminuye en 30, pero si es agua aumenta en 30.
- Todas las pizzas tienen una calidad inicial aleatoria que varía uniformemente entre 50-200 (ver sección **Notas** para más información sobre como aplicar una distribución uniforme). Esta puede aumentar o disminuir dependiendo de los ingredientes y el tiempo de preparación:
  - Probando ingredientes y después de un estudio, los chefs descubrieron que si una pizza contiene piña su calidad disminuye en 50, pero si tiene pepperoni aumenta en 50. Esto depende únicamente de si la pizza contiene piña o pepperoni. Si la pizza tiene doble piña, disminuirá su calidad

en 50, no en 100. Lo mismo aplica al pepperoni. El método encargado de aumentar o disminuir la calidad de la pizza dependiendo de los ingredientes debe llamarse `revisar_ingredientes()`.

- El tiempo de preparación es una variable aleatoria uniforme entre 20 y 100. Si es mayor o igual a 30 minutos, la calidad inicial disminuye en 30. En caso contrario, permanece igual. El método encargado de modificar la calidad dependiendo del tiempo de preparación debe llamarse `revisar_tiempo()`.

- Al crear un plato e instanciar la pizza y la bebida correspondiente, debe chequear inmediatamente los ingredientes y el tiempo de preparación, de tal forma de tener la calidad final del plato al momento de crearlo.

Notas

- El método `randint(a,b)` del módulo `random` retorna un número al azar de un modelo de distribución uniforme entre $[a, b]$.
- El método `choice(lista_elems)` del módulo `random` retorna un elemento al azar de una lista `lista_elems`.
- **Usuarios de pycharm**: En general, para las actividades recomendamos abrir el proyecto en la carpeta de la actividad, no la carpeta de su repositorio ni la carpeta del syllabus, para evitar problemas con `import`s o errores al correr el código.

Requerimientos

- (2.00 pts) Clases `Pizza`, `Gaseosa` y `Agua`
  - (1.00 pts) Completar la clase `Pizza` con los atributos y métodos necesarios.
  - (1.00 pts) Crear las clases `Agua` y `Gaseosa` con los atributos y métodos necesarios.
- (2.00 pts) Clases `Chef` y `Ayudante`
  - (1.50 pts) Crear la clase `Chef` con los atributos y métodos necesarios.
  - (0.50 pts) Crear la clase `Ayudante` con los atributos y métodos necesarios.
- (2.00 pts) Clases `Personalidad`, `Empatico` y `Exigente`
  - (1.00 pts) Definir el método `reaccionar(plato)` en la clase `Personalidad` para llamar a los métodos `feliz()` o `molesto()` dependiendo de la calidad del plato.
  - (0.50 pts) Crear la clase `Empatico` con los atributos y métodos necesarios.
  - (0.50 pts) Crear la clase `Exigente` con los atributos y métodos necesarios.

Entrega

- **Lugar:** En su repositorio privado de GitHub, en la **carpeta** `Actividades/AC02/`
- **Hora del *push*:** 16:30

Auto-evaluación

Como esta corresponde a una actividad formativa, te extendemos la instancia de responder, después de terminada la actividad, una auto-evaluación de tu desempeño. Esta se habilitará a las **16:50 de jueves 21 de marzo** y tendrás plazo para responderla hasta las **23:59 del día siguiente, viernes 22 de marzo**. Puedes acceder al formulario mediante el siguiente enlace:

`https://goo.gl/forms/2cXo9m4jTkgOmNoU2`

El asistir, realizar la actividad y responder la auto-evaluación otorgará como bonificación al alumno 2 décimas para sumar en su peor actividad sumativa del semestre.