PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN
IIC2233 - PROGRAMACIÓN AVANZADA

Actividad 07

1^o semestre 2018

3 de mayo

Metaclases

Introducción

¡El malvado Dr. Mavrakis ataca! Estabas yendo a tu saludable, y bien merecida, parranda nocturna cuando, después de una dura actividad de ***Programación Avanzada***, oyes algo. Es el motor de tu auto. Se averió y debes ir al taller más cercano. El problema es que no hay nadie allí, y el programa para procesar los autos está incompleto. Por suerte, ¡cuentas con tus habilidades de programador para arreglarlo!

Instrucciones

Debes terminar el programa de ingreso de autos al taller usando metaclases, sin hacer cambios a lo que ya está hecho. Este programa tiene **cuatro** archivos, donde solo deberás hacer cambios en `metaclases.py`, y *ejecutar* `main.py` para verificar la correctitud de tus cambios.

Dentro del módulo `funciones.py` se encuentran varias funciones **ya implementadas**, que tendrás que utilizar en tus metaclases:

1. `reparar(trabajador, auto)`
   Esta función se encarga de que un trabajador repare las piezas de un auto.

2. `revisar_estado(trabajador, auto)`
   Esta función se encarga de que un trabajador revise el estado de un auto.

3. `crear_piezas()`
   Esta función retorna las piezas que debe llevar un auto, creadas a partir del archivo `nombre_piezas.txt`.

4. `definir_estado_piezas(auto)`
   Esta función se encarga de darle un estado a las piezas de la instancia de `Auto`. Ten en mente que ninguna pieza quedará en 100 %, porque se han desgastado.

Dentro de `metaclases.py` ya se importa el módulo `funciones.py`, por lo que sólo deberás usar esas funciones donde sea necesario. Las metaclases que tendrás que implementar son las siguientes:

- `MetaTrabajador:` Como se trata de un taller pequeño atendido solamente por su dueño, esta metaclase debe permitir **sólo una instancia** de la clase. En caso de que la clase se intente instanciar más veces, debes retornar el primer objeto creado. Adicionalmente, se debe **eliminar** el método `revizar_ztado(auto)` de la clase, y **agregar** los métodos `revisar_estado(trabajador, auto)` y `reparar(trabajador, auto)`.

- `MetaAuto`: Como **atributo de la clase**, se debe agregar `piezas`, cuyo valor es lo que retorna la función `crear_piezas()`. Además, debes **agregar** el método `definir_estado_piezas`. Luego, debes asegurarte de llamar a `definir_estado_piezas` cuando se cree una instancia. Para no sobrecargar al solitario mecánico, **sólo se pueden instanciar 3 autos**. En caso de que se intente instanciar más autos de lo permitido, se debe retornar `None`.

Requerimientos

- (3,0 pts) `MetaTrabajador`
  - (0,60 pts) Eliminar `revizar_ztado`
  - (0,80 pts) Agregar `revisar_estado`
  - (0,80 pts) Agregar `reparar`
  - (0,80 pts) Correcta instanciación
- (3,0 pts) `MetaAuto`
  - (0,80 pts) Agregar piezas como atributo de la clase, con el resultado de `crear_piezas`
  - (0,80 pts) Agregar `definir_estado_piezas`
  - (0,60 pts) Llamar la función `definir_estado_piezas` de la instancia creada
  - (0,80 pts) Correcta instanciación

Notas

- Tómate un tiempo para pensar donde tiene más sentido definir cada requerimiento.
- Los métodos a utilizar se encuentran en el módulo `funciones.py`.

Entrega

- **Lugar:** En su repositorio de GitHub en la **carpeta** `Actividades/AC07/`
- **Hora:** 16:40
- Si está trabajando en pareja, **basta con que un miembro suba la actividad**. Si se suben actividades distintas, se corregirá una de las dos al azar.