PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC2233 - Programación Avanzada
1º semestre 2015

Actividad 8

Decorators

Inyección y modificación de código

Se tiene un local de comida que vende hamburguesas. El dueño del local quiere que cada vez que cocine un producto (cree un objeto de la clase correspondiente), este le quede guardado en una lista junto con los otros productos para que pueda acceder a ellos mediante `Clase.instancias`. Además, el dueño del local quiere poder comparar los productos que ha creado mediante los operadores $<$,$>$, $=$, $\leq$,$\geq$ según algún atributo del producto que aún no ha decidido.

Por último, se tiene una función calculadora de precio de venta (transforma de precio bruto a neto) que multiplica el precio bruto por 1.19 debido al IVA y luego le suma $100 por costo de transporte. Sin embargo, por los cambios económicos del país, el IVA desde hoy es un 23 % del valor bruto (y no un 19 % como hasta ayer). El dueño del local quiere que su función sea arreglada para que calcule el precio correctamente acorde al nuevo valor del IVA.

To-Do

1. **(2.0 pts)** : Crear el decorador `guardar_instancias` que guarde cada instancia creada de la clase decorada en una lista que pueda ser accedida mediante `Clase.instancias`

2. **(2.0 pts)** : Crear el decorador `comparar_por` que reciba como parámetro el nombre de un atributo y que permita comparar las instancias de una clase mediante aquel atributo.[1]
   La comparación debe funcionar de forma que: `hamburguesa1` $>$ `hamburguesa2` retorna `True` si es que el atributo especificado al decorador de la clase es mayor en `hamburguesa1` que en `hamburguesa2`.

3. **(2.0 pts)** : Crear el decorador `cambiar_precio` que permita calcular los precios brutos correctamente al ser aplicado a la función calculadora de precios. El precio neto debido al cambio del IVA debería ser: $precio\_neto = precio\_bruto \times 1{,}23 + 100$, pero actualmente es: $precio\_neto = precio\_bruto \times 1{,}19 + 100$. El objetivo es lograr que retorne el precio correcto *sin* modificar la función actual.

[1] Recuerde que las comparaciones llaman a los métodos: `__lt__` , `__le__` , `__eq__` , `__ge__` , `__gt__`

Tips y Ejemplos

- **Acceder y modificar atributos**
  Si quiere, puede usar los comandos `getattr` y `setattr` para trabajar con atributos de un objeto.

```
# Acceder
valor_viejo = getattr(objeto, 'nombre_atributo')
# Modificar
setattr(objeto, 'nombre_atributo', valor_nuevo)
```

- **Decorar una clase**
  Este es un ejemplo de un decorador que modifica el método de una clase para que imprima en consola cuando es llamado.

```
def avisar_llamado(nombre_method):

    def decorador(cls):
        method = getattr(cls, nombre_method)

        def new_method(*args, **kwargs):
            print('Llamando el metodo!')
            return method(*args, **kwargs)

        setattr(cls, nombre_method, new_method)
        return cls

    return decorador

# Se aplica a una clase de prueba

@avisar_llamado('caminar')
class Test:
    def caminar(self):
        return 'estoy caminando'
```