PROGRAMACIÒN ORIENTADA A OBJETOS

TALLER No. 3 PYTHON	PROGRAMACIÓN ORIENTADA A OBJETOS

Profesor: Jaime Alberto Guzmán Luna

Contenido del taller:

1) Encapsulamiento
2) Constructores
3) self

El método **isinstance**, sirve para saber si una instancia pertenece a una clase, este método recibe dos argumentos la instancia y la clase, retorna True o False si pertenece o no a la clase.

pass es una instrucción importante para Python ya que permite dejar "pasar" la ejecución y no generar error en métodos o clases sin cuerpo.

Ejercicio 1

a. La siguiente es la estructura del paquete taller:

```
taller/
+-- src/
    +-- main.py
    +-- almacenamientos/
        +-- almacenamiento.py
        +-- ram.py
    +-- dispositivos/
        +--___init__.py
        +-- almacenamientoExterno.py
        +-- celular.py
        +-- computador.py
```

¿Es correcta esta estructura? ¿Si le falta algo, que sería?

b. Analice el siguiente ejercicio y cambie los lugares con **, con el valor que debería ir, para que funcione, cumpla con el encapsulamiento y tenga un uso correcto de los módulos.

Nota: la estructura de los archivos es la especificada en el punto **a**

Resultado esperado al ejecutar main.py

```
Celular Realme 8 pro
Cantidad de celulares: 1
Cantidad de almacenamientos externos: 1
Cantidad de computadores: 1
['_Computador_cantidad', '_class_',
'__delattr__', '__dict__', '__dir__', '__doc__',
```

```
'__eq__', '__format__', '__ge__',
'__getattribute__', '__gt__', '__hash__',
'_init_', '_init_subclass_', '_le_',
'__lt__', '__module__', '__ne__', '__new__',
'_reduce_', '_reduce_ex_', '_repr_',
'__setattr__', '__sizeof__',
'_str_', '_subclasshook_', '_weakref_',
'getConexion', 'getMarca', 'getModelo',
'setConexion', 'setMarca']
```

Archivo main.py

```
1  from dispositivos.** import Celular
2  from **.almacenamientoExterno import AlmacenamientoExterno
3  from dispositivos.computador import **
4  from almacenamientos.ram import Ram
5  from almacenamientos.almacenamiento import Almacenamiento
6
7  if ** == "__main__":
8
9      cel = **("Alejandro", "Realme 8 pro", **(128, "SSD"))
10     alm = AlmacenamientoExterno("Curso POO")
11     pc = Computador("Lenovo", "s330", Almacenamiento(
12         2000, "HDD"), Ram(8, "DDR4"))
13
14     **.setConexion(cel)
15
16     print(pc.getConexion())
17
18     print("Cantidad de celulares: " + **)
19     print("Cantidad de almacenamientos externos: " +
20           **)
21     print("Cantidad de computadores: " + **)
22
23     print(**)
```

Archivo almacenamiento.py

```
1  class **:
2      def_init_(self, tamano, **):
3          self._TAMANO = tamano
4          self.tipo = tipo
5
6      def **(self):
7          return self._TAMANO
8
9      def setTamano(self, tamano):
10         **
```

Archivo ram.py

```
1  class Ram:
```

```
    def___init__(self, **, tipoModulo):
        self._TAMANO = **
        self._tipoModulo = tipoModulo

    def setTipoModulo(self, tipoModulo):
        self.** = tipoModulo

    def getTipoModulo(self):
        return self._tipoModulo

    def getTamano(self):
        return self.**
```

Archivo almacenamientoExterno .py

```
from almacenamientos.** import **


class AlmacenamientoExterno:
    _cantidad = **

    def___init__(self, nombre):
        self._nombre = nombre
        self._almacenamiento = Almacenamiento(1000, "HDD")
        **._cantidad += 1

    def **(**, **):
        self.** = **

    def **(**):
        return self.**

    def setAlmacenamiento(self, almacenamiento):
        self._almacenamiento = almacenamiento

    def getAlmacenamiento(self):
        return self._almacenamiento

    **
    def getCantidad(**):
        return cls._cantidad
```

Archivo celular.py

```
class Celular:
    _cantidad = 0

    def **(self, dueno, modelo, almacenamiento):
        self.dueno = dueno
        self.modelo = modelo
        self.almacenamiento = almacenamiento
```

```
        Celular.** += 1

    **
    def getCantidad():
        return **._cantidad
```

Archivo computador.py

```
from dispositivos.celular import Celular
from dispositivos.almacenamientoExterno import AlmacenamientoExterno


class Computador:
    ** = 0

    def___init__(self, marca, modelo, almacenamiento, ram):
        self._marca = marca
        self._modelo = modelo
        self.almacenamiento = almacenamiento
        self.ram = ram
        self._conectado = **
        Computador._Computador_cantidad += 1

    def setMarca(self, marca):
        self._marca = marca

    def getMarca(self):
        return self._marca

    def getModelo(self):
        return self._modelo

    def setConexion(self, dispositivo):
        self._conectado = dispositivo

    def getConexion(self):
        if(isinstance(self._conectado, Celular)):
            return "Celular " + self._conectado.modelo
        elif(isinstance(self._conectado, AlmacenamientoExterno)):
            return "Almacenamiento " + self._conectado.getNombre
```

Entregue el código corregido

Ejercicio 2 - GitHub

Enlace entrega: https://classroom.github.com/a/_mF1AC3c

PROBLEMA Fábrica de Televisores

Suponga que tiene unos ahorros y desea incursionar en el negocio de los televisores, para esto

usted se olvida de todos los diseños de electricidad componentes, usted solo se encargara del funcionamiento del software del televisor.

Para la creación del software se le proporciona la siguiente guía y requerimientos a seguir:

Primero, todas las clases a desarrollar se deben implementar en el módulo televisores.

Segundo. Se le quiere dar una marca a los televisores. Para esto se definirán las marcas que tendrán estos televisores. Para hacer esto posible se crea una clase Marca que tendrá:

- Un atributo *nombre* (String)
- Un constructor que recibirá el nombre que le daremos al objeto Marca
- Por último, incluir los métodos *get* y *set* del atributo.

Tercero. Definir la clase TV la cual se encargará de modelar el principal funcionamiento y fin de este software. Esta clase tiene inicialmente las siguientes características:

- Los atributos: marca (Marca), canal(int), precio(int), estado(boolean), volumen(int) y control (Control).
- Tendrá un constructor que tiene como parámetros la marca y el estado (si el televisor este encendido o no)
- Por defecto el valor de los atributos canal, volumen y precio serán: 1, 1 y 500 respectivamente.
- Tendrá los métodos set y get para los atributos marca, control, precio, volumen y canal.
- Como somos una fabrica y queremos llevar las cuentas de la fabricación de televisores, implementar el conteo de televisores creados (objetos creados de la clase) para lo cual se debe implementar el atributo numTV (de tipo clase y no de instancia) y el método correspondiente para acceder a este valor.
- Definir los métodos para cambiar el estado de encendido a apagado llamados turnOn y turnOff, que respectivamente se encargaran de encender y apagar el televisor.
- Implementar el método getEstado que retornara el valor del atributo estado.

Otro requerimiento en nuestros televisores es que implementemos el cambio de canal y el cambio de volumen, para esto definimos los métodos canalUp y canalDown, que se encargaran de cambiar el canal aumentándolo o disminuyéndolo, y los métodos volumenUp, y volumenDown que se encargan de cambiar el volumen aumentándolo o disminuyéndolo.

Una de las dos limitaciones que se afronta en estos televisores, es que los canales disponibles solo van del canal 1 al canal 120 y para cambiar de canal necesariamente debe estar encendido el televisor, por obvias razones, así que en este diseño que se está construyendo, se debe implementar las condiciones que sean necesarias para representar lo anterior. También ocurre lo mismo con el volumen, esta ira de 0 a 7 y para que este cambie debe estar encendido el televisor. Para ambos casos, cambio de canal o subir volumen, en caso de un valor por encima o debajo de sus valores límites o que no se cumpla que esté encendido, no se debe hacer nada.

Para finalizar la fabricación de los televisores, se debe incluir un control asociado al televisor. Para esto se define una clase llamada Control, con el fin de controlar a distancia el televisor. Este tendrá las siguientes características:

- Tendrá un único atributo que será **tv**, el cual permite el enlace entre el control y el televisor asociado.

- El control podrá hacer uso de los métodos del televisor de manera remota, para lo cual se definirán los métodos: turnOn, turnOff, canalUp, canalDown, volumenUp, volumenDown y setCanal.
- Definir un método llamado enlazar con el cual se conecta el control con el televisor que se quiere manejar remotamente. Este método recibirá como parámetro un televisor, y se encargará de asignar el valor del atributo **tv** y al igual que de solicitar al televisor que fue pasado cómo parámetro, que se asigne a su atributo control el mismo objeto control dueño del método enlazar.
- Tenga en cuenta agregar los métodos get y set para el atributo tv.

Si todo salió bien al final se tendrá el modelado de los mejores televisores que se habrán podido fabricar (o programar).

Nota 1: Como se quiere que la competencia no se roben los datos o accedan a ellos, se debe modelar los atributos lo menos accesible posible. Igualmente, dejar como público, las clases, constructores y métodos.

Nota 2: Use los nombres de las clases, atributos, métodos y parámetros en donde se indique, pero si no se indican, puede usar cualquier otro valor. Esto con el fin de usar una herramienta con la cual podrá observar si su implementación es correcta.

Nota 3: Se proporciona un archivo principal llamado main.py, asegurese que lo allí descrito se ejecute sin errores.

PARA PENSAR…

1. ¿Qué cambios observa respecto a la implementación en Java?