UNIVERSIDAD PRIVADA FRANZ TAMAYO
FACULTAD DE INGENIERÍA
CARRERA DE INGENIERÍA DE SISTEMAS

Laboratorio 02: Programación orientada a objetos

Estudiante:

Univ. Brayan Erwin Honorio Rojas

Docente:

Ing. William Barra

Asignatura:

PROGRAMACION DE SISTEMAS EMBEBIDOS

La Paz - Bolivia

Ejercicios

1. **Métodos especiales en Python**
    a. **Método del constructor (__init__) - Constructor**

```
def __init__(self, nombres, apellidos, edad):
    self.__nombres = nombres
    self.__apellidos = apellidos
    self.__edad = edad
```

    b. **Método __str__**

```
def __str__(self):
    return f'Movie name es {self.title} con una duracion de {self.duration}'
```

    c. **Método __len__**

```
def __len__(self):
    return self.duration
```

    d. **Método __del__**

```
def __del__(self):
    print('Se elimino la movie', self.title)
```

2. **Manejo de encapsulación**

```python
class Persona:
    __nombres = None
    __apellidos = None
    __edad = None

    def __init__(self, nombres, apellidos, edad):
        self.__nombres = nombres
        self.__apellidos = apellidos
        self.__edad = edad

    def __metodo_privado(self):
        print('Metodo privado')

    def __metodo_publico(self):
        print('Metodo publico')

    def __str__(self):
        return f'Persona: {self.__nombres}, apellidos: {self.__apellidos}, edad: {s
```

3. **Verificar que un método sea privado**

```python
print(persona1)
persona1.metodo_publico()
persona1.__metodo_privado()
```

4. **Verificar que un parámetro sea privado.**

```python
persona1.__nombres = 'Pepito'

print(persona1.__nombres)
print(persona1)
```

5. **Métodos SETTERS and GETTERS.**
   a. Crear el método get and seter para el atributo __nombres

```python
    def get_nombres(self):
        return self.__nombres

    def set_nombres(self, nuevo_nombre):
        self.__nombres = nuevo_nombre
```

b.

```python
persona1 = Persona('Brayan', 'Honorio', 32)
persona1.set_nombres('Pepito')

print(persona1.get_nombres())
print(persona1)
```

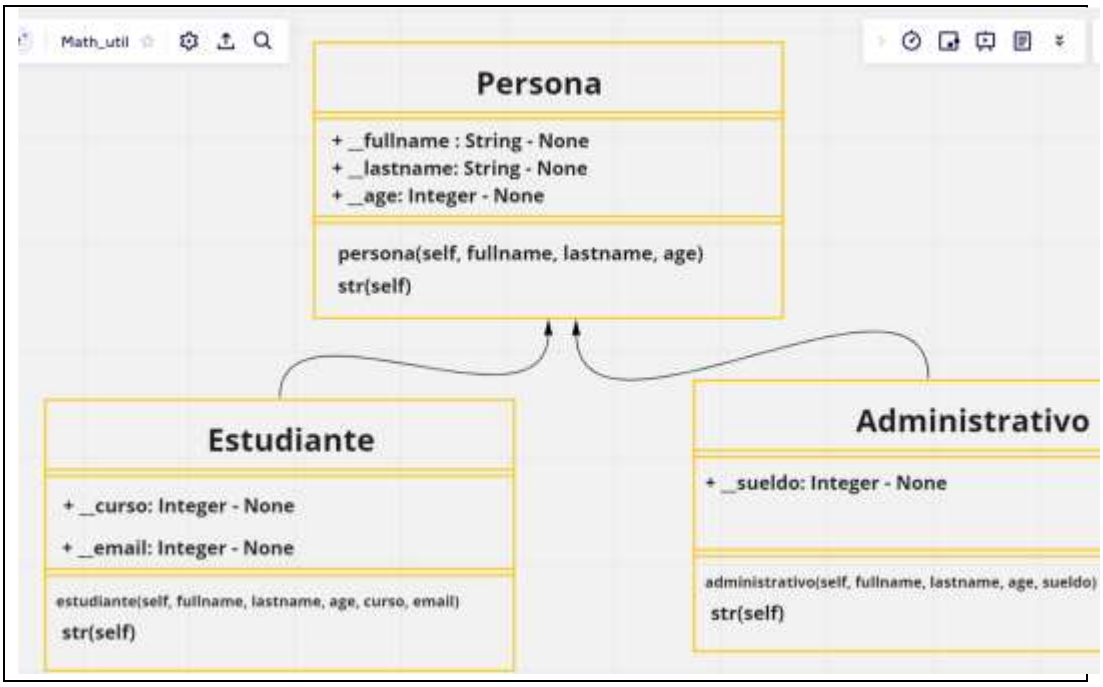
```
Run:  Persona
"C:\Users\BRAYAN HONORIO\PycharmProjects\SEMESTR
Pepito
Persona: Pepito, apellidos: Honorio, edad: 32
```

c. Crear un único método que permite modificar todos los atributos de la persona.

```python
    def update_persona(self, nombres, apellidos, edad):
        self.__nombres = nombres
        self.__apellidos = apellidos
        self.__edad = edad
```

6. Manejo de HERENCIA en Python.

a. Generar las siguientes clases



b. Creación de la clase persona

```python
class Persona:
    __fullname = None
    __lastname = None
    __age = None

    # constructor
    def __init__(self, fullname, lastname, age):
        self.__fullname = fullname
        self.__lastname = lastname
        self.__age = age

    def __str__(self):
        return f'Nombre: {self.__fullname}, \nApellido: {self.__lastname}, \nEdad: {self.__age}'
```

c. Creación de la clase Estudiante

```python
class Estudiante(Persona):
    __curso = None
    __email = None

    def __init__(self, fullname, lastname, age, curso, email):
        Persona.__init__(fullname, lastname, age)
        self.__curso = curso
        self.__email = email

    def __str__(self):
        return Persona.__str__() + f'curso: {self.__curso}, email: {self.__e
```

7. **Manejo de herencia simple y múltiple**

```python
class ClaseA:
    def __init__(self):
        print('Soy de la clase A')

    def metodo_clase_a(self):
        print('Soy un metodo de la clase A')


class ClaseB:

    def __init__(self):
        print('Soy de la clase B')

    def metodo_clase_a(self):
        print('Soy un metodo de la clase B')


class Derivada(ClaseA, ClaseB):
    def __init__(self):
        ClaseA.__init__(self)
        ClaseB.__init__(self)
        print('Soy la clase C-Derivada')

    def metodo_clase_a(self):
        print('Soy un metodo de la clase C-Derivada')
```