Encapsulamiento

Alumno: Puma Huanca Anthony Rusbel
Docente: Ing. Coyla Idme Leonel
Lenguajes de programación II – FINESI
Universidad Nacional del Altiplano
Facultad de Ingeniería Estadística e Informática

Encapsulamiento

Descripción

En POO, el encapsulamiento es un principio que consiste en ocultar los detalles internos del funcionamiento de un objeto y restringir el acceso directo a algunas de sus componentes, exponiendo solo lo necesario a través de interfaces públicas (métodos).

Ejercicio 1

Desarrollar un objeto `Persona`, utilizando atributos y métodos privados.
Clase: `Persona`.
Atributos: nombre, edad.
Acción: `get_edad`, `get_nombre`, `set_edad`, `set_nombre`.
Objeto: `persona = Persona("María", 35)`

```python
class Persona:
    def __init__(self, nombre, edad):
        self.__nombre = nombre  # atributo, con __ se llama atributo privado
        self.__edad = edad      # atributo, con __ se llama atributo privado

    def get_edad(self):
        return self.__edad

    def get_nombre(self):
        return self.__nombre

    def set_edad(self, nueva_edad):
        if nueva_edad > 0:
            self.__edad = nueva_edad
```

```
        else:
            print("Edad no valida")

    def set_nombre(self, nuevo_nombre):
        self.__nombre = nuevo_nombre
        return self.__nombre

persona = Persona("Ana", 30)
print(persona.get_nombre())
print(persona.get_edad())
print(persona.get_nombre(), persona.get_edad())

persona.set_edad(35)
persona.set_nombre("Maria")

print(persona.get_edad())
print(persona.get_nombre())
```

Listing 1: Código Ejercicio 1

Ejecución:

```
Ana
30
Ana 30
35
Maria
```

Ejercicio 2

Hallar el área y el perímetro de un círculo.
Clase: `Circulo`.
Atributo: radio.
Acción: `calcular_area`, `calcular_perimetro`.
Objeto: `circulo = Circulo(3)`

```
import math

class Circulo:
    def __init__(self, radio):
        self.__radio = radio

    def get_radio(self):
        return self.__radio

    def set_radio(self, nueva_radio):
        if nueva_radio > 0:
            self.__radio = nueva_radio
```

```
        else:
            print("Radio no valido")

    def calcular_area(self):
        return math.pi * self.__radio**2

    def calcular_perimetro(self):
        return 2 * math.pi * self.__radio

circulo = Circulo(3)
print(" rea  del circulo:", round(circulo.calcular_area(), 2))
print("Per metro del c rculo:", round(circulo.calcular_perimetro()
    , 2))
```

Listing 2: Código Ejercicio 2

Ejecución:

```
Área del circulo: 28.27
Perímetro del círculo: 18.85
```

Ejercicio 3

Hallar el área y el perímetro de un rectángulo.
Clase: `Rectangulo`.
Atributos: base, altura.
Acción: `calcular_area`, `calcular_perimetro`.
Objeto: `rectangulo = Rectangulo(4, 5)`

```
class Rectangulo:
    def __init__(self, base, altura):
        self.__base = base
        self.__altura = altura

    def get_base(self):
        return self.__base

    def get_altura(self):
        return self.__altura

    def set_base(self, nueva_base):
        if nueva_base > 0:
            self.__base = nueva_base
        else:
            print("Base invalida")

    def set_altura(self, nueva_altura):
        if nueva_altura > 0:
```

```
            self.__altura = nueva_altura
        else:
            print("Altura invalida")

    def calcular_area(self):
        return self.__altura * self.__base

    def calcular_perimetro(self):
        return (self.__altura + self.__base) * 2

rectangulo = Rectangulo(4, 5)
print(" rea  del rect ngulo", rectangulo.calcular_area())
print("Per metro del rect ngulo", rectangulo.calcular_perimetro())
```

Listing 3: Código Ejercicio 3

Ejecución:

```
Área del rectángulo 20
Perímetro del rectángulo 18
```

Ejercicio 4

Crear una calculadora que dado dos números, calcule su suma, resta, producto y división.
Clase: `Calculadora`.
Atributos: num1, num2.
Acción: `suma()`, `resta()`, `producto()`, `cociente()`.
Objeto: `calculadora = Calculadora(2, 5)`

```
class Calculadora:
    def __init__(self, num1, num2):
        self.__num1 = num1
        self.__num2 = num2

    def get_num1(self):
        return self.__num1

    def get_num2(self):
        return self.__num2

    def set_num1(self, nuevo_num1):
        if nuevo_num1 > 0:
            self.__num1 = nuevo_num1
        else:
            print("N mero invalido")

    def set_num2(self, nuevo_num2):
        if nuevo_num2 > 0:
```

```
            self.__num2 = nuevo_num2
        else:
            print("N mero invalido")

    def suma(self):
        return self.__num1 + self.__num2

    def resta(self):
        return self.__num1 - self.__num2

    def producto(self):
        return self.__num1 * self.__num2

    def cociente(self):
        return self.__num1 / self.__num2

def main():
    calculadora = Calculadora(2, 5)
    print("Suma:", calculadora.suma())
    print("Resta:", calculadora.resta())
    print("Producto:", calculadora.producto())
    print("Divisi n:", calculadora.cociente())

if __name__ == "__main__":
    main()
```

Listing 4: Código Ejercicio 4

Ejecución:

```
Suma: 7
Resta: -3
Producto: 10
División: 0.4
```