Herencia en POO

Alumno: Puma Huanca Anthony Rusbel1
Docente: Ing. Coyla Idme Leonel
Lenguajes de Programación II – FINESI
Universidad Nacional del Altiplano
Facultad de Ingeniería Estadística e Informática

Introducción

La **herencia** es un concepto fundamental en la Programación Orientada a Objetos (POO) que permite crear nuevas clases (subclases o clases derivadas) a partir de clases existentes (superclases o clases base). Las subclases heredan atributos y métodos de sus superclases, lo que facilita la reutilización de código y la organización jerárquica de clases.

Ejercicio 1 Herencia simple: Clase Animal

```python
class Animal:  # clase base
    def __init__(self, nombre):
        self.nombre = nombre

    def hacerSonido(self):
        pass

class Perro(Animal):  # clase derivada
    def hacerSonido(self):
        return " Guau !"

class Gato(Animal):
    def hacerSonido(self):
        return " Miauuu !"

perro = Perro("Rex")
print(f"{perro.nombre} dice {perro.hacerSonido()}")

gato = Gato("Charlotte")
print(f"{gato.nombre} dice {gato.hacerSonido()}")
```

Listing 1: Herencia simple con método polimórfico

Ejecución:

```
Rex dice ¡Guau!
Charlotte dice ¡Miauuu!
```

Ejercicio 2 Herencia Simple: Clase Figura Geometrica

```python
import math

class FiguraGeometrica:
    def __init__(self, nombre):
        self.nombre = nombre

    def area(self):
        raise NotImplementedError("Subclases deben implementar este m todo")

    def perimetro(self):
        raise NotImplementedError("Subclases deben implementar este m todo")

class Circulo(FiguraGeometrica):
    def __init__(self, radio):
        super().__init__("Circulo")
        self.radio = radio

    def area(self):
        return math.pi * (self.radio ** 2)

    def perimetro(self):
        return 2 * math.pi * self.radio

class Rectangulo(FiguraGeometrica):
    def __init__(self, base, altura):
        super().__init__("Rect ngulo")
        self.base = base
        self.altura = altura

    def area(self):
        return self.base * self.altura

    def perimetro(self):
        return 2 * (self.base + self.altura)

circulo = Circulo(5)
print(f"Nombre: {circulo.nombre}")
print(f" rea : {circulo.area():.2f}")
print(f"Per metro: {circulo.perimetro():.2f}")

rectangulo = Rectangulo(8, 6)
print(f"Nombre: {rectangulo.nombre}")
print(f" rea : {rectangulo.area()}")
print(f"Per metro: {rectangulo.perimetro()}")
```

Listing 2: Herencia con cálculo de área y perímetro

Ejecución:

```
Nombre: Circulo
Área: 78.54
Perímetro: 31.42
Nombre: Rectángulo
Área: 48
Perímetro: 28
```

Ejercicio 3 Herencia Múltiple: Clase Aves

```python
class Nadador:
    def nadar(self):
        print("Nadando en el agua")

class Volador:
    def volar(self):
        print("Volando por el aire")

class Pato(Nadador, Volador):
    def graznar(self):
        print(" Cuac !")

class Cisne(Nadador, Volador):
    def graznar(self):
        print(" Graa  Graa Graa!")

pato = Pato()
pato.nadar()
pato.volar()
pato.graznar()

cisne = Cisne()
cisne.nadar()
cisne.volar()
cisne.graznar()
```

Listing 3: Herencia múltiple con `Pato` y `Cisne`

Ejecución:

```
Nadando en el agua
Volando por el aire
¡Cuac!
Nadando en el agua
Volando por el aire
¡Graa Graa Graa!
```

Ejercicio 4 Herencia Múltiple: Clase IMC

```python
class Peso:
    def __init__(self, peso_kg):
```

```
        self.peso_kg = peso_kg

class Altura:
    def __init__(self, altura_m):
        self.altura_m = altura_m

class IMC(Peso, Altura):
    def __init__(self, peso_kg, altura_m):
        Peso.__init__(self, peso_kg)
        Altura.__init__(self, altura_m)

    def calcular_imc(self):
        if self.altura_m <= 0:
            raise ValueError("La altura debe ser mayor que 0")
        return self.peso_kg / (self.altura_m ** 2)

    def categoria_imc(self):
        imc = self.calcular_imc()
        if imc < 18.5:
            return "Bajo peso"
        elif imc < 25:
            return "Normal"
        elif imc < 30:
            return "Sobrepeso"
        else:
            return "Obesidad"

    def mostrar_resultado(self):
        imc = self.calcular_imc()
        categoria = self.categoria_imc()
        return f"IMC: {imc:.2f} - Categor a: {categoria}"

def leer_float(mensaje):
    while True:
        try:
            valor = float(input(mensaje))
            if valor <= 0:
                print("Por favor, ingrese un valor positivo")
                continue
            return valor
        except ValueError:
            print("Entrada inv lida, ingrese un n mero v lido"
    )

# peso = leer_float("Ingresa tu peso en kilogramos: ")
# altura = leer_float("Ingresa tu altura en metros: ")
# persona = IMC(peso_kg=peso, altura_m=altura)
# print(persona.mostrar_resultado())
```

Listing 4: Clases `Peso`, `Altura` y `IMC`

Ejecución (ejemplo):

```
Ingresa tu peso en kilogramos: 64.5
Ingresa tu altura en metros: 1.64
IMC: 23.98 - Categoría: Normal
```

Ejercicio 5 Herencia Múltiple: Clase Hipotenusa

```python
class CatetoA:
    def __init__(self, cateto_a):
        self.cateto_a = cateto_a

class CatetoB:
    def __init__(self, cateto_b):
        self.cateto_b = cateto_b

class Hipotenusa(CatetoA, CatetoB):
    def __init__(self, cateto_a, cateto_b):
        CatetoA.__init__(self, cateto_a)
        CatetoB.__init__(self, cateto_b)

    def calcular_hipotenusa(self):
        if self.cateto_a <= 0 or self.cateto_b <= 0:
            raise ValueError("Los catetos deben ser mayores que 0
    ")
        return (self.cateto_a ** 2 + self.cateto_b ** 2) ** 0.5

    def mostrar_resultado(self):
        hipotenusa = self.calcular_hipotenusa()
        return f"La hipotenusa es: {hipotenusa}"

def leer_float(mensaje):
    while True:
        try:
            valor = float(input(mensaje))
            if valor <= 0:
                print("Por favor, ingrese un valor positivo")
                continue
            return valor
        except ValueError:
            print("Entrada inv lida , ingrese un n mero v lido"
    )

# cateto_a = leer_float("Ingrese el cateto a: ")
# cateto_b = leer_float("Ingrese el cateto b: ")
# triangulo = Hipotenusa(cateto_a, cateto_b)
# print(triangulo.mostrar_resultado())
```

Listing 5: Herencia múltiple con catetos e hipotenusa

Ejecución (ejemplo):

```
Ingrese el cateto a: 3
Ingrese el cateto b: 4
```

```
La hipotenusa es: 5.0
```

Ejercicio 6 Herencia Múltiple: Clase PersonasMultirol

```python
class Persona:
    def __init__(self, nombre, edad):
        self.nombre = nombre
        self.edad = edad

    def presentarse(self):
        print(f"Hola, soy {self.nombre} y tengo {self.edad} a os
   ")

class Trabajador:
    def __init__(self, profesion, salario):
        self.profesion = profesion
        self.salario = salario

    def trabajar(self):
        print(f"Estoy trabajando como {self.profesion} y gano ${
   self.salario}")

class Estudiante:
    def __init__(self, carrera, universidad):
        self.carrera = carrera
        self.universidad = universidad

    def estudiar(self):
        print(f"Estudio {self.carrera} en la {self.universidad}")

class PersonaMultirol(Persona, Trabajador, Estudiante):
    def __init__(self, nombre, edad, profesion, salario, carrera,
    universidad):
        Persona.__init__(self, nombre, edad)
        Trabajador.__init__(self, profesion, salario)
        Estudiante.__init__(self, carrera, universidad)

    def mostrar_informacion(self):
        print("==== INFORMACI N DE LA PERSONA ====")
        self.presentarse()
        self.trabajar()
        self.estudiar()

def main():
    persona1 = PersonaMultirol(
        nombre="Juanita",
        edad=25,
        profesion="Desarrollador de software",
        salario=2500,
        carrera="Ingenier a Estad stica e Inform tica",
```

```
        universidad="Universidad Nacional del Altiplano"
    )
    persona1.mostrar_informacion()

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

Listing 6: Persona que es estudiante y trabajador

Ejecución:

```
==== INFORMACIÓN DE LA PERSONA ====
Hola, soy Juanita y tengo 25 años
Estoy trabajando como Desarrollador de software y gano $2500
Estudio Ingeniería Estadística e Informática en la Universidad Nacional del Altiplano
```

Ejercicio 7 Herencia Múltiple: Clase Vehiculo

```
class VehiculoTerrestre:
    def conducir(self):
        print("Conduciendo por la carretera")
    def frenar(self):
        print("El vehiculo terrestre se ha detenido")
class VehiculoAcuatico:
    def navegar(self):
        print("Navegando por el agua")
    def fondear(self):
        print("El vehiculo automatico ha fondeado")

class VehiculoAnfibio(VehiculoTerrestre,VehiculoAcuatico):
    def transformar(self,modo):
        if modo == "tierra":
            print("Cambiando al modo terrestre..")
        elif modo == "agua":
            print("Cambiando al modo acuatico..")
        else:
            print("Modo no reconocido")

def main ():
    anfibio = VehiculoAnfibio()
    anfibio.transformar("tierra")
    anfibio.conducir()
    anfibio.frenar()

    print("\n")

    anfibio.transformar("agua")
    anfibio.navegar()
    anfibio.fondear()

if __name__=="__main__":
```

```
34      main()
```

Listing 7: Vehiculo

Ejecución:

```
Cambiando al modo terrestre..
Conduciendo por la carretera
El vehiculo terrestre se ha detenido

Cambiando al modo acuatico..
Navegando por el agua
El vehiculo automatico ha fondeado
```