Curso: Programación Orientada a Objetos

Tema:

Definición de Herencia

Herencia

¿Qué entiende por herencia?

Herencia

¿Qué es una herencia en programación orientada a objetos?

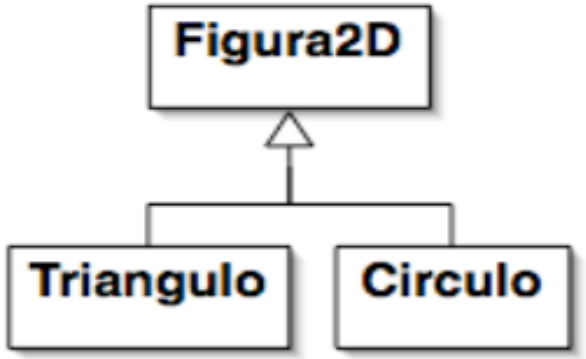
La herencia es un mecanismo de la programación orientada a objetos que sirve para crear clases nuevas a partir de clases preexistentes.

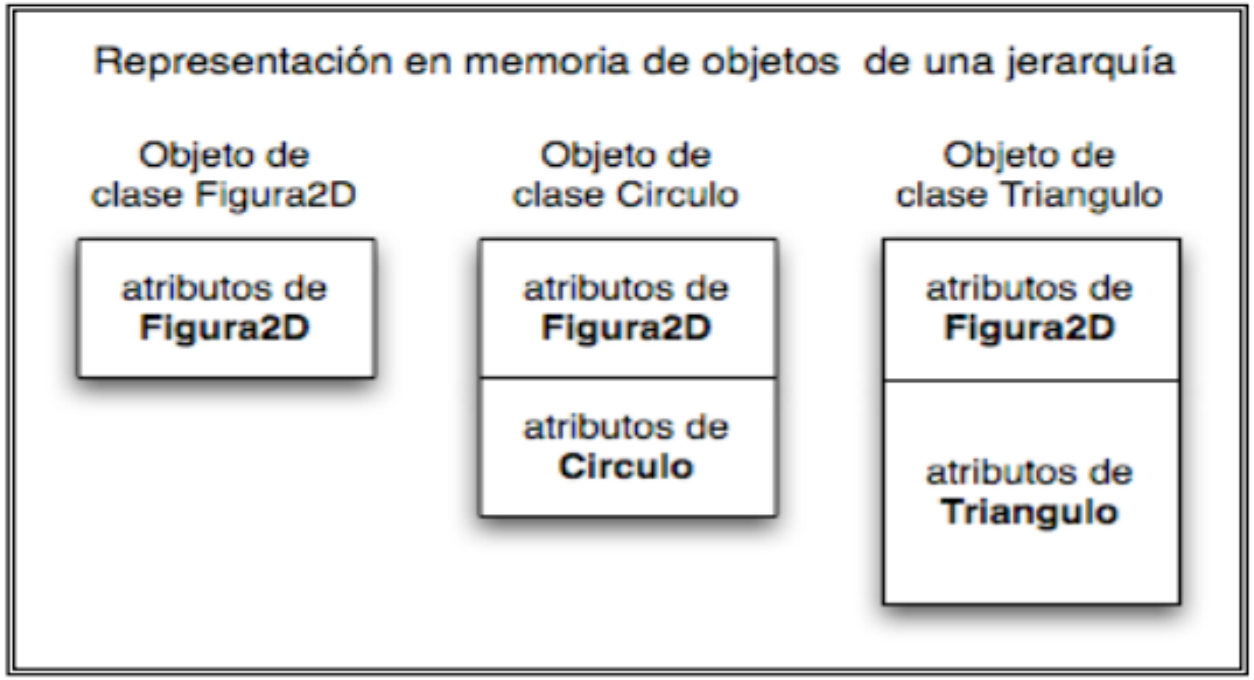
La herencia nos permite definir una clase que hereda todos los métodos y propiedades de otra clase.

Herencia

- Mediante la herencia, las propiedades definidas en una clase base son heredadas por la clase derivada.

- La clase derivada puede añadir propiedades específicas (atributos, métodos o roles)





Herencia

- Gracias a la herencia es posible **especializar** o **extender** la funcionalidad de una clase, derivando de ella nuevas clases.

- La herencia es siempre **transitiva**: una clase puede heredar características de superclases que se encuentran muchos niveles más arriba en la jerarquía de herencia.
  - Ejemplo: si la clase *Perro* es una subclase de la clase *Mamífero*, y la clase *Mamífero* es una subclase de la clase *Animal*, entonces el *Perro* heredará atributos tanto de *Mamífero* como de *Animal*.

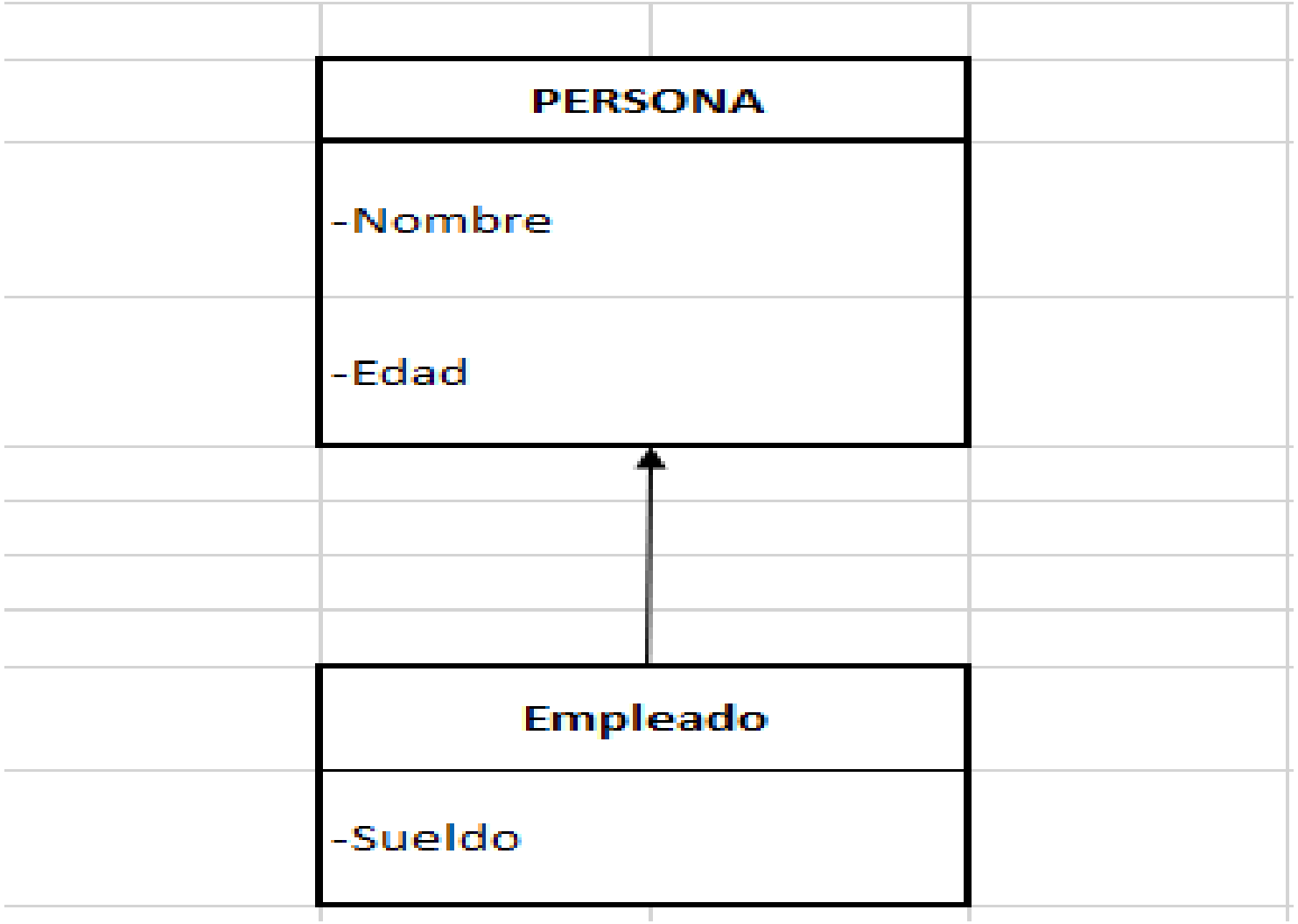
Herencia

¿Qué necesitamos para programar utilizando herencia?

Necesitamos dos clases, una clase principal y una clase hija que cumplen con las siguientes funciones:

- ✓ La clase principal es la clase de la que se hereda, también llamada clase base.
- ✓ La clase hija es la clase que hereda de otra clase, también llamada clase derivada.

Herencia

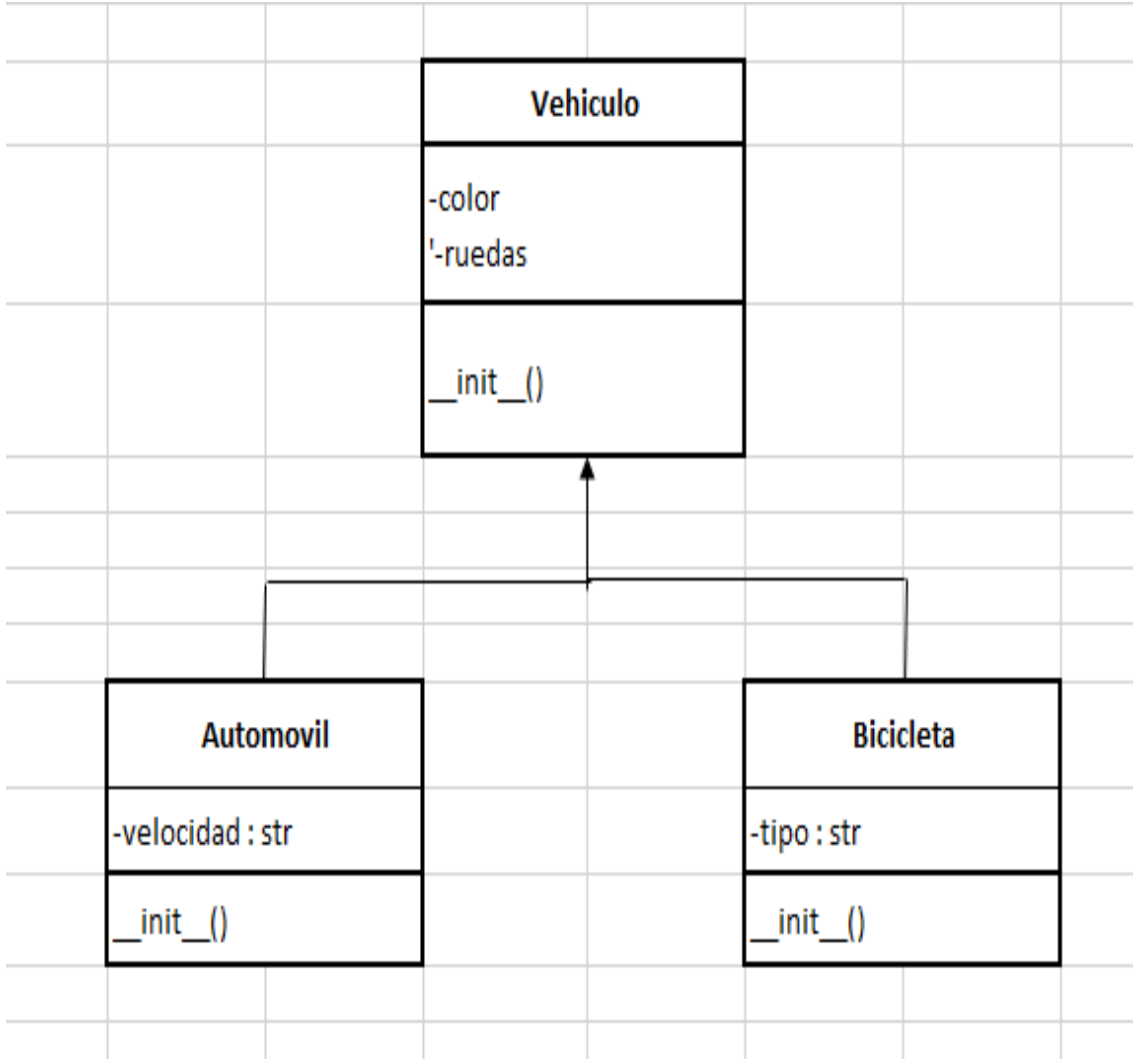


Solución

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

class Empleado(Persona):
    def __init__(self, sueldo):
       self.sueldo = sueldo

empleado1 = Empleado('Juan',25,5000)
print(empleado1.nombre)
print(empleado1.edad)
print(empleado1.sueldo)
```

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

class Empleado(Persona):
    def __init__(self, nombre, edad, sueldo):
        super().__init__(nombre, edad)
        self.sueldo = sueldo

empleado1 = Empleado('Juan', 28, 5000)
print(empleado1.nombre)
print(empleado1.edad)
print(empleado1.sueldo)
```

Problema Propuesto




```
Creamos un objeto de la clase Vehiculo
color: Rojo
ruedas: 4
Creamos un objeto de la clase hija Coche
color: Azul
ruedas: 4
velocidad: 150
Creamos un objeto de la clase hija Bicicleta
color: Blanca
ruedas: 2
tipo: Urbano
```

```
class Vehiculo:
    def __init__(self, color, ruedas):
        self.color = color
        self.ruedas = ruedas

class Automovil(Vehiculo):
    def __init__(self, color, ruedas, velocidad):
        super().__init__(color, ruedas)
        self.velocidad = velocidad

class Bicicleta(Vehiculo):
    def __init__(self, color, ruedas, tipo):
        super().__init__(color, ruedas)
        self.tipo = tipo
```

```
print('Creamos un objeto de la clase Vehiculo')
vehiculo = Vehiculo('Rojo', 4)
print(f'color: {vehiculo.color}')
print(f'ruedas: {vehiculo.ruedas}')

print('Creamos un objeto de la clase hija Coche')
coche = Automovil('Azul', 4, 150)
print(f'color: {coche.color}')
print(f'ruedas: {coche.ruedas}')
print(f'velocidad: {coche.velocidad}')

print('Creamos un objeto de la clase hija Bicicleta')
bicicleta = Bicicleta('Blanca', 2, 'Urbano')
print(f'color: {bicicleta.color}')
print(f'ruedas: {bicicleta.ruedas}')
print(f'tipo: {bicicleta.tipo}')
```

Herencia

Problema 1

Plantear una clase Persona que contenga dos atributos: nombre y edad. Definir como responsabilidades la carga por teclado y su impresión.
Declarar una segunda clase llamada Empleado que herede de la clase persona y agregue un atributo sueldo y muestre si debe pagar impuestos (sueldo superior a 3000)
En el bloque principal del programa crear un objeto de la clase empleado y llamar a sus métodos.

Solución

```
class Persona:
        def __init__ (self):
                self.nombre=input("Ingrese el nombre:")
                self.edad=int(input("Ingrese la edad:")

        def imprimir(self):
                print("Nombre:",self.nombre)
                print("Edad:", self.edad)
```

Herencia

```
class Empleado (Persona):
        def __init__(self):
                super().__init__()
                self.sueldo=float(input("Ingrese Sueldo:"))

        def imprimir(self):
                super().imprimir()
                print("Sueldo:", self.sueldo)

        def paga impuestos(self):
                if self.sueldo>3000:
                        print(self.nombre, "debe pagar impuesto")
                else:
                        print(self.nombre, "no paga impuesto")
#Bloque Principal

empleado1=Empleado()
empleado1.imprimir()
empleado1.paga_impuestos()
```

Herencia

Problema 2

Implementar dos clases que llamaremos Suma y Resta. Cada clase tiene como atributo valor1, valor2 y resultado. Los métodos a definir son cargar1 (que inicializa el atributo valor1). Cargar2 (que inicializa el atributo valor2), operar (que en el caso de la clase “Suma” suma los dos atributos y en el caso de la clase “Resta” hace la diferencia entre valor1 y valor2). Y otro método mostrar_resultado.

Si analizamos ambas clases encontramos que muchos atributos y métodos son idénticos. En estos casos es bueno definir una clase padre que agrupe dichos atributos y responsabilidades comunes.
La relación de herencia que podemos disponer para este problemas es:

Operación

Suma Resta

Solamente el método operar es distinto para las clases Suma y Resta (esto hace que no lo podamos disponer en la clase Operación en principio). Luego los métodos cargar1, cargar2 y mostrar_resultado son idénticos a las dos clases, esto hace que podamos disponerlos en la clase Operación. Lo mismo los atributos valor1, valor2 y resultado se definirían en la clase padre Operación.

Herencia

Problema 3

Declarar una clase cuenta y dos subclases CajaAhorro y PlazoFijo. Definir los atributos y métodos comunes entre una caja de ahorro y un plazo fijo y agruparlos en la clase Cuenta.
Una caja de ahorro y un plazo fijo tienen un nombre de titular y un monto. Un plazo fijo añade un plazo de imposición en días y una tasa de interés. Hacer que la caja de ahorro no genera intereses.
En el bloque principal del programa definir un objeto de la clase CajaAhorro y otro de la clase PlazoFijo.

Herencia

Problema 4

Crear la clase padre "Poligono", con el atributo protegido "nombre".- Heredar de la clase "Polígono" a la clase "Cuadrilatero" con los atributos privados:

- num_lados: Indica el número de lados
- tipo: Indica si es regular o irregular

Heredar de la clase "Cuadrilatero" a la clase "Rectangulo" con los atributos protegidos:

- largo: Expresa la medida en un valor numérico.
- alto: Expresa la medida en un valor numérico.

Además implementar:

- Los métodos get y set de los atributos de instancia
- El método "área", que retorna el valor del área
- El método "perimetro" que retorna el valor del perímetro
- El método listado que muestra todo lo referente a la clase "Rectangulo“

Crear un objeto de tipo rectángulo y utilizar el método listado() para mostrar lo referente al objeto de tipo "Rectángulo"