



# Universidad Autónoma de Baja California

## Facultad de Ingeniería, Arquitectura y Diseño



### Paradigmas de la Programación

## *Práctica 3*

# Paradigma Orientado a Objetos

DIEGO OSUNA ACEVEDO

372273

22/marzo/2024.

## Introducción

El objetivo principal de esta práctica es comprender y aplicar estos conceptos de Clases, Objetos, Abstracción de datos, Encapsulamiento, Herencia y Polimorfismo mediante la implementación de un sistema bancario simplificado que permita gestionar cuentas bancarias y realizar operaciones comunes, como depósitos, retiros y cálculo de intereses.

Pasos que se siguieron:

Crear una aplicación en Python utilizando el paradigma orientado a objetos. En grupo definir la aplicación y sus requerimientos.

Debe de manejar e indicar en el código los siguientes conceptos:

Clases

```
class Cuenta:
    def __init__(self, numero_cuenta, nombre, saldo=0):
        self._numero_cuenta=numero_cuenta
        self._nombre=nombre
        self._saldo=saldo

    def depositar(self, monto):
        if monto>0:
            self._saldo += monto
            print(f"Deposito de {monto} realizado. Saldo : {self._saldo}")
        else:
            print("Cantidad no valida")

    def retirar(self, monto):
        if 0 < monto<=self._saldo:
            self._saldo-=monto
            print(f"Retiro de {monto} realizado. Saldo : {self._saldo}")
        else:
            print("Cantidad no valida")

    def mostrar_saldo(self):
        print(f"Saldo actual:{self._saldo}")

    def mostrar_informacion(self):
        print(f"Numero de Cuenta: {self._numero_cuenta}")
        print(f"Titular: {self._nombre}")
        print(f"Saldo: {self._saldo}")
```

## Objetos

```
banco=Banco("Ejemplo")
cuenta1=Cuenta("004", "Diego Osuna", 500)
cuenta2=CuentaDeAhorros("444", "Osuna Diego", 1000, 0.05)
```

## Abstracción de datos

```
class Cuenta:
    def __init__(self, numero_cuenta,nombre,saldo=0):
        self._numero_cuenta=numero_cuenta
        self._nombre=nombre
        self._saldo=saldo

    def depositar(self,monto):
        if monto>0:
            self._saldo += monto
            print(f"Deposito de {monto} realizado. Saldo : {self._saldo}")
        else:
            print("Cantidad no valida")

    def retirar(self,monto):
        if 0 < monto<=self._saldo:
            self._saldo-=monto
            print(f"Retiro de {monto} realizado. Saldo : {self._saldo}")
        else:
            print("Cantidad no valida")

    def mostrar_saldo(self):
        print(f"Saldo actual:{self._saldo}")

    def mostrar_informacion(self):
        print(f"Numero de Cuenta: {self._numero_cuenta}")
        print(f"Titular: {self._nombre}")
        print(f"Saldo: {self._saldo}")
```

La clase Cuenta actúa como una abstracción de datos para representar una cuenta bancaria.

## Encapsulamiento

```
class Banco:
    def __init__(self,nombre):
        self._nombre=nombre
        self._cuentas=[]
```

## Herencia

```
# Herencia:
class CuentaDeAhorros(Cuenta):
    def __init__(self, numero_cuenta, nombre, saldo=0, tasa_interes=0.01):
        super().__init__(numero_cuenta, nombre, saldo)
        self._tasa_interes = tasa_interes

    def aplicar_interes(self):
        if self._saldo > 0:
            interes = self._saldo * self._tasa_interes
            self._saldo += interes
            print(f"Interes de {interes} Saldo: {self._saldo}")
        else:
            print("Error.")

    def mostrar_informacion(self):
        super().mostrar_informacion()
        print(f"Tasa de Interes: {self._tasa_interes}")
```

La herencia se encuentra ya que esta clase (Cuenta de ahorros) es una subclase de la clase base (cuenta).

## Polimorfismo

```
# Polimorfismo:
class Banco:
    def __init__(self, nombre):
        self._nombre = nombre
        self._cuentas = []

    def agregar_cuenta(self, cuenta):
        self._cuentas.append(cuenta)
        print(f"Cuenta {cuenta._numero_cuenta} agregada al banco {self._nombre}.")

    def mostrar_cuentas(self):
        print(f"Cuentas en el banco {self._nombre}:")
        for cuenta in self._cuentas:
            cuenta.mostrar_informacion()
            print("-----")
```

En la clase Banco, el polimorfismo se encuentra en el método mostrar cuentas(). Ya que puede mostrar información sobre diferentes tipos de cuentas bancarias.

