

Python

```
# =====  
# EJEMPLO DE ABSTRACCIÓN  
# Objetivo: Ocultar la complejidad interna y exponer una interfaz simple.  
# =====
```

```
from abc import ABC, abstractmethod
```

```
# 1. Clase Abstracta (Define la Interfaz Pública)
```

```
class CuentaBancaria(ABC):
```

```
    """
```

```
    Clase abstracta que define las operaciones públicas esenciales.
```

```
    Los usuarios solo necesitan conocer estos métodos.
```

```
    """
```

```
    def __init__(self, saldo_inicial):
```

```
        self._saldo = saldo_inicial # Atributo protegido
```

```
    @abstractmethod
```

```
    def depositar(self, monto):
```

```
        pass # La implementación concreta se hará en las subclases
```

```
    @abstractmethod
```

```
    def retirar(self, monto):
```

```
        pass # La implementación concreta se hará en las subclases
```

```
# Método concreto expuesto
```

```
def obtener_saldo(self):
```

```
    """Método público y simple para consultar el saldo."""
```

```
return self._saldo
```

2. Clase Concreta (Implementa y Oculta la Complejidad)

```
class CuentaAhorros(CuentaBancaria):
```

```
    """
```

Clase concreta que implementa la lógica interna de las operaciones sin exponer los detalles al usuario.

```
    """
```

```
def depositar(self, monto):
```

```
    """Implementación simple de depósito."""
```

```
    if monto > 0:
```

```
        self._saldo += monto
```

```
        print(f'Depósito de ${monto} realizado. Saldo actual: ${self._saldo}')
```

```
    else:
```

```
        print("Error: El monto a depositar debe ser positivo.")
```

```
def retirar(self, monto):
```

```
    """
```

Implementación de retiro con una regla de negocio (validación).

Esta validación es un detalle oculto al usuario final.

```
    """
```

```
    if monto > 0 and self._saldo >= monto:
```

```
        self._saldo -= monto
```

```
        print(f'Retiro de ${monto} realizado. Saldo actual: ${self._saldo}')
```

```
    else:
```

```
        print(f'Error: No se pudo retirar ${monto}. Saldo insuficiente o monto inválido.')
```

```
# =====  
# PRUEBA Y DEMOSTRACIÓN  
# =====
```

```
cuenta = CuentaAhorros(saldo_inicial=500)
```

```
print(f"Saldo inicial: ${cuenta.obtener_saldo()}")
```

```
# El usuario interactúa solo con la interfaz simple
```

```
cuenta.depositar(200)
```

```
cuenta.retirar(75)
```

```
# La complejidad (validación) está oculta, pero se aplica
```

```
cuenta.retirar(1000)
```