

2. Encapsulación: Ejemplo de Empleado ☒

La encapsulación se logra protegiendo los atributos internos de una clase y forzando el acceso a través de métodos públicos que pueden aplicar validaciones o reglas de negocio.

Encapsulacion/EncapsulacionEjemplo.py

Python

```
# =====  
# EJEMPLO DE ENCAPSULACIÓN  
# Objetivo: Proteger los datos (atributos privados) y controlar  
# su acceso mediante métodos públicos (getters y setters).  
# =====
```

```
class Empleado:
```

```
    """
```

```
    Clase que encapsula el salario del empleado para protegerlo de  
    modificaciones directas e inválidas.
```

```
    """
```

```
    def __init__(self, nombre, salario_base):
```

```
        self._nombre = nombre    # Atributo protegido
```

```
        self._salario = self._validar_salario(salario_base) # Atributo
```

```
        privado/protegido
```

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

```
        """Método privado para asegurar que el salario es válido."""
```

```
        if monto < 0:
```

```
            print("ADVERTENCIA: Salario no puede ser negativo. Estableciendo a 0.")
```

```
            return 0
```

```

        return monto

# Método GETTER (Para consultar el dato)
def obtener_salario(self):
    """Expone el valor del salario de forma segura."""
    return self._salario

# Método SETTER (Para modificar el dato con control)
def establecer_salario(self, nuevo_salario):
    """Permite modificar el salario solo si pasa la validación."""
    salario_validado = self._validar_salario(nuevo_salario)
    if self._salario != salario_validado:
        self._salario = salario_validado
        print(f'Salario de {self._nombre} actualizado a: ${self._salario}')

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

empleado = Empleado("Carlos", 50000)

# 1. Acceso indirecto (Uso correcto de la Encapsulación)
salario_actual = empleado.obtener_salario()
print(f'Salario actual de {empleado._nombre}: ${salario_actual}')

# 2. Modificación controlada (El Setter aplica la validación)
empleado.establecer_salario(55000)

```

3. Intento de modificación inválida (El Setter previene el cambio)
empleado.establecer_salario(-10000)
