**Ejercicios de Encapsulamiento**

**Ejercicio 1: Control de Acceso a Atributos**

Crea una clase `CuentaBancaria` que tenga los siguientes atributos privados: `saldo`, `titular`, y `numero\_cuenta`. Implementa métodos públicos para:

- Depositar dinero en la cuenta.

- Retirar dinero de la cuenta (si el saldo es suficiente).

- Consultar el saldo actual.

Instrucciones:

- Asegúrate de que el saldo no se pueda modificar directamente desde fuera de la clase.

- Añade un método `mostrar\_detalles` para mostrar la información de la cuenta (titular y número de cuenta), pero no el saldo.

**Ejercicio 2: Control de Acceso en una Clase de Persona**

Crea una clase `Persona` con los atributos privados `nombre`, `edad`, y `dni`. Implementa los siguientes métodos públicos:

- `cumplir\_años`: Incrementa la edad en 1.

- `mostrar\_informacion`: Imprime la información de la persona (nombre y edad).

- `cambiar\_dni`: Permite cambiar el DNI solo si el nuevo DNI tiene 8 dígitos.

Instrucciones:

- Implementa las restricciones adecuadas para asegurar el encapsulamiento de los atributos.

**Ejercicios de Abstracción**

**Ejercicio 1: Clase Abstracta de Figura Geométrica**

Crea una clase abstracta `FiguraGeometrica` que tenga los métodos abstractos `area` y `perimetro`. Luego, crea dos clases concretas `Rectangulo` y `Circulo` que hereden de `FiguraGeometrica` e implementen los métodos abstractos.

Instrucciones:

- La clase `Rectangulo` debe tener atributos para el largo y el ancho.

- La clase `Circulo` debe tener un atributo para el radio.

- Implementa los métodos `area` y `perimetro` en ambas clases.

**Ejercicio 2: Clase Abstracta de Vehículo**

Crea una clase abstracta `Vehiculo` con los métodos abstractos `arrancar`, `acelerar`, y `frenar`. Luego, crea dos clases concretas `Coche` y `Bicicleta` que hereden de `Vehiculo` e implementen estos métodos.

Instrucciones:

- `Coche` debe tener un atributo para la marca y la velocidad actual.

- `Bicicleta` debe tener un atributo para el tipo de bicicleta y la velocidad actual.

- Implementa los métodos `arrancar`, `acelerar`, y `frenar` en ambas clases.

**Ejercicios de Decoradores**

**Ejercicio 1: Decorador para Medir el Tiempo de Ejecución**

Crea un decorador `medir\_tiempo` que mida el tiempo de ejecución de una función. Luego, aplícalo a una función que calcule la suma de los primeros N números naturales.

Instrucciones:

- Usa el módulo `time` para medir el tiempo.

- Asegúrate de que el decorador funcione correctamente para cualquier función a la que se aplique.

**Ejercicio 2: Decorador para Comprobar Permisos**

Crea un decorador `verificar\_permiso` que simule la verificación de permisos antes de ejecutar una función. Si el usuario no tiene permiso, se debe mostrar un mensaje indicando la falta de acceso y no ejecutar la función.

Instrucciones:

- El decorador debe aceptar un argumento que indique el tipo de permiso requerido.

- Aplica el decorador a una función `acceso\_restringido` que solo pueda ser ejecutada si el usuario tiene el permiso adecuado.