Taller de practica

Ejercicios con Herencia

La herencia permite crear nuevas clases basadas en clases existentes, reutilizando código y organizando jerarquías.

1. Clase `Animal` y subclases

Crea una clase base `Animal` con atributos como `nombre` y `edad`, y métodos como `comer()` y `dormir()`. Luego, define subclases como `Perro`, `Gato` y `Pájaro` que hereden de `Animal` y añadan métodos específicos (`ladrar()`, `maullar()`, `volar()`).

2. Jerarquía de empleados

Define una clase `Empleado` con atributos como `nombre`, `salario` y método `calcular_bono()`. Luego, crea subclases como `Gerente`, `Desarrollador` y `Asistente` que hereden de `Empleado` y modifiquen el cálculo del bono según su rol.

3. Figuras geométricas

Crea una clase base `Figura` con métodos `area()` y `perimetro()`. Luego, implementa subclases como `Circulo`, `Rectangulo` y `Triangulo` que hereden de `Figura` y calculen sus áreas y perímetros según sus atributos.

4. Sistema de vehículos

Define una clase `Vehiculo` con atributos como `marca`, `modelo` y método `arrancar()`. Luego, crea subclases como `Coche`, `Moto` y `Camion` que hereden de `Vehiculo` y añadan métodos específicos (`abrir_maletero()`, `hacer_caballito()`).

5. Sistema de videojuegos

Crea una clase `Personaje` con atributos como `nombre`, `vida` y método `atacar()`. Luego, define subclases como `Guerrero`, `Mago` y `Arquero` que hereden de `Personaje` y modifiquen el comportamiento del ataque según su tipo.

◇ Ejercicios con Polimorfismo

El polimorfismo permite que diferentes clases compartan un mismo método pero con comportamientos distintos.

6. Método `hablar()` en animales

Define una clase `Animal` con un método `hablar()`. Luego, implementa subclases como `Perro`, `Gato` y `Vaca` que sobrescriban `hablar()` para que cada animal emita su sonido característico ("Guau", "Miau", "Muu").

7. Método `dibujar()` en figuras

Crea una clase `Figura` con un método abstracto `dibujar()`. Implementa subclases como `Circulo`, `Cuadrado` y `Triangulo` que sobrescriban `dibujar()` para imprimir cómo se dibuja cada figura.

8. Sistema de pagos

Define una clase `MetodoPago` con un método `pagar(monto)`. Luego, crea subclases como `Tarjeta`, `Paypal` y `Efectivo` que implementen `pagar()` de manera diferente (ej: "Pagando con tarjeta...", "Pagando en efectivo...").

9. Reproductor multimedia

Crea una clase `Reproductor` con un método `reproducir()`. Luego, implementa subclases como `ReproductorMP3`, `ReproductorVideo` y `ReproductorStreaming` que sobrescriban `reproducir()` según el tipo de archivo.

10. Sistema de notificaciones

Define una clase `Notificacion` con un método `enviar(mensaje)`. Luego, crea subclases como `Email`, `SMS` y `PushNotification` que implementen `enviar()` de forma distinta según el canal.

◇ Ejercicios con Encapsulamiento

El encapsulamiento protege los atributos de una clase, permitiendo acceso controlado mediante métodos (getters y setters).

11. Clase `CuentaBancaria`

Crea una clase `CuentaBancaria` con atributos privados (`__saldo`, `__titular`) y métodos públicos (`depositar()`, `retirar()`, `get_saldo()`). Asegúrate de validar que no se pueda retirar más dinero del disponible.

12. Clase `Persona` con datos sensibles

Define una clase `Persona` con atributos privados (`__nombre`, `__edad`, `__dni`) y métodos públicos (`get_nombre()`, `set_edad()`) que validen que la edad no sea negativa.

13. Clase `Coche` con atributos protegidos

Crea una clase `Coche` con atributos protegidos (`_marca`, `_modelo`) y métodos públicos (`get_info()`, `set_modelo()`). Añade validación para que el modelo no sea una cadena vacía.

14. Sistema de inventario

Implementa una clase `Producto` con atributos privados (`__nombre`, `__precio`, `__stock`) y métodos públicos (`vender()`, `reponer()`, `get_info()`). Valida que el stock nunca sea negativo.

15. Clase `Alumno` con notas privadas

Crea una clase `Alumno` con atributos privados (`__nombre`, `__notas`) y métodos públicos (`agregar_nota()`, `promedio()`, `mejor_nota()`). Asegúrate de que las notas estén entre 0 y 10.

Ejercicio Integrador (Herencia + Polimorfismo + Encapsulamiento)

16. Sistema de reservas de hotel

- Crea una clase base `Huesped` con datos encapsulados (`__nombre`, `__dni`).
- Implementa subclases como `HuespedVIP` y `HuespedGrupo` con beneficios distintos.
- Usa polimorfismo en métodos como `calcular_descuento()`.