Herencia Múltiple

1. Sistema de Vehículos Híbridos:

- Crea una clase general llamada Vehiculo que tenga atributos básicos como nombre, marca, modelo, velocidad_maxima.
- Crea clases VehiculoAcuatico y VehiculoAereoque hereden de Vehiculo.
    - VehiculoAcuatico debe tener atributos como profundidad_maxima.
    - VehiculoAereo debe tener atributos como altura_maxima.
    - Ambos deben tener métodos particulares como navegar() o volar().

- Crea la clase VehiculoHibrido que herede de ambos (VehiculoAcuatico y VehiculoAereo).
    - Implementa métodos adicionales indicando si puede navegar, volar, o ambos.
    - Incluye en su constructor atributos como capacidad_combustible.

- Prueba creando diferentes objetos y llamando sus métodos para mostrar sus capacidades.

2. Empleados Multifuncionales en una Empresa de Tecnología:

- Crea una clase base Empleado con atributos comunes (nombre, apellido, id, salario).
- Hereda de ella las clases Programador y Diseñador, cada uno con atributos específicos (por ejemplo, lenguajes_programacion, herramientas_diseno).
- Diseña una clase EmpleadoMultifuncional que herede de ambas.
    - Añade atributos adicionales para gestionar tareas y proyectos específicos a cada rol.
    - Incluye métodos que permitan:
        - Agregar nuevas habilidades.
        - Listar todas las habilidades y tareas del empleado multifuncional.
- Crea instancias y demuestra cómo gestionar las habilidades en tiempo de ejecución.

3. Sistema de Dispositivos Electrónicos:

- Define la clase Dispositivo con atributos básicos (marca, modelo) y método mostrar_info().
- Crea clases Smartphone y Tablet que hereden de Dispositivo.
    - Smartphone tiene atributos como sistema_operativo, numero_telefono.

    - Tablet tiene atributos como tamaño_pantalla, tipo_conectividad.
- Implementa una clase DispositivoPortatil que herede de ambas clases.
    - Esta clase debe incluir métodos para verificar compatibilidad con accesorios y funciones específicas.
- Crea objetos y demuestra cómo gestionar estos dispositivos desde su clase base y concreta.

4. Personajes en RPG Multiclase:

- Diseña una clase base Personaje con atributos comunes: nombre, nivel, salud.
- Clases Mago y Guerrero heredan de Personaje y añaden habilidades específicas:
    - Mago con atributos como mana, tipo_magia, y método escribir_hechizo().
    - Guerrero con atributos como fuerza, armadura, y método atacar().
- Crea una clase MagoGuerrero que herede de ambas clases.
    - Implementa un método que combine habilidades y atributos, por ejemplo, atacar_conmagia().
    - Gestiona niveles y atributos combinados para demostrar la interoperabilidad.
- Incluye una demostración de cómo cambiar habilidades o niveles.

5. Animales con habilidades múltiples (Volar y Nadar):

- Crea clases: Animal (como clase base), Volador, y Nadador.
- Cada habilidad tiene atributos y métodos:
    - Volador: altura que puede alcanzar, método volar().
    - Nadador: profundidad de buceo, método nadar().
- Crea una clase PinguinoVoladorNadador, que herede de Animal, Volador y Nadador.
- Agrega atributos que controlen la energía y habilidades especiales.
- Crea métodos que permitan mostrar las habilidades disponibles y gestionar sus atributos.

Clases Abstractas

1. Figuras geométricas:

- Crea una clase abstracta Forma con atributos comunes (como color o línea).
- Declara métodos abstractos area() y perimetro().
- Implementa clases concretas:
    - Rectangulo con atributos base y altura, que calcula area() y perimetro().
    - Circulo con atributo radio.
    - Triangulo con atributos lado1, lado2, lado3.
- La tarea es que cada clase implemente los métodos de cálculo de forma correcta, considerando sus fórmulas específicas.
- Agrega también un método mostrar_detalles() en la clase base, que cada clase sobrescriba para mostrar sus atributos y resultados.

2. Sistema de Métodos de Pago

- Diseña una clase abstracta MetodoPago con el método abstracto pagar().
- Incluye atributos comunes como monto y numero_transaccion.
- Crea clases concretas:
    - TarjetaCredito, que en pagar() verifica la validez, realiza una simulación de cargo y muestra un mensaje.
    - PayPal, que simula la transferencia y muestra un mensaje de éxito.
    - Bitcoin, con método pagar() que muestra el hash generado y un mensaje de confirmación.
- Añade en cada clase atributos específicos relevantes (por ejemplo, numero_tarjeta, email para PayPal).
- La clase base puede definir métodos comunes como validar_monto().

3. Vehículos con comportamiento según combustible

- Crea una clase abstracta Vehiculo con atributos tipo_combustible y velocidad.
- Declara método abstracto acelerar() y frenar().
- Implementa clases Auto y Motocicleta que hereden de Vehiculo.
    - En acelerar(), mostrar mensajes específicos y aumentar la velocidad.
    - En frenar(), disminuir la velocidad.
    - Incluye lógica para comprobar la compatibilidad del combustible (gasolina, electricidad, etc.).

- Agrega atributos adicionales como numero_de_ruedas y métodos para mostrar el estado del vehículo.

4. Sistema de empleados con diferentes modelos de sueldo

- Crea una clase abstracta Empleado con atributos comunes: nombre, id, sueldo_base.
- Declara un método abstracto calcular_sueldo().
- Implementa clases:
    - EmpleadoTiempoCompleto: en calcular_sueldo() suma sueldo_base + bonos.
    - EmpleadoPorHoras: en calcular_sueldo(), multiplica horas_trabajadas por tarifa_por_hora.
- Agrega atributos adicionales en cada clase, como bonos, horas_trabajadas, tarifa.
- Implementa métodos para modificar estos atributos y calcular el sueldo en tiempo de ejecución.

5. Dispositivos electrónicos

- Crea una clase abstracta Dispositivo con atributos marca, modelo, y método abstracto encender().
- Define clases concretas:
    - Laptop con atributos como cantidad_ram, procesador.
    - Tableta con atributos como tamaño_pantalla y tipo_bateria.
- Cada clase implementa encender() de forma diferente, mostrando mensajes específicos.
- Añade métodos secundario como apagar(), reiniciar() y atributos que describan su estado actual.
- Incluye en cada clase un método mostrar_informacion().