TALLER JAVA 9 - PROGRAMACIÓN ORIENTADA A OBJETOS

Ejercicio 1: Clase CuentaBancaria

Imagina que trabajas para un banco y debes diseñar un sistema seguro para manejar cuentas bancarias.

1. Crea una clase llamada **CuentaBancaria** con estos atributos privados:

•**numeroCuenta** (String): Identificador único de la cuenta.
•**nombreTitular** (String): Nombre del dueñx de la cuenta.
•**saldo** (double): Dinero disponible (debe ser siempre ≥ 0).
•**tipoCuenta** (String): "Ahorros" o "Corriente".

2. Implementa estos métodos:

•**depositar(double monto)**: Suma al saldo solo si el monto es positivo.
•**retirar(double monto)**: Resta del saldo si hay fondos suficientes y el monto es válido.
•**mostrarDatos()**: Imprime todos los datos de la cuenta.
•Getters y setters para todos los atributos.

3.Crea dos constructores:

•Constructor vacío.
•Constructor con parámetros.

4. En el **main**:

•Crea una cuenta con el constructor vacío y usa setters para asignar datos.
•Crea otra cuenta con el constructor parametrizado.
•Realiza depósitos y retiros (incluyendo casos inválidos como montos negativos).
•Muestra los datos finales de ambas cuentas.

Ejercicio 2: Clase Libro

Trabajas en una biblioteca y necesitas gestionar préstamos de libros.

1.Clase **Libro** con atributos privados:

•**titulo** (String).
•**autor** (String).
•**anioPublicacion** (int).
•**disponible** (boolean): Indica si está prestado o no (valor por defecto: **true**).
2.Métodos:

•**mostrarDetalles()**: Imprime título, autor, año y disponibilidad.
•**prestar()**: Cambia **disponible** a **false** si el libro está disponible.
•**devolver()**: Reestablece **disponible** a **true**.
3.En el **main**:

•Crea un libro con el constructor parametrizado.
•Intenta prestarlo dos veces (la segunda debe fallar).
•Devuélvelo y vuelve a intentar el préstamo.
•Muestra los detalles después de cada operación.

Ejercicio 3: Clase Auto

Desarrollas un sistema para un concesionario de autos.

1.Clase **Coche** con atributos privados:

•**marca** (String).
•**modelo** (String).
•**anio** (int).
•**kilometraje** (int): Valor por defecto: 0.
2.Métodos:

•**mostrarInformacion()**: Imprime marca, modelo y año.
•**MostrarInformacion:** Si **detallado** es **true**, muestra también el kilometraje.
•**actualizarKilometraje(int km)**: Suma kilómetros solo si el valor es positivo.

3.En el **main**:

•Crea un coche con el constructor vacío y usa setters para asignar datos.
•Crea otro coche con el constructor parametrizado.
•Actualiza el kilometraje del primer coche (con valores válidos e inválidos).
•Muestra la información de ambos coches (normal y detallada).

Ejercicio 4: Clase Pedido

Gestionas pedidos para una empresa de delivery.

1.Clase **Pedido** con atributos privados:

•**numeroPedido** (int).
•**nombreCliente** (String).
•**total** (double).
•**productos** (List<String>): Lista de ítems en el pedido.
2.Métodos:

•**agregarProducto(String producto, double precio)**: Añade un producto y actualiza el total.
•**calcularDescuento(double porcentaje)**: Aplica descuento solo si el porcentaje está entre 0 y 100.
•**mostrarPedido()**: Imprime número de pedido, cliente, productos y total.
3.En el **main**:

•Crea un pedido con constructor vacío y agrega 3 productos.
•Crea otro pedido con constructor parametrizado y agrega 2 productos.
•Aplica descuentos válidos e inválidos .
•Muestra ambos pedidos.