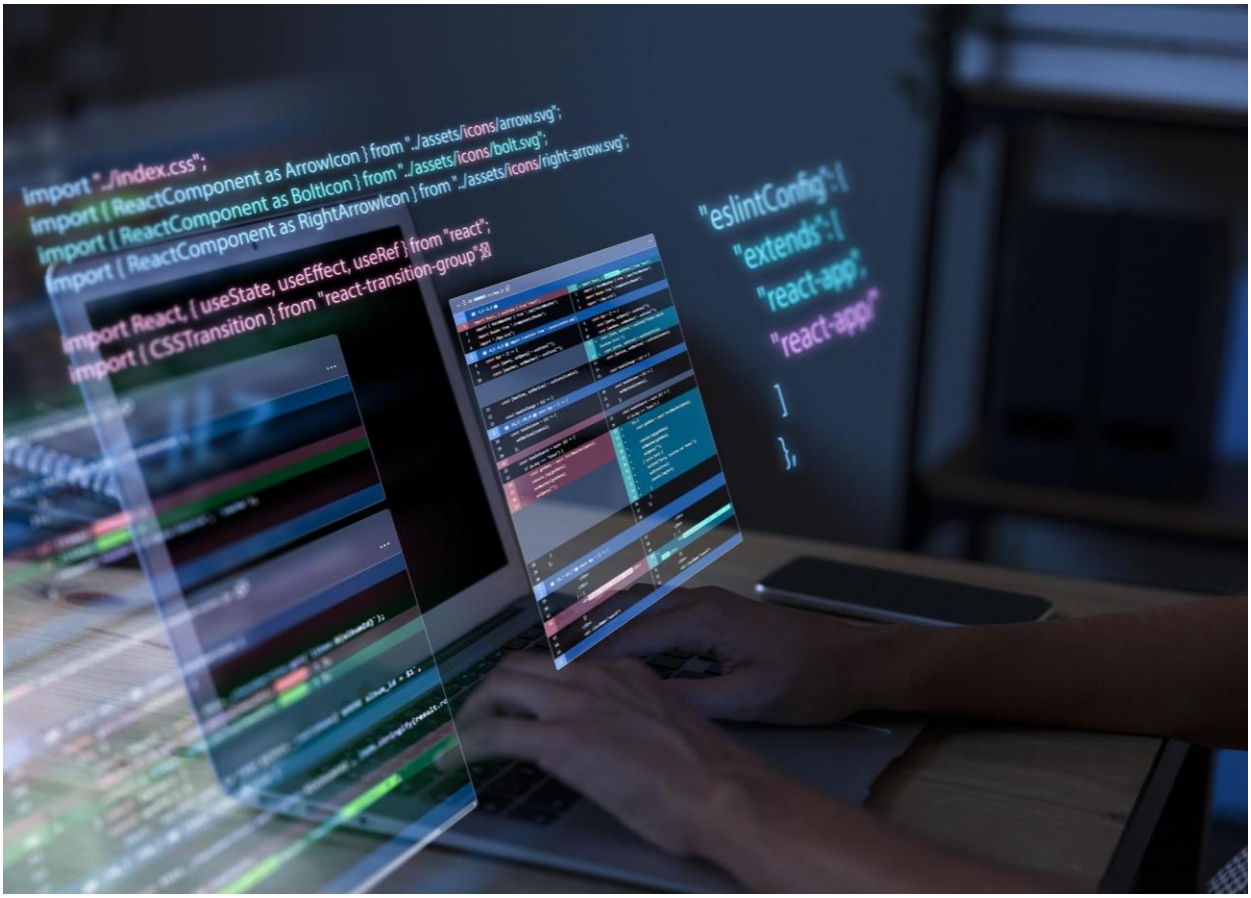
DAM-DAW/PROGRAMACIÓN

Ejercicios

POO (III)



Autor: Marcela Martín

Fecha: enero 2026

Ejercicio 1. Diseña un sistema para la gestión de un concesionario.

- Crea la clase **Vehiculo** que almacena los datos: matrícula, marca y kilometraje.
- Implementa las siguientes operaciones para un vehículo:
    - Crear un vehículo: Se necesita la matrícula y la marca. El kilometraje inicial es 0.
    - Registrar kilómetros: Incrementa el kilometraje del vehículo.
    - Mostrar información: Muestra la información disponible del vehículo.
- Sobrecarga el constructor para poder crear objetos:
    - Con la matrícula y el kilometraje inicial.
    - Con la matrícula, marca y el kilometraje inicial.
- Modifica la visibilidad de la clase Vehiculo para que sea visible desde clases externas. Además, respecto a la visibilidad de los atributos:
    - kilometraje no es visible para otras clases.
    - marca sea visible para cualquier clase.
    - matrícula solo es visible por clases vecinas.
- Todos los vehículos de la flota pertenecen a la misma empresa. Añade un atributo estático para almacenar el nombre de la empresa.
- Existen **vendedores** que se encargan de la venta de los vehículos. Cada vehículo está asignado a un vendedor. Diseña la clase Vendedor que almacena:
    - Nombre del vendedor, visible desde cualquier clase.
    - Email, no modificable
    - Límite de kilometraje autorizado, visible solo por clases vecinas.
- Con respecto a los vendedores las restricciones son:
    - Un vendedor siempre tendrá un nombre y un email.
    - Si no se asigna, el límite de kilometraje será de 50000 km.
    - El email no podrá cambiar una vez asignado, pero todo el mundo podrá consultarlo.
- Modifica la clase Vehiculo para asignarle un Vendedor.
- Crea una clase AppVehiculo para probar todos los requerimientos citados anteriormente.

Ejercicio 2. Diseña un sistema para la gestión de una biblioteca.

- Implementa la clase **Libro** que tiene los atributos:
    - Título : visible desde cualquier clase
    - ISBN : visible solo para clases vecinas
    - Disponibilidad ,indica si está disponible para préstamo o no. No visible para otras clases
    - TotalLibros: contador de libros creados: dato compartido por todos los 'Libros'.
- Funcionalidades de Libro:
    - Crear un libro con título e ISBN: Por defecto, un libro está disponible.
    - Con la creación de cada libro se incrementa el contador 'TotalLibros'.
    - Pedir un libro: Cambia el estado de disponibilidad a 'no disponible'. Si el libro ya está prestado se informa al usuario.
    - Devolver un libro: Cambia el estado de disponibilidad a 'disponible'.
    - Mostrar información: Muestra los datos del libro.
- Implementa la clase **Autor** que tiene los atributos:
    - Nombre: visible desde cualquier clase
    - Email: se puede consultar pero no modificar una vez asignado.
    - Número de libros publicados: visible desde las clases vecinas
- Funcionalidades de Autor:
    - Un autor siempre tiene un nombre y un email.
    - Se puede incrementar el número de libros publicados desde dentro de la clase o por clases vecinas.
    - Los datos del autor deben poder mostrarse.
- Todos los libros tienen un
- Crea una clase AppBiblioteca para probar todos los requerimientos citados anteriormente.

Ejercicio 3. Diseña un sistema para la gestión de un centro educativo.

- De las Aulas se quiere almacenar:
    - **Nombre**, visible por todas las clases
    - **Planta**, visible por clases vecinas

    - **Ocupación**, número de plazas ocupadas. no visible
- Crea un constructor para Aula que reciba los 3 atributos
- Es posible actualizar la ocupación en un Aula
- En la actualización de la ocupación se valida que el nuevo valor sea positivo
- Se habilita la consulta de los datos de las aulas.
- De los profesores se quiere almacenar:
    - **Nombre**, visible por todas las clases
    - **Especialidad**, visible por todas las clases
    - **Número de cursos asignados**, no visible
- Al asignar un profesor a un curso, se incrementa el contador de número de cursos asignados del profesor.
- Se habilita la consulta de los datos de los profesores
- De los cursos se quiere almacenar:
    - Nombre del curso
    - Código del curso
    - Profesor
    - Aula
- Implementa
    - el constructor para los cursos, con todos sus atributos
    - método para asignar un profesor al curso
    - método para asignar una aula al curso
    - método para consultar los datos del curso, con el nombre del profesor y del curso.
- Crea una clase AppCentroEducativo para probar todos los requerimientos citados anteriormente.

Ejercicio 4. Diseña un sistema para gestionar productos, clientes y compras en una tienda.

- De los productos de la tienda se quiere almacenar:
    - Nombre, visible para cualquier clase
    - Código del producto, visible únicamente para las clases vecinas
    - Precio, no modificable directamente por ninguna otra clase
- Las acciones que se podrán realizar con los productos son :
    - Crear productos a partir de un nombre, código y precio
    - Mostrar la información de un producto
- De los clientes de la tienda se quiere definir:

    - Nombre del cliente, visible para cualquier clase
    - Identificador del cliente, no modificable directamente por ninguna otra clase
    - Saldo disponible, no modificable directamente por ninguna otra clase
- Las funciones disponibles para los clientes son:
    - Crear un cliente a partir de un nombre, identificador y saldo inicial
    - Mostrar datos del cliente
    - Actualizar el saldo tras realizar una compra. Si no hay suficiente saldo se informa al usuario y no se actualiza el importe.
- Por último, de las compras que se realizan en la tienda se quiere saber:
    - Producto comprado (código)
    - Cliente que realiza la compra (identificador)
    - Unidades de producto compradas
- Las operaciones sobre las compras son:
    - Crear una compra a partir de un cliente, producto y número de unidades
    - Procesar una compra:
        - Calcular el total de la compra.
        - Actualizar el saldo del cliente tras la compra. Si el saldo es insuficiente no se puede realizar la compra
    - Mostrar el detalle de la compra

Ejercicio 5. Diseña un sistema para gestionar vuelos, pasajeros y reservas en un aeropuerto.

- De los vuelos se quiere almacenar:
    - Código del vuelo
    - Origen y destino
    - Capacidad de pasajeros
    - Asientos disponibles
- Acciones sobre los vuelos:
    - Crear vuelos con código, origen, destino y capacidad.
    - Mostrar información del vuelo.
- De los pasajeros se quiere definir:
    - Nombre del pasajero
    - Identificador del pasajero
- Funciones sobre los pasajeros:
    - Crear un pasajero a partir de un nombre e identificador.
    - Mostrar datos del pasajero.
- De las reservas realizadas se quiere saber:

    - El vuelo reservado.
    - El pasajero que reserva
    - Número de asientos a reservar.
- Operaciones sobre las reservas:
    - Crear una reserva a partir de un vuelo, pasajero y número de asientos a reservar.
    - Mostrar los detalles de la reserva.
- Cuando se confirma una reserva se actualiza el número de asientos disponibles del vuelo relacionado.
- Antes de realizar una reserva se debe validar si hay asientos disponibles suficientes en el vuelo

Ejercicio 6. Diseña un sistema para gestionar mesas, clientes y órdenes en un restaurante.

- De las mesas se quiere almacenar:
    - Número de mesa
    - Capacidad de la mesa
    - Estado de la mesa (libre/ocupada)
- Acciones sobre las mesas:
    - Crear una mesa con número y capacidad.
- De los clientes se quiere definir:
    - Nombre del cliente
    - Identificador del cliente
    - Mesa asignada, inicialmente sin mesa.
- Funciones sobre los clientes:
    - Crear un cliente a partir de un nombre e identificador.
    - Mostrar datos del cliente.
    - Asignar una mesa a un cliente. Se considera que el cliente siempre tiene acompañantes que llenan la mesa asignada (no se realiza el control). La mesa asignada debe estar libre.
- Se actualiza el estado de una mesa a ocupada cuando es asignada a un cliente
- De los platos que se ofrecen se quiere saber:
    - Descripción
    - Precio
- Funciones sobre los platos : Mostrar datos de los platos
- De las órdenes realizadas se quiere saber:

    - Mesa que realiza la orden.
    - Cliente
    - Plato ordenado
- Operaciones sobre las órdenes:
    - Crear una orden a partir de un cliente, mesa y lista de platos.
    - Calcular el coste total de la orden: precio del plato * nº comensales
    - Mostrar los detalles de la orden.