Ejercicios de práctica — POO en JavaScript (nivel intermedio)

---

1) Gestión de empleados

Objetivo: practicar clase simple + método con cálculo + añadidos útiles.

Define la clase `Empleado` con:

- Propiedades: `nombre`, `salarioBase` (número), `añosServicio` (entero ≥ 0).
- Método: `calcularSueldo()` → devuelve `salarioBase` incrementado un **3% por año**.
- **Añadidos:**
  - `subirSueldo(cantidad)` → suma una cantidad fija al salario base (si `cantidad <= 0`, no hace nada).
  - `toString()` → "Empleado: | Base: X€ | Años: Y".

Tareas:

1. Crea 3 empleados con datos distintos.
2. Muestra por consola el resultado de `calcularSueldo()` de cada uno.
3. Sube 100€ al salario base del segundo empleado y vuelve a mostrar su sueldo calculado.
4. Imprime el `toString()` de todos.

---

2) Tienda online (herencia y polimorfismo)

Objetivo: herencia básica + polimorfismo + pequeño cálculo.

Define:

- `Producto` con `nombre`, `precio`, `stock` y método `mostrar()`.
- `Alimento extends Producto` con `fechaCaducidad`.
- `Electrodomestico extends Producto` con `garantia` (años).
- **Añadidos:**
  - En `Producto`, `aplicarDescuento(porcentaje)` → baja el **precio** un `%` (0–50). Si está fuera de rango, ignora.
  - En `Producto`, `enStock()` → `true` si `stock > 0`.
  - Sobrescribe `mostrar()` en cada subclase (incluye su extra).

Tareas:

1. Crea un `Alimento` y un `Electrodomestico`.
2. Llama a `mostrar()` de ambos (polimorfismo).
3. Aplica un 10% de descuento al `Alimento`.
4. Muestra si cada producto está en stock.

---

3) Registro de cursos y alumnos

Objetivo: arrays + pequeños métodos de utilidad.

Define la clase `Curso` con:

- `nombre`, `horas`, `alumnos` (array de strings).
- Métodos:
  - `agregarAlumno(nombre)` → añade si **no está vacío**.
  - `listarAlumnos()` → muestra cada nombre con numeración.
  - **Añadidos:**
    - `numAlumnos()` → total.
    - `tieneAlumno(nombre)` → `true/false` (búsqueda exacta, sin mayúsculas/minúsculas).

Tareas:

1. Crea un curso "JS POO" de 30 h.
2. Matricula 4 alumnos (incluye uno repetido para comprobar la búsqueda).
3. Lista los alumnos y muestra `numAlumnos()`.
4. Comprueba `tieneAlumno("pepa")` con mayúsculas/minúsculas cambiadas.

---

4) Catálogo multimedia

Objetivo: herencia + método sobrescrito + un auxiliar simple.

Define:

- `ElementoMultimedia` con `titulo`, `autor`, `año` y método `reproducir()`.
- `Pelicula` y `Cancion` que sobrescriben `reproducir()` con mensajes distintos.
- **Añadidos:**
  - `descripcion()` en la clase base → `"titulo — autor (año)"`.
  - En `Cancion`, `duracionMinutos` (número) y método `esLarga()` → `true` si ≥ 5 min.

Tareas:

1. Crea una `Pelicula` y una `Cancion`.
2. Llama a `reproducir()` de ambas.
3. Muestra `descripcion()` y, en el caso de canción, si `esLarga()`.

---

5) Análisis de objetos

Objetivo: recorridos + eliminación de propiedad.

Crea un objeto `producto` con `nombre`, `precio`, `stock`, `categoria`.

- Método `listarPropiedades()` → recorre con `for...in` y saca `clave: valor`.
- **Añadidos:**
  - `eliminarPropiedad(clave)` → borra con `delete` si existe.

- `propiedadesPropias()` → muestra **solo** `Object.getOwnPropertyNames(producto)`.

Tareas:

1. Lista todas las propiedades.
2. Elimina `categoria`.
3. Vuelve a listar con ambos enfoques.

---

6) Inventario de videojuegos

Objetivo: métodos de instancia + un estático sencillo.

Define la clase `Videojuego` con `titulo`, `plataforma`, `precio` y:

- `mostrar()` → imprime todos los datos.
- **Añadidos:**
  - `rebajar(euros)` → resta euros al **precio** (no menor de 0).
  - `subir(euros)` → suma euros.
  - `static contarPorPlataforma(lista, plataforma)` → número de juegos de esa plataforma.

Tareas:

1. Crea 4 juegos en un array.
2. Rebaja 5€ al primero y sube 2€ al segundo.
3. Muestra cuántos hay de "PC".

---

7) Figuras geométricas

Objetivo: polimorfismo numérico sencillo.

Define:

- `Figura` (base) con método `area()` que devuelva `0` por defecto.
- `Cuadrado` con `lado` y `area()` = `lado*lado`.
- `Circulo` con `radio` y `area()` = `Math.PI * radio * radio`.
- **Añadidos:**
  - En cada subclase, `perimetro()` correspondiente (cuadrado: `4*lado`; círculo: `2*Math.PI*radio`).

Tareas:

1. Crea 2 cuadrados y 2 círculos.
2. Mete todo en un array de `Figura` y recórrelo mostrando `area()` y `perimetro()`.
3. Calcula la **suma total de áreas** del array.

---

8) Biblioteca avanzada

Objetivo: arrays de objetos + pequeños filtros y ordenación.

Define:

- `Libro` con `titulo`, `autor`, `paginas` y `toString()`.
- `Biblioteca` con:
  - `libros` (array).
  - `agregar(libro)` → añade si `libro` tiene `titulo` y `autor` no vacíos.
  - `listarLibros()` → imprime cada libro usando `toString()`.
  - **Añadidos:**
    - `buscarPorAutor(nombre)` → devuelve array con coincidencias exactas.
    - `ordenarPorTitulo()` → ordena el array alfabéticamente.
    - `totalPaginas()` → suma de `paginas`.

Tareas:

1. Crea 5 libros y añádelos a una biblioteca.
2. Lista los libros.
3. Busca por autor “Asimov”.
4. Ordena por título y vuelve a listar.
5. Muestra `totalPaginas()`.