



# Ejercicios javascript

## Prácticas con P00

### Clases, objetos y métodos

---

#### Crear objetos

Dificultad: ●

1- Crea un objeto llamado auto que tenga algunas características como el color, marca, modelo y si está encendido o apagado. Crea los métodos necesarios para permitir encender y apagar el auto.

Output:

```
objeto.encender();  
objeto.apagar();
```

```
auto encendido  
El auto se apagó
```

#### Modelando clases

Cuenta bancaria - Dificultad: ●●

2-Escribe un programa que cree un objeto "cuenta" con las siguientes propiedades:

- Una propiedad titular con el valor "Alex".
- Una propiedad saldo, teniendo como valor inicial 0.
- Un método ingresar() que permita añadir dinero a la cuenta, pasando la cantidad como parámetro
- Un método extraer() que permita retirar la cantidad pasada como parámetro.
- Un método informar() que retorne la información del estado de la cuenta.

Utiliza este objeto para mostrar la descripción, ingresar y extraer dinero y volver a mostrar la descripción del estado de la cuenta.

Rectángulos

Dificultad: ●

3-Escribe una clase que permita crear distintos objetos “rectángulos”, con las propiedades de alto y ancho, mas los métodos necesarios para modificar y mostrar sus propiedades, calcular el perímetro y el área






*Producto - Dificultad:* ●●●

4- Escribe una clase Producto para crear objetos. Estos objetos, deben presentar las propiedades código, nombre y precio, además del método imprime datos, el cual escribe por pantalla los valores de sus propiedades.  
Posteriormente, cree tres instancias de este objeto y guárdalas en un array.  
Por último, utilice el método imprime datos para mostrar por pantalla los valores de los tres objetos instanciados.

*Generaciones - Dificultad:* ●●●

5- Crea una clase llamada Persona que siga las siguientes condiciones:  
Sus propiedades son: nombre, edad, DNI, sexo (H hombre, M mujer), peso y altura, año de nacimiento. Si quieres añadir alguna propiedad extra puedes hacerlo.  
Los métodos que se debe poder utilizar son:  
mostrarGeneracion: este método debe mostrar un mensaje indicando a qué generación pertenece la persona creada y cual es el rasgo característico de esta generación.  
Para realizar este método tener en cuenta la siguiente tabla de generaciones:

## TAXONOMÍA DE GENERACIONES

NOMBRE DE LA GENERACIÓN	MARCO TEMPORAL EN ESPAÑA	POBLACIÓN DE LAS GENERACIONES *	CIRCUNSTANCIA HISTÓRICA	RASGO CARACTERÍSTICO
<b>Generación Z</b>	1994 - 2010	7.800.000	Expansión masiva de internet	 Irreverencia
<b>Generación Y</b> <i>millennials</i>	1981 - 1993	7.200.000	Inicio de la digilitación	 Frustración
<b>Generación X</b>	1969 - 1980	9.300.000	Crisis del 73 y transición española	 Obsesión por el éxito
<b>Baby Boom</b>	1949 - 1968	12.200.000	Paz y explosión demográfica	 Ambición
<b>Silent Generation</b> Los niños de la posguerra	1930 - 1948	6.300.000	Conflictos bélicos	 Austeridad

**LA VANGUARDIA**

\* Datos correspondientes a la población residente en España. Fuente: INE, 2015.

esMayorDeEdad: indica si es mayor de edad, devuelve un mensaje indicando que la persona es mayor de edad.

mostrarDatos: devuelve toda la información del objeto.

generaDNI(): genera un número aleatorio de 8 cifras.

*Libros - Dificultad:* ●●●●


6- Crear una clase Libro que contenga al menos las siguientes propiedades:

- ISBN
- Título
- Autor
- Número de páginas

Crear sus respectivos métodos get y set correspondientes para cada propiedad. Crear el método mostrarLibro() para mostrar la información relativa al libro con el siguiente formato:

“El libro xxx con ISBN xxx creado por el autor xxx tiene páginas xxx”

Crear al menos 2 objetos libros y utilizar el método `mostrarLibro()`;  
Por último, indicar cuál de los 2 objetos "libros" tiene más páginas.

*Agenda telefónica - Dificultad:* 

7- Nos piden realizar una agenda telefónica de contactos.

Un contacto está definido por un nombre y un teléfono. Se considera que un contacto es igual a otro cuando sus nombres son iguales.

Una agenda de contactos está formada por un conjunto de contactos. Se podrá crear de dos formas, indicando nosotros el tamaño o con un tamaño por defecto (10).

Los métodos de la agenda serán los siguientes:

- `añadirContacto(Contacto)`: Añade un contacto a la agenda, sino la agenda no puede almacenar más contactos indicar por pantalla.
- `existeContacto(Contacto)`: indica si el contacto pasado existe o no.
- `listarContactos()`: Lista toda la agenda
- `buscarContacto(nombre)`: busca un contacto por su nombre y muestra su teléfono.
- `eliminarContacto(Contacto c)`: elimina el contacto de la agenda, indica si se ha eliminado o no por pantalla
- `agendaLlena()`: indica si la agenda está llena.
- `huecosLibres()`: indica cuántos contactos más podemos ingresar.

Crea un menú con opciones que serán seleccionadas por el usuario usando un prompt, las salidas de las operaciones seleccionadas por el usuario se pueden mostrar en pantalla y por consola.

### Ejercicios adicionales

8- Crea una clase llamada "Persona" que tenga las propiedades "nombre", "edad" y "profesión", y los métodos "saludar" y "despedirse". Luego, crea dos objetos de la clase "Persona" con diferentes valores para sus propiedades y llama a sus métodos "saludar" y "despedirse".

9- Crea una clase llamada "Animal" que tenga las propiedades "nombre" y "edad", y el método "emitirSonido". Luego, crea dos clases hijas llamadas "Perro" y "Gato" que hereden de "Animal" y tengan su propio método "emitirSonido". Finalmente, crea dos objetos, uno de la clase "Perro" y otro de la clase "Gato", y llama a sus métodos "emitirSonido" para verificar que cada animal emite el sonido adecuado.

10- crear una clase aeropuerto con las propiedades nombreAeropuerto y lista de aviones, esta clase deberá contener el método agregarAvion, el cual recibirá un objeto de tipo Avión, además de un método buscarAvion el cual recibirá el nombre de un avión y devolverá información en caso de encontrarlo, si no lo encontró indicar con un mensaje.

Por su parte los aviones tendrán las propiedades: nombre, capacidad, destino, lista de pasajeros. Los aviones tienen el método abordar el cual permite que un pasajero suba al avión solo si hay capacidad disponible en el mismo, caso contrario mostrar un mensaje que indique que el avión está lleno.

Crear un objeto de tipo aeropuerto llamado "Aeropuerto Internacional", crear 3 objetos aviones con diferentes destinos. Agregar los 3 aviones al aeropuerto, buscar un avión y usar el método abordar.