 IES ILÍBERIS Instituto de Educación Secundaria	2º DAW	DESARROLLO WEB EN ENTORNO CLIENTE UD1,2,3 Javascript
	Nombre: _____	Curso 2025-2026

1. Cree la funcionalidad necesaria para crear la **clase Factura** , en ella representaremos los datos de una Factura: numero, fecha, nombre_cliente, subtotal, iva y las líneas: (1,5 puntos)

fecha: Será de tipo Date.

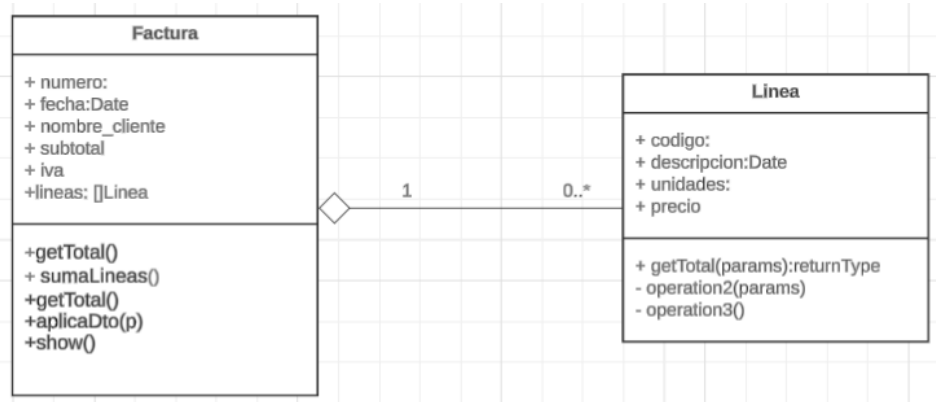
Líneas: Atributo que contendrá las líneas de la factura para, será un array de objetos de tipo línea.

Clase Línea: Representará cada línea de la factura y tendrá los atributos, codigo (representaría el código de producto), descripcion, unidades, pvp.


El método getTotal de la case Línea devolverá unidades * pvp

Añadiremos los siguientes métodos a la clase la **clase Factura**:

- Crearemos el constructor, métodos get y set para las propiedades, numero y fecha:
- sumaLineas: devolverá la suma del importe de las lineas de la factura
- getTotal: devolverá el total de la factura aplicando al subtotal (el suma líneas) el importe del iva
- show: mostrará toda la información de las factura, incluyendo las lineas de la misma, puede optar por mostrar la información por consola o por html.



Nos crearemos el método o función **demuestraUso** que generará un array de facturas con datos de ejemplo y posteriormente mostrará la información de las facturas.

 IES ILÍBERIS Instituto de Educación Secundaria	2º DAW	DESARROLLO WEB EN ENTORNO CLIENTE UD1,2,3 Javascript
	Nombre: _____	Curso 2025-2026

2. En este apartado trabajaremos con el array de facturas generado por `demuestraUso` y se pide que se implementen las siguientes funciones: (8 puntos)

- **`incrementaFactura(Factura[],num,incremento): Factura[]`** → Recibe un array de Facturas y devuelve un array de facturas con la factura número `num` incrementada todas sus líneas con el importe `incremento`.
- **`analizaFacturas(Factura[]): objAnálisis`** → Recibe un array de Facturas y devolverá un objeto con tres campos `numLineas`, `totalImporte`, `medialImporte`, `codigoArticuloMasVendido` que corresponden a número total de líneas, importe total de ventas, importe medio de ventas, código del artículo más vendido.
- **`eliminaFactura(Factura[],num): Factura[]`** → Recibe un array de Facturas y devuelve un array de Facturas donde no se encuentra la factura con número `num` pasado como parámetro.
- **`encuentraxCodigo(Factura[],cod): Factura[]`** → Recibe un array de Facturas y devuelve un array de Facturas donde solo se encuentran las facturas que tienen líneas con ese código de producto.
- **`filtraFacturas(Factura[],fechamin): -->`** → Recibe un array de Facturas y devuelve un array de Facturas donde las fecha son `>=` que `fechamin`
- **`modFacturas(Factura[])`** → devolverá un array de objetos `{numero, nombre_cliente, total}` donde `total` será el total de la factura incluido el iva. Se debe implementar con `map` y `destructuración`. (implementar de otra forma, solo en caso de no encontrar la solución con `map` y `destructuring`)
- **`ordenaFacturas(Factura[],orden)`** → Recibe un array de Facturas , y devuelve un array de Facturas ordenadas por número ascendente o descendente, dependiendo del parámetro `orden`. Si `orden` es 0 el orden es acendente, si `orden` es 1 el orden es descendente.
- **`updateFacturas(Factura[],objFactura)`** → Recibe un array de Facturas y actualiza el array con el `objFactura`, dos facturas son iguales si tienen el mismo número. La función devuelve -1 si el `objFactura` no se encuentra en el array de Facturas, y el numero de la factura en caso de que la modificación haya sido correcta

NOTA: Se valora Diseño, código estructurado, uso de `import/export`, comentarios.(0,5 puntos)