Actividades – Crear las siguientes clases con sus respectivos métodos y propiedades:

- Usar propiedades privadas y métodos privados cuando no sean necesarios.
- Cuando se devuelvan valores, si son objetos y arrays de las propiedades, devolver copias de los mismo, para evitar que puedan ser modificados.
- Incluye ejemplos de uso.

Calculadora (2,5 puntos)

- Propiedades
  - resultado (número), inicializa en 0
- Métodos
  - **sumar(numero)**: suma *numero* a resultado, y actualiza resultado con el valor de dicha operación
  - **restar(numero)**: suma *numero* a resultado, y actualiza resultado con el valor de dicha operación
  - **multiplicar(numero)**: suma *numero* a resultado, y actualiza resultado con el valor de dicha operación
  - **dividir(numero)**: divide *resultado* por numero, y actualiza resultado con el valor de dicha operación
  - **obtenerResultado()**: devuelve resultado
  - **reiniciar()**: pone resultado en 0
- Observaciones
  - arrojar un error cuando el argumento ingresado no sea un número, con Number.isFinite()
  - arrojar un error cuando se intenta dividir por 0

```
// Ejemplo de uso:
const calc = new Calculadora();
console.log(calc.sumar(10)); // 10
console.log(calc.restar(5)); // 5
console.log(calc.multiplicar(3)); // 15
console.log(calc.dividir(5)); // 3
console.log(calc.obtenerResultado()); // 3
calc.reiniciar(); // resultado vuelve a 0
```

Auto (2,5 puntos)

- Propiedades
  - encendido (booleano), inicializa en false
  - velocidad (número), inicializa en 0
  - marca (string)
  - modelo (número)
  - patente (string)
- Constructor
  - pide como argumentos marca, modelo, patente y los asigna a sus respectivas propiedades
- Métodos
  - **arrancar()** pone encendido en true
  - **apagar()** pone encendido en false
  - **acelerar()** suma 10 a velocidad y actualiza dicha propiedad
  - **desacelerar()** resta 10 a velocidad y actualiza dicha propiedad
  - **toString()** devuelve un *string* con la siguiente plantilla ${marca} ${modelo}, patente ${patente}
- Observaciones
  - sólo se puede acelerar o desacelerar si el auto se encuentra encendido
  - sólo se puede apagar el auto si la velocidad es 0
  - la velocidad mínima es 0

```
// Ejemplo de uso:
const miAuto = new Auto('Toyota', 2021, 'ABC123');
console.log(miAuto.toString()); // Toyota 2021, patente ABC123
miAuto.arrancar();
miAuto.acelerar();
console.log(miAuto.getVelocidad()); // 10
miAuto.desacelerar();
console.log(miAuto.getVelocidad()); // 0
miAuto.apagar();
```

Carrito con Productos (5 puntos)

Producto

- Propiedades
  - id (string)
  - nombre (string)
  - precio (número)
  - cantidad (número)
  - tieneImpuesto (booleano)
- Constructor
  - toma como argumentos nombre, precio, cantidad y tieneImpuesto y los asigna a sus respectivas propiedades
  - genera un id automático y lo asigna a su propiedad
- Métodos
  - getters para obteners los valores de todas las propiedades
  - setter para modificar el valor de cantidad
  - método privado para generar el id de manera automática
- Observaciones
  - la cantidad nunca puede ser 0
  - el precio no puede ser menor a 0

Carrito

- Propiedades
  - productos (array de Producto), inicializa vacío
- Métodos
  - **agregarProducto(producto)** agrega producto a `productos
  - **actualizarCantidadProducto(id, cantidad)** actualiza la cantidad del producto en productos con id
  - **eliminarProducto(id)** elimina de productos el producto con id
  - **calcularTotal()** devuelve el total del carrito (con impuestos incluidos)
  - **calcularImpuestoTotal()** devuelve el total de la suma de los impuestos de cada producto que tieneImpuesto
  - **obtenerCantidadTotal()** devuelve la cantidad total de ítems en el producto
  - **toString()** devuelve un string con
    - la lista de productos, detallando nombre, precio por unidad y cantidad
    - subtotal de todos los productos sin sumar impuesto
    - subtotal de todos los impuestos sumados
    - total final (suma de los subtotales)
- Observaciones
  - el impuesto es del 10% sobre el precio del producto

```
// Ejemplo de uso:
const producto1 = new Producto('Manzana', 1.5, 10, true);
const producto2 = new Producto('Pan', 2, 5, false);
const carrito = new Carrito();

carrito.agregarProducto(producto1);
carrito.agregarProducto(producto2);

console.log(carrito.toString());
console.log('Total con impuestos:', carrito.calcularTotal());
console.log('Total impuestos:', carrito.calcularImpuestoTotal());
console.log('Cantidad total de ítems:',
carrito.obtenerCantidadTotal());

carrito.actualizarCantidadProducto(producto1.getId(), 20);
console.log(carrito.toString());

carrito.eliminarProducto(producto2.getId());
console.log(carrito.toString());
```