Ejercicio de Examen: Gestión de Pedidos

Información General

Puntuación total: 10 puntos (sobre 20 total del examen)

ENTREGA

ej01_NombreApellido_gestionTienda.js

Descripción

Debes implementar las clases **`Articulo`**, **`ArticuloFisico`**, **`Pedido`** y **`TiendaOnline`** para que funcione correctamente el programa principal que se entrega. El programa se ejecutará con Node.js.

1. Clases Articulo y ArticuloFisico

1.1 Clase **`Articulo`** (1,5 puntos)

Crea una clase **`Articulo`** con las siguientes propiedades:

- **`id`** (number)
- **`nombre`** (string)
- **`precio`** (number)
- **`fechaEpochMs`** (number)

Métodos:

- El constructor: (1 puntos)
  - Recibe como parámetros **`id`**, **`nombre`** y **`precio`**, los asigna a sus propiedades correspondientes.
  - Llama al método **`setFechaEpochMs()`**.
- **`calcularPrecioFinal()`**:
  - Devuelve el precio aplicando un IVA del 21%.
- **`setFechaEpochMs()`**:
  - Utilizando la clase **`Date`**, almacena en la propiedad **`fechaEpochMs`** la fecha y hora actual medida en milisegundos desde el 1 de enero de 1970.

1.2 Clase 'ArticuloFisico (1 punto)'

Crea una clase **`ArticuloFisico`** que herede de **`Articulo`** con las siguientes características:

- Propiedad: **`pesoKg`** (tipo number).
- Método sobrescrito: **`calcularPrecioFinal()`** — debe calcular el precio final sumando 0,5 € por cada kilogramo de peso al precio y luego el IVA del 21%.
- Constructor: debe recibir **`pesoKg`** además de los mismos parámetros que el constructor de la clase **`Articulo`** y llamar al constructor de la clase padre

2 Clase Pedido (4 puntos)

Define una clase **`Pedido`** con las siguientes propiedades:

- **`id`** (string)
- **`nombreCliente`** (string)
- **`articulos`** (array de objetos **`Articulo`**)
- **`entregado`** (boolean)

Métodos:

- **`agregarArticulos(nuevosArticulos)`**:
  - Añade objetos tipo **`Articulo`** al array del pedido
- **`printArticulos()`** : Empleando el método **`map()`** o un bucle **`forEach`**, muestra por consola la información de todos los artículos del pedido.
  - Para cada artículo debe mostrarse la siguiente cadena de caracteres: '[<nombre> -#- <precioFinal> €]'
  - siendo <nombre> el nombre del artículo y <precioFinal> el resultado de llamar al método calcularPrecioFinal() para ese artículo.
- **`eliminarArticuloPorId(idArticulo)`**:
  - Busca el artículo por ID y lo elimina. **Debes usar los métodos `find()` y `splice()`.**
- **`calcularTotal()`**:
  - Calcula el total del pedido obteniendo el precio final de cada artículo. **Debes usar el método de array `reduce()`.**
- **`ordenarArticulosPrioritarios()`**: Ordena el array **`articulos`** siguiendo estos criterios:
  1. Los objetos **`ArticuloFisico`** deben aparecer primero que los **`Articulo`** base.
  2. Dentro de cada grupo, los artículos deben ordenarse de mayor a menor precio (descendente). **Debes usar el método de array `sort()` y el operador `instanceof`** así como una única función flecha callback para el método `sort() (1,5 pts)

1.3 Clase TiendaOnline (3,5 pts)

Define una clase **`TiendaOnline`** que gestione un array de pedidos. Debe incluir los siguientes métodos **además del *constructor** que inicializa el array de pedidos como vacío:

- **`addPedido(pedido)`**: Añade un pedido a la tienda. **Debe usar el método de array**.
- **`buscarPedidoPorId(id)`**: Devuelve el pedido correspondiente. **Debe usar el método de array `find()`.**
- **`procesarPedido(id)`**: Busca el pedido por ID, lo marca como entregado (true) y devuelve el importe total. **Importante: Debe usar métodos ya implementados siempre que se pueda evitar duplicar lógica.**