El RA que trabajaremos es:

RA4.- Programa código para clientes Web analizando y utilizando estructuras definas por el usuario

1.- Aplicación de música en Streaming

Se está desarrollando una aplicación para reproducir música en streaming, y se necesita tu colaboración para implementar las clases y funcionalidades relacionadas con la gestión de canciones y listas de reproducción. Este proyecto involucra el uso de **herencia**, **métodos estáticos**, **getters/setters**, **atributos privados**, y la aplicación de **Patrones de Diseño**.

Objetivo

Debes implementar dos clases principales, Cancion y ListaReproduccion, distribuidas en tres archivos: cancion.js, listaReproduccion.js, y app.js, además de un archivo HTML provisto (index.html). Las clases deben contener métodos para interactuar con los datos, aplicar validaciones y lanzar excepciones cuando sea necesario. Además, se debe aplicar un **Patrón de Diseño** adecuado para optimizar la arquitectura de la aplicación.

Requisitos:

1. Clase Cancion (cancion.js)

La clase Cancion debe tener los siguientes atributos y métodos:

- **Atributos privados**:
  - ✓ #titulo: Representa el título de la canción.
  - ✓ #artista: Representa el artista de la canción.
  - ✓ #numMeGusta: Número de "Me gusta" de la canción. Por defecto será 0.
  - ✓ #esPremium: Almacena si la canción es premium o no. Por defecto será false.

- **Métodos**:
  - ✓ **Constructor**: Recibe el título y el artista para inicializar los atributos. Los demás atributos tienen valores por defecto.
  - ✓ **Getters y Setters** para todos los atributos, con validación de tipos y restricciones:
    - titulo y artista deben ser cadenas no vacías.
    - numMeGusta debe ser un número no negativo.
    - esPremium debe ser un valor booleano. Si no se cumplen las restricciones, los setters deben lanzar una excepción.
  - ✓ **darMeGusta()**: Incrementa el atributo numMeGusta en 1.
  - ✓ **toString()**: Devuelve una cadena con el formato "Título - Artista".

- ✓ **Método estático obtenerTotalCanciones()**: Devuelve el número total de instancias de Cancion creadas.

2. Herencia:

Debes implementar una subclase de Cancion llamada CancionPremium que herede todos los atributos y métodos de Cancion. Esta subclase:

- ✓ Establece automáticamente el atributo esPremium en true.
- ✓ Sobrescribe el método toString() para indicar que la canción es premium.

3. Clase ListaReproduccion (listaReproduccion.js)

La clase ListaReproduccion debe gestionar una lista de canciones. Los atributos y métodos requeridos son:

- **Atributos privados**:
  - ✓ #nombre: Nombre de la lista de reproducción.
  - ✓ #canciones: Array de objetos Cancion, inicializado vacío.
- **Métodos**:
  - ✓ **Constructor**: Recibe el nombre de la lista y lo asigna.
  - ✓ **Getters y Setters** para el nombre, validando que no esté vacío.
  - ✓ **anadirCancion(cancion)**: Añade una instancia de Cancion al array #canciones.
  - ✓ **darMeGusta()**: Incrementa el número de "Me gusta" de todas las canciones en la lista.
  - ✓ **obtenerCancionesPremium()**: Devuelve un array con las canciones que sean premium.
  - ✓ **ordenarCanciones()**: Ordena las canciones de mayor a menor cantidad de "Me gusta".
  - ✓ **obtenerPrimerasCanciones()**: Devuelve las tres primeras canciones de la lista.
  - ✓ **toString()**: Devuelve una cadena con el nombre de la lista y las canciones contenidas.

4. Patrón de Diseño: Singleton para Gestionar Listas de Reproducción:

Implementa el **Patrón Singleton** en la clase ListaReproduccion para asegurarte de que solo exista una instancia de una lista de reproducción específica dentro de la aplicación. Esto garantiza que todas las partes de la aplicación interactúen con la misma instancia de la lista de reproducción.

5. Funcionalidades adicionales (app.js)

Impleméntalas en el archivo app.js:

- **anadirCanciones(lista)**: Recibe un objeto de ListaReproduccion y añade las siguientes canciones:
  - "First Light" de Yao Chen (Premium).
  - "Lekko" de Marcin Starosta (Premium).
  - "A Long Goodbye" de The Magic Lantern (1 "Me gusta").
  - "Alt jeg" de Elise Lindahl (2 "Me gusta").
  - "De seu" de Mirta da Silva (3 "Me gusta").

  Utiliza un bloque try-catch para capturar posibles excepciones.

- **obtenerListaHTML(canciones)**: Recibe un array de canciones y genera una cadena con código HTML en el siguiente formato:

```
<li>
        <div>
        <h3>Título: [Título de la canción]</h3>
        <p>Artista: [Artista de la canción]</p>
        <p>Número de "Me gusta": [Número de Me gusta]</p>
        <p>Premium: [Sí/No]</p>
        </div>
</li>
```

- Si la canción es premium, el párrafo Premium debe tener la clase CSS premium.

6. Tareas a realizar en el archivo app.js:

- Crea una nueva lista de reproducción llamada "Wake Up Gently" utilizando el **Patrón Singleton**, e invoca a anadirCanciones para añadir las canciones a esta lista.
- Muestra el nombre de la lista de reproducción en un elemento h1 con el id nombre-lista.
- Muestra las canciones de la lista en un elemento div con el id canciones-originales utilizando obtenerListaHTML.
- Muestra solo las canciones premium en el div con el id canciones-premium.
- Aumenta los "Me gusta" de todas las canciones de la lista usando el método darMeGusta(), y muestra las canciones actualizadas en el div con el id canciones-aumento-megusta.
- Ordena las canciones por número de "Me gusta" en orden descendente y muestra las tres primeras canciones en el div con el id canciones-mas-megusta.

- Imprime en consola el total de canciones creadas utilizando el método estático obtenerTotalCanciones().

Consideraciones

- **Patrón de Diseño (Singleton)**: Implementa el patrón Singleton en la clase ListaReproduccion para controlar la creación de instancias de listas de reproducción y asegurar que una lista específica tenga una única instancia en toda la aplicación.
- **Estructura Modular**: Organiza tu aplicación en los archivos cancion.js, listaReproduccion.js, y app.js usando módulos ES6.
- **Validaciones y Excepciones**: Asegúrate de lanzar excepciones cuando no se cumplan las restricciones en los atributos.
- **Métodos Estáticos**: Utiliza métodos estáticos para funcionalidades que no dependen de una instancia específica, como contar el total de canciones creadas.
- **Atributos Privados**: Emplea atributos privados para encapsular los datos y proteger la integridad de las clases.

Resultado:

Wake Up Gently

Canciones Originales

- **Título: First Light**

  Artista: Yao Chen

  Número de "Me gusta": 0

  Premium: Sí

- **Título: Lekko**

  Artista: Marcin Starosta

  Número de "Me gusta": 0

  Premium: Sí

- **Título: A Long Goodbye**

  Artista: The Magic Lantern

  Número de "Me gusta": 1

  Premium: No

- **Título: Alt jeg**

  Artista: Elise Lindahl

  Número de "Me gusta": 2

  Premium: No

- **Título: De seu**

  Artista: Mirta da Silva

  Número de "Me gusta": 3

  Premium: No

Canciones Premium

- **Título: First Light**

  Artista: Yao Chen

  Número de "Me gusta": 0

  Premium: Sí

- **Título: Lekko**

  Artista: Marcin Starosta

  Número de "Me gusta": 0

  Premium: Sí

Canciones con Aumento de Me Gusta

Las 3 Canciones con Más Me Gusta

- **Título: De seu**

  Artista: Mirta da Silva

  Número de "Me gusta": 4

  Premium: No

- **Título: Alt jeg**

  Artista: Elise Lindahl

  Número de "Me gusta": 3

  Premium: No

- **Título: A Long Goodbye**

  Artista: The Magic Lantern

  Número de "Me gusta": 2

  Premium: No

Números de canciones

Total de canciones creadas: 5