Ejercicios - En Clase

Registro Automotor

Objetivo: Desarrollar una aplicación en TypeScript que simule un Registro Automotor. La aplicación deberá gestionar una lista de vehículos mediante la implementación de clases y métodos que permitan realizar las operaciones básicas de un CRUD (Crear, Leer, Actualizar, Borrar).

1. **Clases Base y Herencia:**
   - Implementar la clase base `Vehiculo` que contendrá los atributos y métodos comunes a todos los vehículos.
   - Crear tres clases hijas que hereden de `Vehiculo`: `Auto`, `Moto` y `Camion`.
   - Cada clase hija debe implementar su propia versión del método `toString()`, que deberá devolver una cadena representativa del objeto con sus atributos principales.
2. **Composición:**
   - Implementar la clase `RegistroAutomotor`, que contendrá una lista de vehículos (composición).
   - La clase `RegistroAutomotor` debe permitir agregar, buscar, modificar y eliminar vehículos de su lista mediante métodos específicos:
     - `agregarVehiculo(vehiculo: Vehiculo): void`
     - `buscarVehiculo(patente: string): Vehiculo | undefined`
     - `modificarVehiculo(patente: string, datosActualizados: Partial<Vehiculo>): void`
     - `eliminarVehiculo(patente: string): void`
     - `mostrarVehiculos(): string` - Este método deberá utilizar el método `toString()` de cada vehículo para mostrar la lista completa de vehículos en el registro.
3. **Método `toString()` en Clases Hijas:**
   - En cada clase hija (`Auto`, `Moto`, `Camion`), implementar el método `toString()`, que retornará una representación en texto del vehículo, incluyendo detalles específicos de cada tipo.
4. **Persistencia de Datos**
   - Si el tiempo lo permite, considera la posibilidad de implementar una capa de persistencia simple, utilizando almacenamiento en txt o archivos JSON, para guardar y recuperar la lista de vehículos.