Advanced Vehicle Management System - C# (40% Final Project)

Objective: This project is designed to test students' understanding of **Object-Oriented Programming (OOP), Inheritance, Exception Handling, File Operations, Namespaces, Sorting Algorithms, and Logical Problem-Solving**using C#.

Marks Distribution (40%):
OOP & Inheritance - **10%**
Exception Handling - **5%**
File Operations (Saving & Loading) - **5%**
Object Arrays & Data Manipulation - **5%**
Sorting Algorithms & Logical Thinking - **10%**
Code Quality & Documentation - **5%**

□ Project Structure & Hierarchy

□ *VehicleManagementSystem*

```
|── □ Vehicles
|     |── Vehicle.cs (Abstract base class with shared properties)
|     |── Train.cs (Extends Vehicle, includes Units)
|     |── Airplane.cs (Extends Vehicle, includes Altitude)
|     |── CargoAirplane.cs (Extends Airplane, includes CargoCapacity)
|     |── Car.cs (Extends Vehicle, includes Model and Horsepower)
|     |── RaceCar.cs (Extends Car, includes TurboBoost)
|     |── Truck.cs (Extends Vehicle, includes LoadCapacity)
|     |── Boat.cs (Extends Vehicle, includes SeatingCapacity)
|     |── LuxuryYacht.cs (Extends Boat, includes Helipad)
|
|── □ IndependentClasses
|     |── VehicleComparer.cs (Implements Sorting & Comparison)
|     |── TaxCalculator.cs (Calculates vehicle tax based on type)
|     |── VehicleStatistics.cs (Analyzes average speed, price, etc.)
|
|── □ Exceptions
|     |── VehicleException.cs (Base Exception Class)
```

```
|   |── InvalidPriceException.cs (Thrown when price is negative)
|   |── InvalidSpeedException.cs (Thrown when speed is invalid)
|   |── InvalidCargoCapacityException.cs (Thrown for unrealistic cargo)
|
|── □ Services
|   |── FileHandler.cs (Handles file saving/loading)
|   |── VehicleManager.cs (Manages object array of vehicles)
|
|── Program.cs (Main execution logic)
```

Key Functionalities

Vehicle Base Class (Abstract)

- All vehicles share common properties:
    - string Name
    - double Price
    - double Speed
    - string VehicleType
- Implements:
    - virtual DisplayInfo() to print details.
    - abstract CalculateTax() to force child classes to define tax rules.
- Throws **custom exceptions** for invalid input values.

Multiple Inheritance Levels

- Vehicle → Car, Boat, Train, Airplane → Specialized Types
- **Example of Deep Inheritance:**
    - Airplane has Altitude
    - CargoAirplane extends Airplane and adds CargoCapacity
    - LuxuryYacht extends Boat and adds Helipad
    - RaceCar extends Car and has TurboBoost

Independent Utility Classes

VehicleComparer.cs (Sorting Mechanism)

- Implements **custom sorting algorithms**:
    - SortByPrice()
    - SortBySpeed()
    - SortByType()
- Students **must implement algorithm manually** instead of built-in methods.

TaxCalculator.cs

- **Different tax rates** for vehicle types.
    - Cars: **10% of price**
    - Airplanes: **15% of price**

    - Boats: **5% of price**
    - Trucks: **20% of price**
- Students **must override CalculateTax()** in child classes.

VehicleStatistics.cs

- Uses **LINQ** to:
    - Find **average price** of all vehicles.
    - Find **fastest vehicle** in each category.
    - Count the number of each **vehicle type**.

Exception Handling (Custom)

- Students **must implement**:
    - InvalidPriceException (Negative price not allowed)
    - InvalidSpeedException (Speed cannot be zero or negative)
    - InvalidCargoCapacityException (Cargo Airplane should have realistic limits)
- Proper **try-catch blocks** should be placed in the Program.cs file.

Object Array for Vehicle Storage

- Students must use an **array of objects (Vehicle[])**

- Object array must be **sorted** before displaying.
- Object array **should be populated from a file** and allow manual addition.

File Handling (Save & Load)

- **All vehicle objects** must be **saved to vehicles.txt**.
- **Loading mechanism** should read from the file and recreate objects.

Mandatory Programming Tasks (Logical Thinking & Sorting)

Students **must complete the following**:

Sorting Scenario (10%)

- Implement sort vehicles by **Price**.
- Implement vehicles by **Speed**.
- Implement a **custom comparator** to sort vehicles **alphabetically by type**.

LINQ Query Challenge (5%)

- Find all vehicles **faster than 200 km/h**.
- Find the **most expensive vehicle**.
- Find all **Trucks with Load Capacity > 5000kg**.

Exception Handling Challenge (5%)

- Write test cases that **intentionally throw custom exceptions**.
- Handle **negative speed, unrealistic cargo, and invalid prices**.

Data Analysis Challenge (5%)

- Calculate **average price of each vehicle type**.
- Count the **number of each vehicle type**.
- Display the **fastest vehicle in each category**.

Bonus Challenge

 Implement a **Graphical User Interface (Class diagram)**

 Add **a search function** where users can type a name and retrieve the vehicle's details.

 Final Notes

Code should be well-documented with comments.

Follow C# coding conventions and use meaningful variable names.

Test cases should be included for validation.

 Failure to use Object Arrays will result in a penalty!

 Submission Requirements

 Deadline: 13th April 2025

 Deliverables: Source code in a **GitHub repo**

Screenshot proof of **successful execution with sample outputs**