OOP Exam – 05.07.2025

Car Rental System

# Project Overview:

This project involves developing a console-based application, focusing on Object-Oriented Programming (OOP) principles. The application will serve as a Car Rental System, enabling users to manage vehicle inventory and rental transactions for a car rental agency.

## Key Objectives:

* Implement OOP concepts like encapsulation, inheritance, polymorphism, and abstraction.
* Create a user-friendly console interface.
* Ensure data persistence by reading and writing rental data to/from a CSV file.

## Car Data Management:

* Add Car: Input car details (ID, make, model, year, type, availability) and add them to the system.
* Edit Car: Modify existing car details (e.g., change availability, update info).
* Remove Car: Flag a car as no longer in the fleet.
* List Cars: Display a list of all available cars with their details.
* Rent Car: Assign a car to a customer (with name, ID, rental start date, expected return).
* Return Car: Mark a rented car as returned and update availability.

## Data Persistence:

* Read Data: Initialize the system with car data from a CSV file.
* Write Data: Save current car data to a CSV file upon exit.
* Search Functionality: Search for a car by model, ID, or status (available/rented).

## Technical Requirements:

* Classes like Car, Customer, Rental, and other relevant classes encapsulating respective properties and methods.
* Define interfaces for common functionalities (e.g., Rentable, Searchable).
* Use abstract classes where appropriate (e.g., for common vehicle types).

## File Handling:

* Implement CSV file operations for data persistence (**DO NOT USE LIBRARIES FOR READING CSV FILES**). You need to implement your logic to read and write into files.

## Exception Handling:

* Robust error and exception management for user inputs and file operations.

## Data Structures:

* Use appropriate collections (like Lists, Maps) to manage and organize data. Think about why and when to use different data structures!

## Documentation:

* Comprehensive documentation of classes, methods, and logic flow.
* Source Code: Well-organized and commented source files.
* Use git commit frequently, don’t just dump some code in the repo

## Evaluation Criteria:

* Adherence to OOP principles.
* Code quality and organization.
* Functionality and correctness.
* Error handling and validations.
* Documentation and ease of use.

**Each criterion gives you 20% of total points.**

**Example Usage of the Application /this is just a demo!!!, use your own architecture and files/:   
Example code is in Pseudocode.**

| **public class CarRentalApp {**  **public main() {**  **reader = new CarFileReader("cars.csv");**  **writer = new CarFileWriter("cars.csv");**  **service = new CarRentalService(reader, writer);**  **manager = new RentalManager(service);**  **PRINT("Welcome to the Car Rental System");**  **displayCommands();**  **isRunning = true;**  **while (isRunning) {**  **command = ReadLine();**  **isRunning = manager.execute(command);**  **// Add Car**  **// 1, Toyota Corolla, 2019, Sedan, Available**  **// Rent Car**  **// 1, John Doe, 2025-06-28**  **// Return Car**  **// 1**  **// Edit 1**  **// Toyota Corolla, 2019, Sedan, Rented**  **// List Cars**  **// Search Model Corolla**  **// Remove 1**  **// Save & Exit**  **}**  **}**  **}** |
| --- |
| **example csv**  **Id,Make,Model,Year,Type,Status,CurrentRenter**  **1,Toyota,Corolla,2019,Sedan,Available,**  **2,Ford,Focus,2020,Hatchback,Rented,Jane Smith**  **3,Audi,A4,2022,Sedan,Available,...** |

**Good luck and have fun!**