**Car Showroom Sales System – Project Explanation  
Name: Muhammad Waleed  
Roll Number: 24l-6023 2B-1**

**Introduction**

This project is an Object-Oriented Programming (OOP) based **Car Showroom Sales System** designed to automate the process of managing a car showroom using encapsulation , Composition / Aggregation & Inheritance . The system keeps track of available cars, allows customers to purchase vehicles, and maintains records of buyers and sellers. The goal is to create a structured and efficient way to handle car sales while ensuring proper documentation of transactions.

**Class Structure and Functionality**

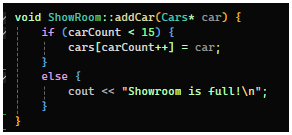
The project follows an OOP approach, where different functionalities are divided into separate classes for better organization. Here’s an overview of the main classes:

**1. ShowRoom Class**

This is the main class that represents the car showroom itself. It stores details like the **name of the showroom, address, contact number, email, and sales tax registration number**. It also maintains an array of cars (up to 15 cars in this implementation).

**Key Functions: (for car class)**

* addCar(Cars\* car): Adds a new car to the showroom.



* showCars(): Displays all available cars.

A computer code on a black background

AI-generated content may be incorrect.

* getCar(int index): Retrieves a car based on its index.

A computer code on a black background

AI-generated content may be incorrect.

* sellCar(int index): Removes a car from inventory when it is sold and generates a receipt.

A screen shot of a computer code

AI-generated content may be incorrect.

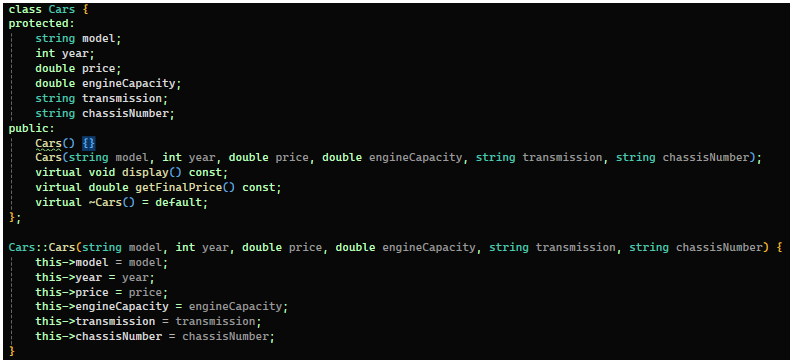
* generateReceipt(Cars\* car, const Buyer& buyer): Generates a receipt for a sold car, including buyer details.

A black background with yellow and green text

AI-generated content may be incorrect.

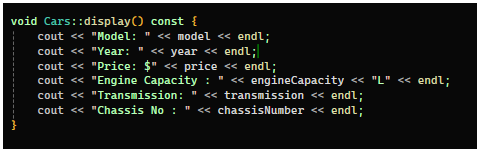
**2. Cars Class**

This class represents the cars in the showroom. Each car has the following attributes:

* **Model Name**
* **Year of Manufacture**
* **Price**
* **Engine Capacity**
* **Transmission Type (Manual/Automatic)**
* **Chassis Number**
* **Registration Number (for used cars)**
* **Auction Date and Price (for imported cars)**

**Key Functions:**

* display(): Shows all the details of a car.



* getFinalPrice(): Calculates the final price of the car, taking into account withholding tax if applicable.

A screen shot of a computer code

AI-generated content may be incorrect.

**3. Buyer Class**

This class stores information about the buyer of the car. It includes:

* **Buyer’s Name**
* **CNIC (National ID number)**
* **Contact Information**

**Key Function:**

* displayBuyerInfo(): Displays the buyer’s information on the receipt.

**4. Seller Class**

This class is for customers who want to sell their cars to the showroom. It stores seller details and allows them to sell their used car to the showroom, which is then added to the inventory.

**Key Function:**sellCar(): Takes car details from the seller and creates a new car object to be added to the showroom.

A computer code on a black background

AI-generated content may be incorrect.

**Implementation Details**

**Adding a Car to the Showroom**

When a new car arrives at the showroom, it is added using the addCar() function. The system ensures that the showroom does not exceed its limit of 15 cars.

A computer screen with text and stars

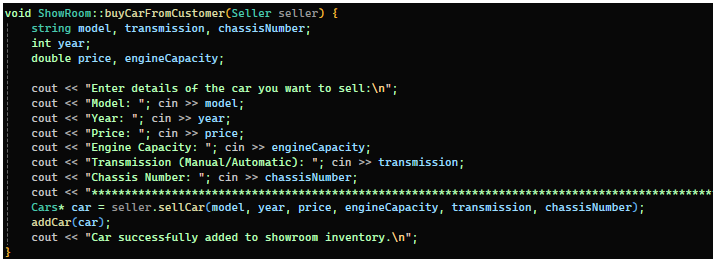
AI-generated content may be incorrect.

**Displaying Cars**

Customers can view available cars using the showCars() function. This prints the details of all cars currently in the showroom.

**Selling a Car**

When a customer decides to buy a car, the system does the following:

1. It retrieves the car using getCar(index).
2. The car's details are displayed along with the price.
3. The buyer’s details are collected and stored.
4. A receipt is generated using generateReceipt().
5. ****The car is removed from the showroom inventory.

**Buying a Car from a Seller**

The showroom also buys cars from customers using the buyCarFromCustomer(Seller seller) function. The seller provides car details, and if accepted, the car is added to the inventory.

**Challenges Faced & How They Were Solved**

**1. Managing Memory for Dynamic Objects**

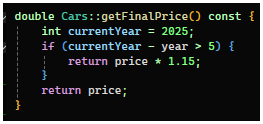
* Since we store cars as pointers (Cars\* cars[15]), we needed to ensure proper memory management. To solve this, we:  
   1. Used delete to free memory after selling a car.  
   2. Shifted the remaining elements in the array after a sale.

**2. Handling Buyer Details**

Initially, we were not storing buyer information properly. We solved this by passing a Buyer object to generateReceipt(), ensuring that buyer details appear on the final receipt.

**3. Implementing Withholding Tax for Older Cars**

We had to check if a car was older than 5 years to avoid applying the 15% withholding tax. We handled this in getFinalPrice(), where we conditionally apply the tax based on the car's age.



**User Centered Design (Interface outputs)**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Conclusion**

This project successfully simulates a **car showroom management system** with essential functionalities like adding cars, displaying available cars, selling cars to customers, and purchasing cars from sellers. The use of **OOP principles** made the system well-structured and easier to maintain.