# Car Tracker Application (with REST API)

## Overview

The Car Tracker system is built using ASP.NET Core Web API and a C# console client. It enables users to manage car records through a RESTful interface, with persistent storage handled via JSON files.

The Car Tracker allows users to:  
- Add new cars  
- View cars by manufacturing year  
- Remove cars   
- Saves data to a json file.

## Key Concepts Used

- Object-Oriented Programming (OOP):  
 - Encapsulation: Logic is encapsulated in classes (Car, Tasks, Events, EventHandlers).  
 - Inheritance: Tasks inherits from Events to reuse event functionality.  
 - Polymorphism: Event handling allows different responses to the same event trigger.

-Collections & Generics:  
 -List is used to store and manage car objects (List<Car>).

- Events and Delegates:  
 - Custom event Notification is defined and triggered using startEvent.  
  
- Exception Handling:  
 - try-catch blocks are used to handle invalid user input gracefully.

- Serialization & FileI/O:  
 - Car data is saved to a .json file using System.Text.Json.

- REST Api Integration:  
 - Standard HTTP methods like GET, POST, PUT, and DELETE are used to add, update, delete and retrieve data.

- Async and Await:  
 - Used to perform non-blocking HTTP operations.  
  
- Separation of Concerns:  
 - UI logic is in Main().  
 - Business logic is in Tasks.  
 - Event logic is in Events and EventHandlers.  
 - Client layer(CarApiClient) to interact with the API using HTTPClient.  
 - API Layer (CarsController) to handle HTTP requests and responses.   
 - Data Access Layer (CarFileReader) to manage reading from and writing to a json file.

### REST API & Client Integration:

API Endpoints (CarsController)

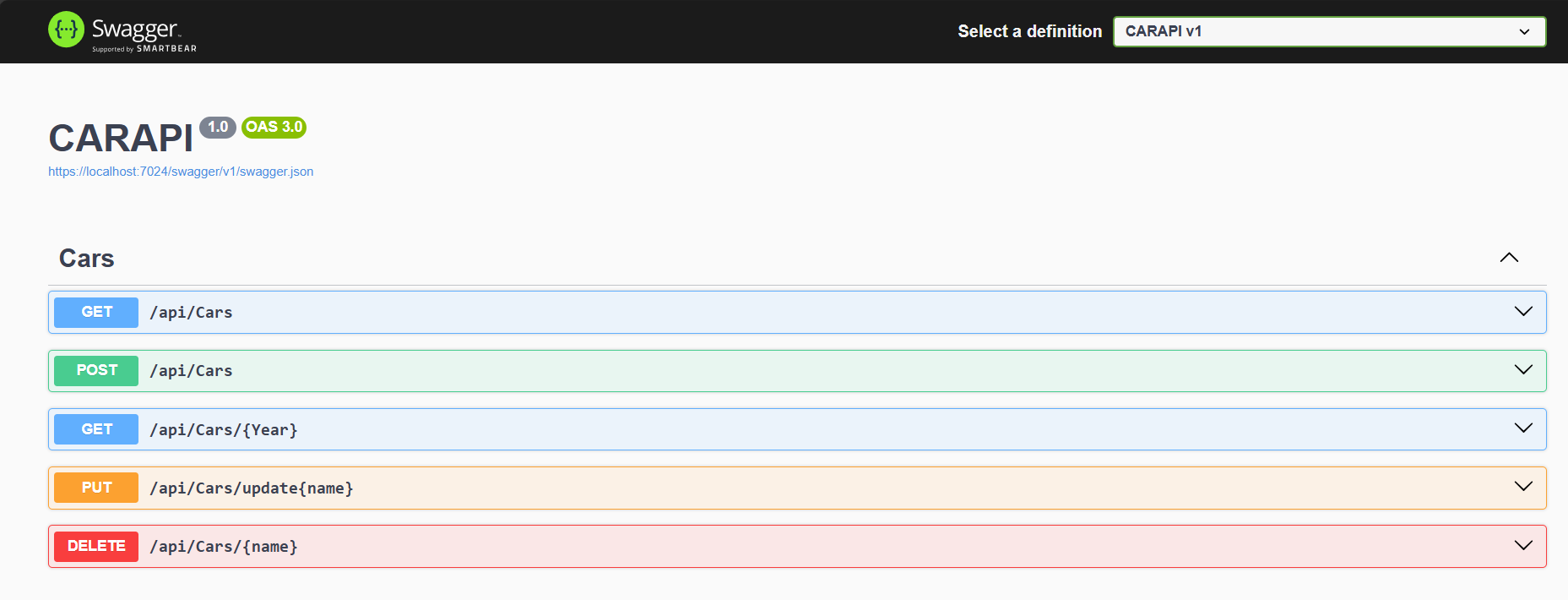
* GET /api/cars  
  Retrieves all car records from the JSON file.
* GET /api/cars/{year}  
  Filters and returns cars in the specified year.
* POST /api/cars  
  Adds a new car to the system and updates the JSON file.
* PUT /api/cars/update{name}  
  Updates details of a car identified by its name.
* DELETE /api/cars/{name}  
  Removes a car from the system by name.

Client Application (CarApiClient)

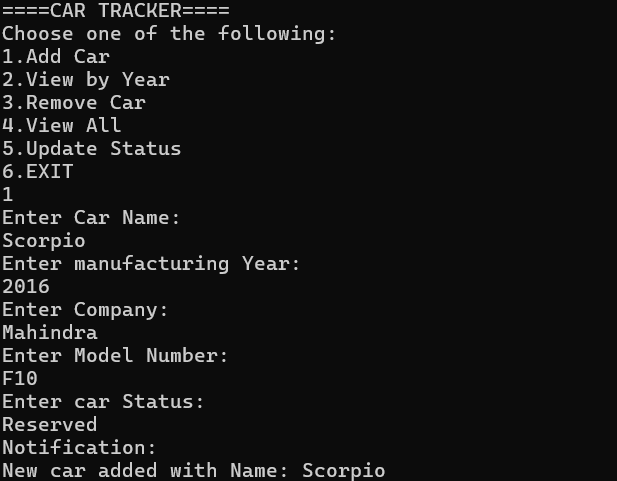
The console-based client interacts with the API using HttpClient and provides a user-friendly interface to:

* Add new cars
* View cars by manufacturing year
* Remove cars by name
* View all cars
* Update car status

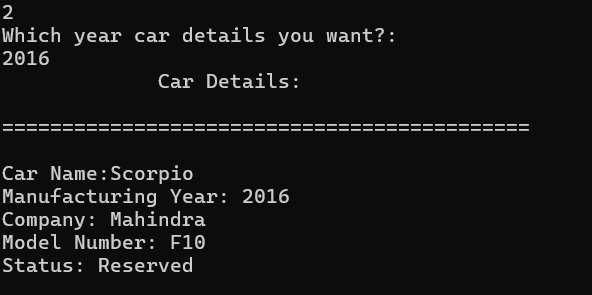
### Output Screenshots:



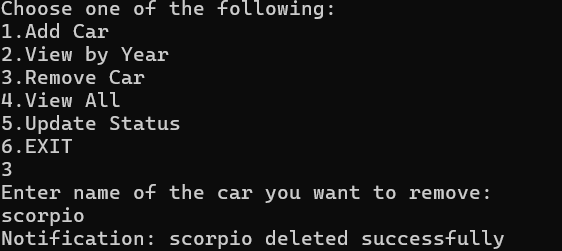
### Add a car:



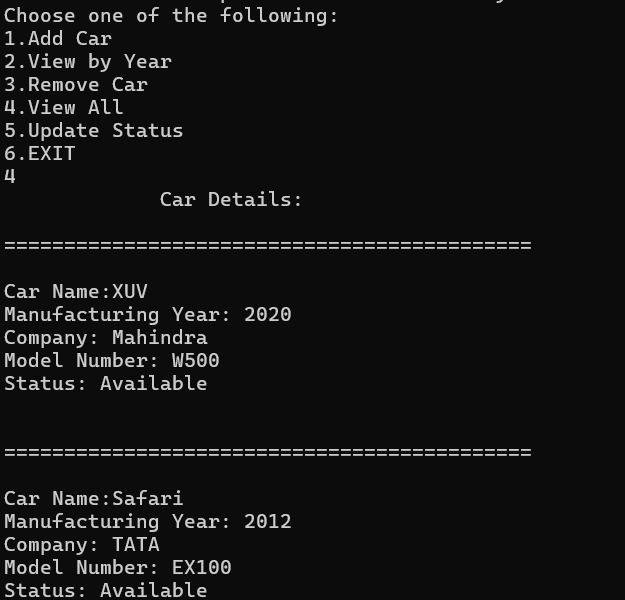
### View by Year:

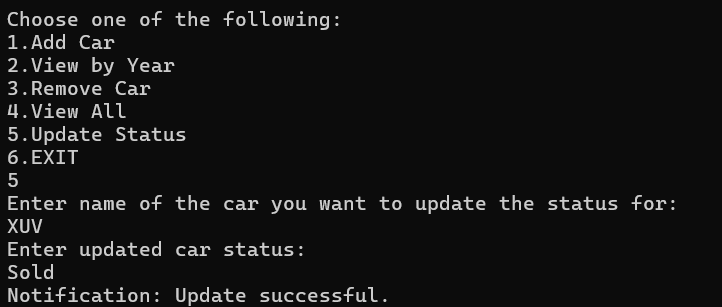


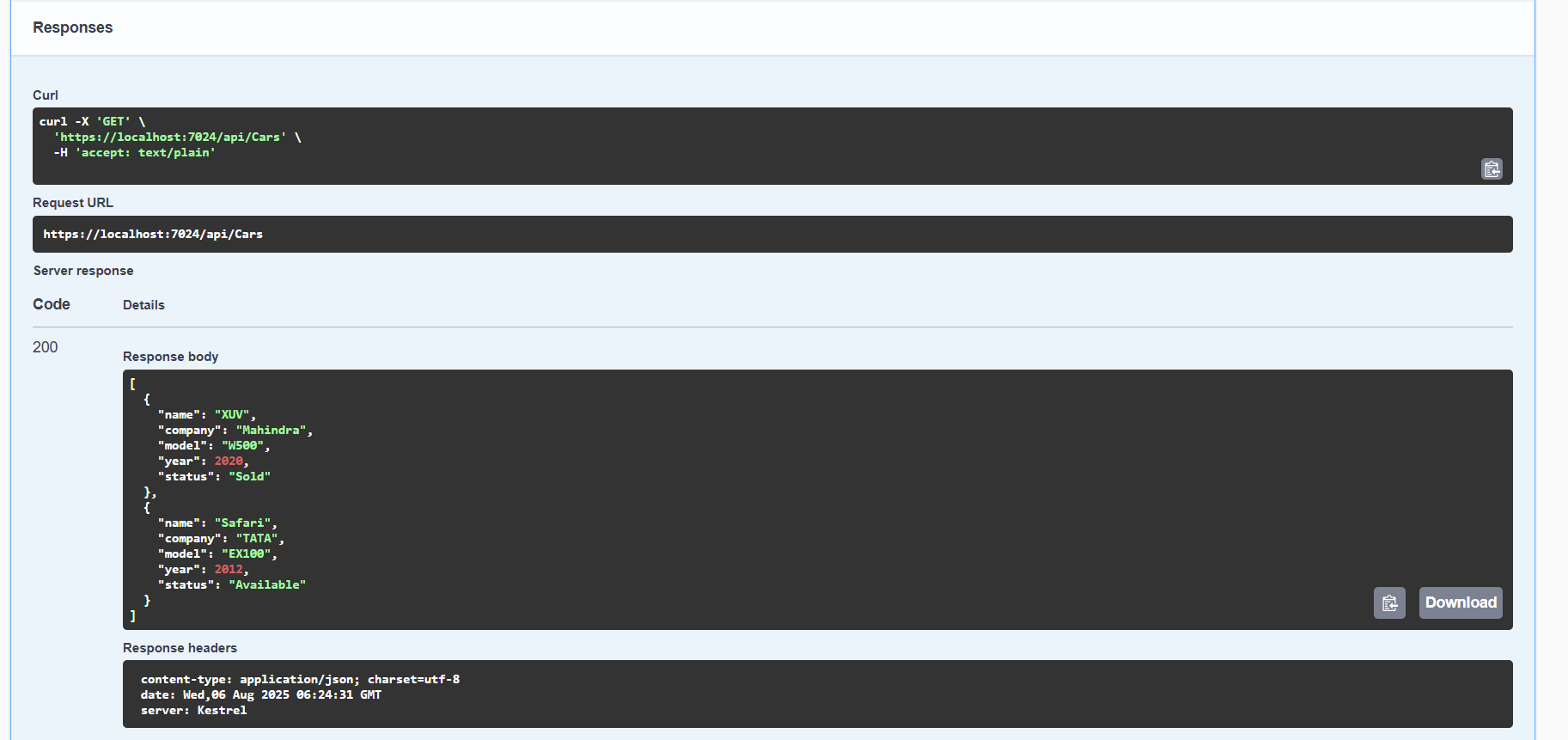
### Remove Car:



### View All:



Update Status: 



### Saved Data:



### Code:

### CARAPI:

#### CarsController

using CARAPI;

using Microsoft.AspNetCore.Mvc;

[ApiController]

[Route("api/[controller]")]

public class CarsController : ControllerBase

{

[HttpGet]

public ActionResult<IEnumerable<Car>> Get()

{

var cars = CarFileHelper.LoadCars();

return Ok(cars);

}

[HttpGet("{Year}")]

public ActionResult<Car> Get(int Year)

{

var cars = CarFileHelper.LoadCars();

var car = cars.Where(c => c.Year == Year);

if (car == null) return NotFound();

return Ok(car);

}

[HttpPost]

public ActionResult<Car> Post(Car car)

{

var cars = CarFileHelper.LoadCars();

cars.Add(car);

CarFileHelper.SaveCars(cars);

return CreatedAtAction(nameof(Get), new { year = car.Year }, car);

}

[HttpPut("update{name}")]

public IActionResult Put(string name, Car updatedCar)

{

var cars = CarFileHelper.LoadCars();

var car = cars.FirstOrDefault(c => c.name.Equals(name, StringComparison.OrdinalIgnoreCase));

if (car == null) return NotFound();

car.name=updatedCar.name;

car.company = updatedCar.company;

car.Model = updatedCar.Model;

car.Year = updatedCar.Year;

car.status=updatedCar.status;

CarFileHelper.SaveCars(cars);

return Content($"{name} updated successfully");

}

[HttpDelete("{name}")]

public IActionResult Delete(string name)

{

var cars = CarFileHelper.LoadCars();

var car = cars.FirstOrDefault(c => c.name.Equals(name, StringComparison.OrdinalIgnoreCase));

if (car == null) return NotFound();

cars.Remove(car);

CarFileHelper.SaveCars(cars);

return Content($"{name} Deleted Successfully");

}

}

#### CarFileHelper:

using CARAPI;

using System.Text.Json;

public static class CarFileHelper

{

private static readonly string filePath = @"C:\Users\GBO3KOR\source\repos\CARAPI\CARAPI\cars.json";

public static List<Car> LoadCars()

{

if (!File.Exists(filePath)) return new List<Car>();

var json = File.ReadAllText(filePath);

return JsonSerializer.Deserialize<List<Car>>(json) ?? new List<Car>();

}

public static void SaveCars(List<Car> cars)

{

var json = JsonSerializer.Serialize(cars, new JsonSerializerOptions { WriteIndented = true });

File.WriteAllText(filePath, json);

}

}

#### Car.cs:

namespace CARAPI

{

public class Car

{

public string name { get; set; }

public string company { get; set; }

public string Model { get; set; }

public int Year { get; set; }

public string status { get; set; }

}

}

### CarApiClient

#### Program.cs

using System;

using System.Net.Http;

using System.Net.Http.Json;

using System.Runtime.ConstrainedExecution;

using System.Threading.Tasks;

namespace CarApiClient

{

class Program

{

static async Task Main(string[] args)

{

Car c = new Car();

EventHandlers eventHandlers = new EventHandlers();

c.Event+=eventHandlers.handleNotificationEvent;

//Since car inherits tasks and tasks inherit Events class,

//we can directly use car object to access Events class parameter or methods.

Console.WriteLine("====CAR TRACKER====");

try

{

int option = 0;

while (option!=6)

{

try

{

Console.WriteLine("Choose one of the following:\n1.Add Car\n2.View by Year\n3.Remove Car\n4.View All\n5.Update Status\n6.EXIT");

option = int.Parse(Console.ReadLine());

}

catch (Exception e)

{

Console.WriteLine("Please eneter a valid number between 1 to 6 :)");

}

switch (option) {

case 1: //Insert a car //POST

await c.insert();

break;

case 2: //Get car by Year

Console.WriteLine("Which year car details you want?: ");

int y = int.Parse(Console.ReadLine());

await c.GetCarsByYear(y);

break;

case 3: //Delete Car

Console.WriteLine("Enter name of the car you want to remove: ");

String cname = Console.ReadLine();

await c.removeCar(cname);

break;

case 4: //View All

await c.details();

break;

case 5: //update //put

Console.WriteLine("Enter name of the car you want to update the status for:");

string carName = Console.ReadLine();

Console.WriteLine("Enter updated car status:");

String cstatus= Console.ReadLine();

await c.updateStatus(cstatus, carName);

break;

case 6:

return;

default: Console.WriteLine("Invalid option");break;

}

}

// GET all cars

// var cars = await client.GetFromJsonAsync<Car[]>("api/cars")

/\*

// POST a new car

var newCar = new Car { Make = "Ford", Model = "Focus", Year = 2022 };

var response = await client.PostAsJsonAsync("api/cars", newCar);

if (response.IsSuccessStatusCode)

{

var createdCar = await response.Content.ReadFromJsonAsync<Car>();

Console.WriteLine($"\nNew car created with ID: {createdCar.Id}");

}

else

{

Console.WriteLine("Failed to create car.");

}

\*/

}

catch (Exception ex)

{

Console.WriteLine($"Error: {ex.Message}");

}

}

}

}

Tasks.cs:

using CarApiClient;

using System;

using System.Collections;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Net.Http;

using System.Net.Http.Json;

using System.Net.NetworkInformation;

using System.Runtime.ConstrainedExecution;

using System.Text;

using System.Text.Json;

using System.Threading.Tasks;

using System.Threading.Tasks;

using System.Xml.Linq;

namespace CarApiClient

{

public class Tasks : Events

{

string msg = "";

public static HttpClient connect()

{

HttpClient client = new HttpClient();

client.BaseAddress = new Uri("https://localhost:7024/");

return client;

}

public async Task insert()

{

HttpClient client = connect();

var cars = client.GetFromJsonAsync<Car[]>("api/cars");

Console.WriteLine("Enter Car Name: ");

string name = Console.ReadLine();

Console.WriteLine("Enter manufacturing Year: ");

int year = int.Parse(Console.ReadLine());

Console.WriteLine("Enter Company: ");

string company = Console.ReadLine();

Console.WriteLine("Enter Model Number: ");

string model\_number = Console.ReadLine();

Console.WriteLine("Enter car Status: ");

string status = Console.ReadLine();

var newCar = new Car { name=name, Year=year, company=company, status = status, Model=model\_number };

var response = await client.PostAsJsonAsync("api/cars", newCar);

if (response.IsSuccessStatusCode)

{

var createdCar = await response.Content.ReadFromJsonAsync<Car>();

msg = $"\nNew car added with Name: {createdCar.name}";

}

else

{

msg = $"Failed to Add car.";

}

startEvent(msg);

client.Dispose();

} //Working

public async Task GetCarsByYear(int year) //not working

{

HttpClient client = connect();

var cars = await client.GetFromJsonAsync<Car[]>($"api/cars/{year}");

if (!cars.Any())

{

string msg = $"Zero cars present for the entered year:{year}"; //Use of Event

startEvent(msg);

}

else

{

Console.WriteLine(" Car Details: ");

foreach (var car in cars) { car.display(); }

}

client.Dispose();

}

public async Task removeCar(string name)

{

HttpClient client = connect();

var cars = client.GetFromJsonAsync<Car[]>("api/cars");

var response = await client.DeleteAsync($"api/cars/{name}");

if (response.IsSuccessStatusCode)

{

msg=$"{name} deleted successfully";

}

else

{

msg="Failed to Delete car as it is not found.";

}

startEvent(msg);

client.Dispose();

}

public async Task details()

{

HttpClient client = connect();

var cars = await client.GetFromJsonAsync<Car[]>("api/cars");

if (!cars.Any())

{

string msg = "There are no cars present in the inventory";

startEvent(msg);

}

else

{

Console.WriteLine(" Car Details: ");

foreach (var c in cars)

{

c.display();

}

}

client.Dispose();

}

public async Task updateStatus(string status, string carName)

{

HttpClient client = connect();

var cars = await client.GetFromJsonAsync<Car[]>("api/cars");

var updatedcar = cars.FirstOrDefault(c => c.name.Equals(carName, StringComparison.OrdinalIgnoreCase));

bool update = true;

if (updatedcar == null)

{

msg = "No car found with the given name to update the status";

}

else

{

string stat = status.ToLower();

switch (stat)

{

case "sold":

case "reserved":

case "available":

if (!updatedcar.status.Equals(status, StringComparison.OrdinalIgnoreCase))

{

updatedcar.status = status;

}

else

{

msg = $"Car is already {status}";

update = false;

}

break;

default:

msg = "Invalid status code";

update = false;

break;

}

}

if (update != false)

{

var response = await client.PutAsJsonAsync<Car>($"api/cars/update{carName}", updatedcar);

if (response.IsSuccessStatusCode)

{

msg="Update successful.";

}

else

{

msg=$"Update failed. Status code: {(int)response.StatusCode} - {response.ReasonPhrase}";

}

}

startEvent(msg);

}

}}

#### Events.cs

namespace CarApiClient

{

public class Events

{

//Delegates adn Events

public delegate void notify(string message);

public event notify Event;

public void startEvent(string message)

{

Event?.Invoke(message);

}

}

}

#### EventHandlers.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CarApiClient

{

public class EventHandlers

{

public void handleNotificationEvent(string msg)

{

Console.WriteLine("Notification: "+msg);

}

}

}

#### Car.cs:

using System.Net.NetworkInformation;

namespace CarApiClient

{

public class Car:Tasks

{

public string name { get; set; }

public string company { get; set; }

public string Model { get; set; }

public int Year { get; set; }

public string status { get; set; }

public void display()

{

Console.WriteLine("\n============================================\n");

Console.WriteLine($"Car Name:{name}\nManufacturing Year: {Year}\nCompany: {company}\nModel Number: {Model}\nStatus: {status}\n");

}

}

}