# Lab: Defining Classes

Tasks for exercise in class and for homework to the course ["Programming Advanced for QA" @ SoftUni](https://softuni.bg/trainings/4257/programming-advanced-for-qa-november-2023)

Test your tasks in the Judge system: <https://judge.softuni.org/Contests/4485>

## Car

Create a public **class** named Car within the namespace CarManufacturer:

|  |
| --- |
| Car.cs |
| namespace CarManufacturer  {  class Car  {  // TODO: define the Car class members here …  }  } |

Define in the above class **private** **fields** for:

* make: string
* model: string
* year: int

The class should also have **public** **properties** for:

* Make: string
* Model: string
* Year: int

Create a public class StartUp class within the same namespace CarManufacturer to hold your program’s entry point:

|  |
| --- |
| StartUp.cs |
| namespace CarManufacturer  {  public class StartUp  {  static void Main()  {  // TODO: define the Main() method here ...  }  }  } |

You should be able to use your Car class like this:



## Car Extension

**NOTE**: You need a StartUp class with the namespace CarManufacturer.

Create a class Car (you can use the **class** from **the previous task**).

The class should have private fields for:

* make: string
* model: string
* year: int
* fuelQuantity: double
* fuelConsumption: double

The class should also have properties for:

* Make: string
* Model: string
* Year: int
* FuelQuantity: double
* FuelConsumption: double

The class should also have methods for:

* Drive(double distance): void – This method checks if the car fuel quantity minus the distance multiplied by the car fuel consumption is bigger than zero. If it is removed from the fuel quantity, the result of the multiplication between the distance and the fuel consumption. Otherwise, write on the console the following message:

"Not enough fuel to perform this trip!".

* WhoAmI(): string – returns the following message:

"Make: {this.Make}

Model: {this.Model}

Year: {this.Year}

Fuel: {this.FuelQuantity:F2}"

You should be able to use the class like this:



## Car Constructors

Using the class from the previous problem create one parameterless constructor with default values:

* Make – VW
* Model – Golf
* Year – 2025
* FuelQuantity – 200
* FuelConsumption – 10

Create a second constructor accepting make, model and year upon initialization and call the base constructor with its default values for fuelQuantity and fuelConsumption.



Create a third constructor accepting make, model, year, fuelQuantity and fuelConsumption upon initialization and reuse the second constructor to set the make, model and year values.



Go to **StartUp.cs** file and make 3 different instances of the **class Car**, using the **different** overloads of the constructor.



## Car Engine and Tires

Using the Car class, you already created, define another class Engine.

The class should have private fields for:

* horsePower: int
* cubicCapacity: double

The class should also have properties for:

* HorsePower: int
* CubicCapacity: double

The class should also have a constructor, which accepts **horsepower** and **cubicCapacity** upon initialization:



Now create a class Tire.

The class should have private fields for:

* year: int
* pressure: double

The class should also have properties for:

* Year: int
* Pressure: double

The class should also have a constructor, which accepts **year** and **pressure** upon initialization:



Finally, go to the Car class and create **private fields** and **public properties** for **Engine** and **Tire[]**.Create another constructor, which accepts make, model, year, fuelQuantity, fuelConsumption, Engine and Tire[] upon initialization:



You should be able to use the classes like this:

