* Types we defined:
* ConsoleUI Types:

|  |  |  |
| --- | --- | --- |
| **Type** | **Type Name** | **Description** |
| Class | Program | A class containing a Main function which serves as an entry point to the program |
| Class | MainMenu | A class (the only class) for communicating with the user. It Operates the entire garage by getting commands from the user and executing them. It serves as a link between the user and the garage logic. |
| Class | Messages | A class aggregating every message that should be delivered to the user. |

* GameLogic Types:

|  |  |  |
| --- | --- | --- |
| **Type** | **Type Name** | **Description** |
| Abstract Class | Vehicle | An abstract class representing a vehicle. It doesn't have any abstract methods but we don't want any instances of it – only of it's extending classes. It aggregates the information and logic which every type of vehicle have in common, and is used for polymorphism |
| Class | Car | A class extending Vehicle. It adds information and logic which are unique to cars on top of the Vehicle info and logic it inherits |
| Class | Truck | A class extending Vehicle. It adds information and logic which are unique to trucks on top of the Vehicle info and logic it inherits |
| Class | Motorcycle | A class extending Vehicle. It adds information and logic which are unique to motorcycles on top of the Vehicle info and logic it inherits |
| Class | Wheel | A class representing a wheel object. Every vehicle should have a list of wheels. It aggregates information and logic relevant to wheels |
| Class | VehicleOwner | A class aggregating information about the owner of a vehicle. Currently it only holds 2 fields but we decided to write it anyway for the sake of extending information about an owner in the future (adding his address for example) |
| Abstract Class | Engine | An abstract class representing an engine. It aggregates information and logic concerning every type of engine and is used for polymorphism. Every Vehicle has an engine object via composition |
| Class | FuelEngine | A class extending Engine. It adds information and logic which are unique to fuel engines on top of the Engine info and logic it inherits |
| Class | ElectricEngine | A class extending Engine. It adds information and logic which are unique to electric engines on top of the Engine info and logic it inherits |
| Class | VehicleFactory | A class which provides functionality for creating vehicles and setting their properties. If we want to support more vehicles in our garage, we only need to change this class accordingly |
| Class | GarageManager | A class in charge of managing our garage. It holds a dictionary of all the vehicles we've inserted and provides functionality for changing their state |
| Class | Validation | A class for validating and parsing information which is passed from the user by the system's UI before we use it in our logic. It throws exceptions in cases of invalid input |
| Class | ValueOutOfRangeException | A class extending Exception. A new ValueOutOfRangeException is thrown when an input exceeds the allowed range. |
| Enum | eFuelType | An Enum representing the different types of fuel an engine could run on |
| Enum | eColor | An Enum representing the different colors a can have |
| Enum | eVehicleStatus | An Enum representing the status of a vehicle in the garage |
| Enum | eLicenseType | An Enum representing the different types of motorcycle license |
| Enum | eNumOfDoors | An Enum representing number of doors in a car |
| Enum | VehicleFactory.eVehicleType | An Enum representing the different types of vehicles the garage supports. It is nested in the VehicleFactory class so that in order to add a vehicle type in the future, we only need to change this class |
| Class | EnumOperations | A class providing functionality on our enums. Namely it provides a function for showing the values of an enum |

* Classes inheritance diagram:



