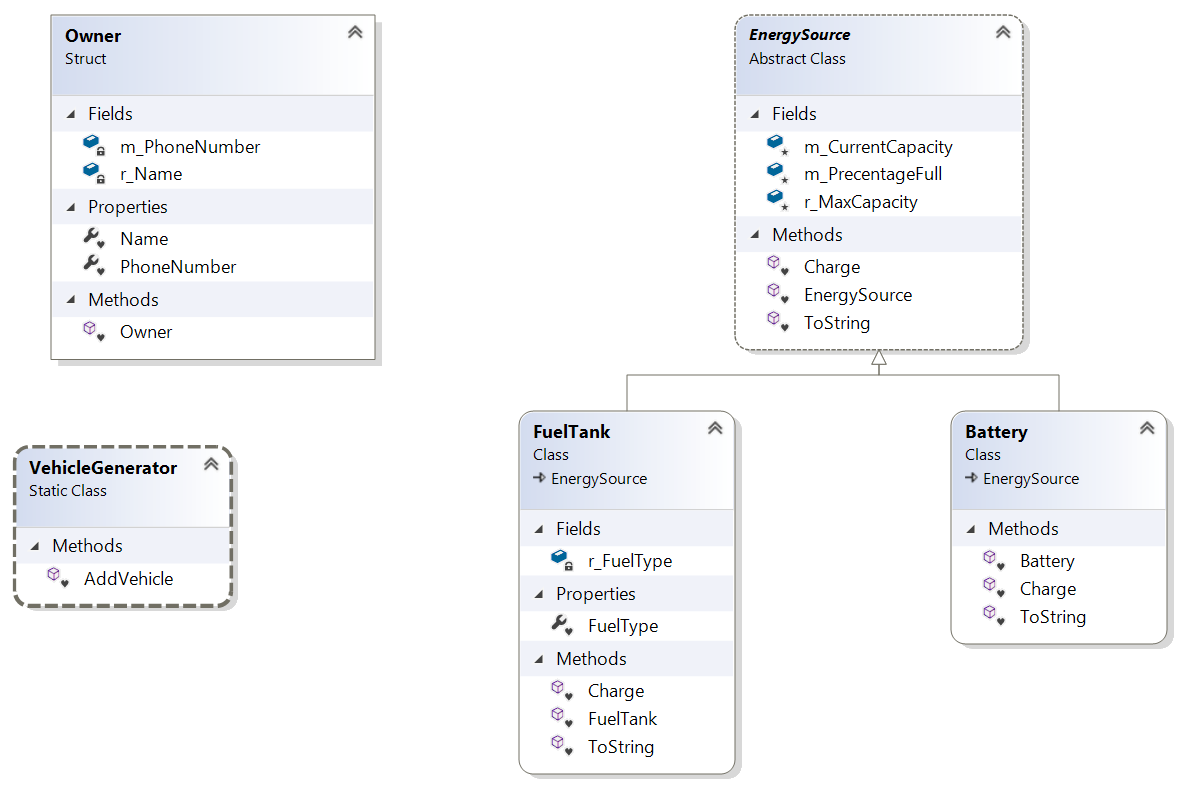
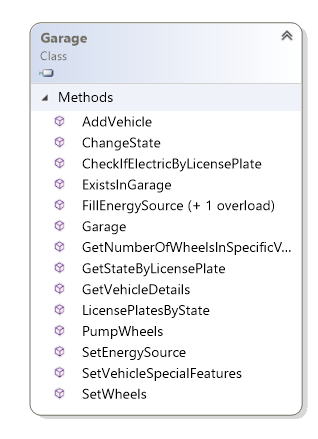
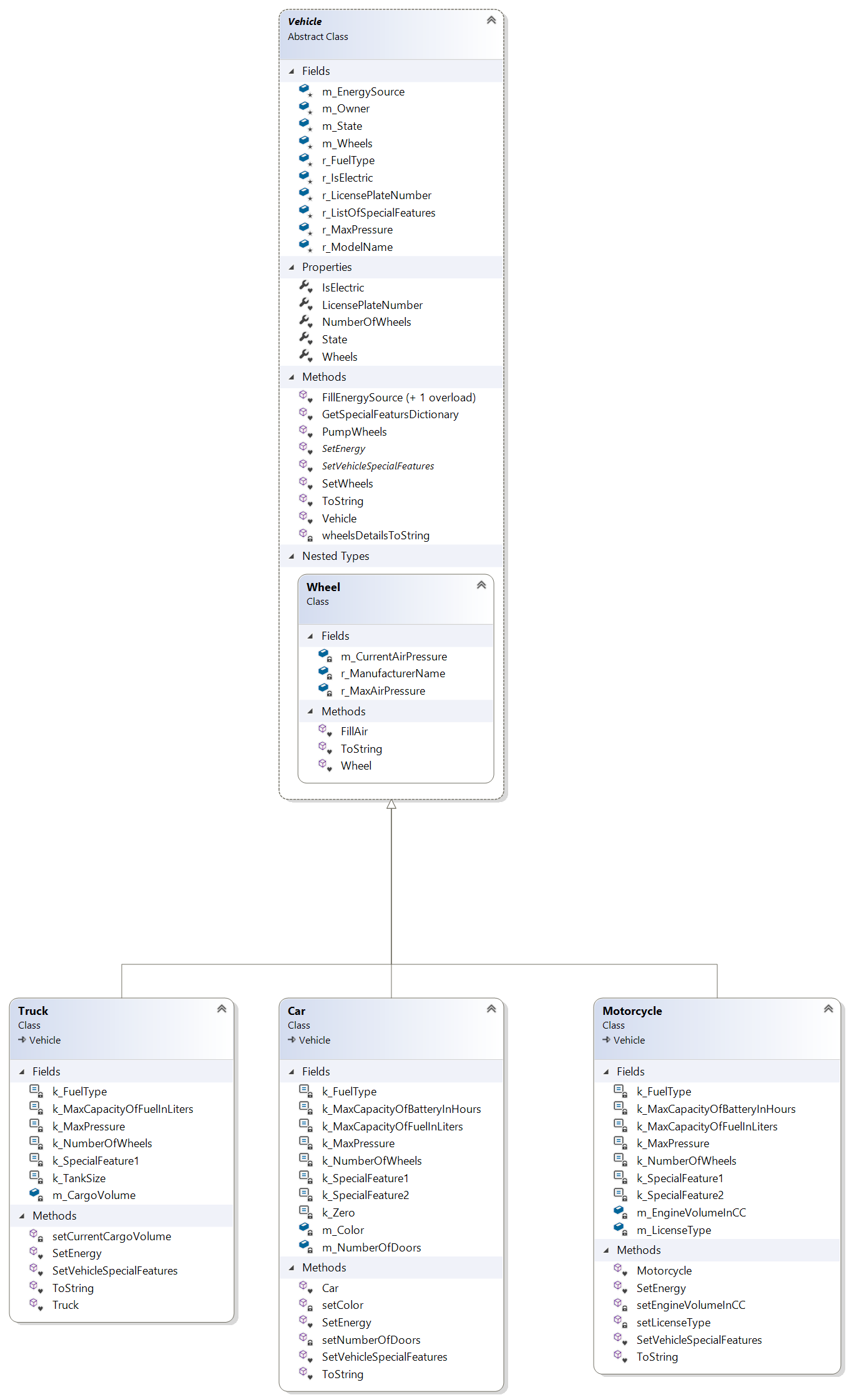
**GarageLogic**

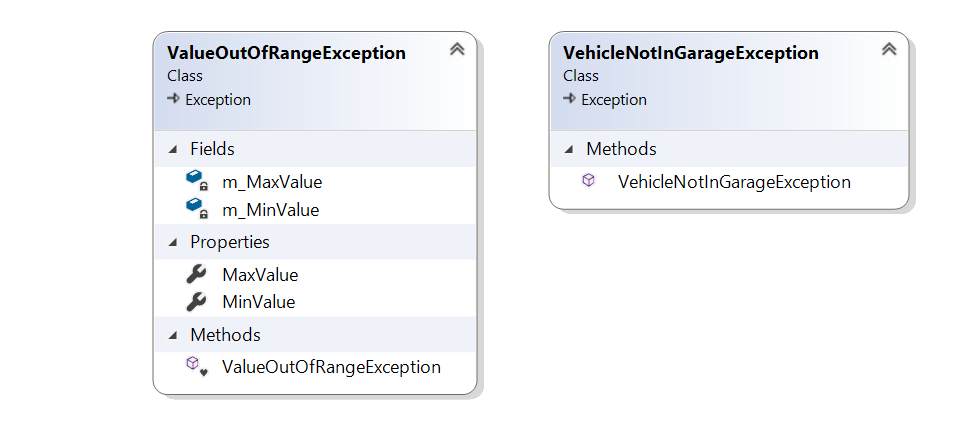
|  |  |
| --- | --- |
| **Short Explanation** | **Class** |
| Represents an energy source. Has max and current capacity and has a Charge() method that fills the EnergySource with the given value if it's legal. | **EnergySource**  **(Abstract)** |
| Implements energy source. Represents an energySource for a Fueled Car. | **FuelTank** |
| Implements energy source. Represents an energySource for an electric Car. | **Battery** |
| Represents a Vehicle in the Garage. Contains the data of the vehicle, setters, getters and Methods that Maintain the vehicle.  To make the program modular (so we are able to add a new vehicle without changing any of the existing classes) every class extending vehicle has a function that sets all the special properties that a specific vehicle type has in addition to the basic vehicle fields. | **Vehicle (Abstract)** |
| Implements Vehicle. Represents a Car typed Vehicle. Can be electric or fueled. | **Car** |
| Implements Vehicle. Represents a Motorcycle typed Vehicle. Can be electric or fueled. | **Motorcycle** |
| Implements Vehicle. Represents a Motorcycle typed Vehicle. A truck is always fueled. | **Truck** |
| Represents a wheel of a Vehicle. Each vehicle has an array Of it's wheels as a field. The Wheel Object contains Data about the current and max air pressure and a FillAir(float amount) function that fills the wheel with a given amount of air (if it is legal). | **Wheel** |
| Represents a Garage. Contains 3 dictionaries of type <String (license plate) Vehicle> (Fixed, Fixing and payed Vehicles) Contains methods for Running a garage, and controlling the Vehicles Registered in it. | **Garage** |
| A Class that generates a Vehicle (Car, Truck or Motorcycle) based on the arguments given in the function call by overriding one method for each vehicle type in the system. | **VehicleGenerator**  **(static)** |

|  |  |
| --- | --- |
| **Short Explanation** | **Struct** |
| Represents an owner of a car. Holds owner name and phone number.  A field Of Vehicle Class. | **Owner** |

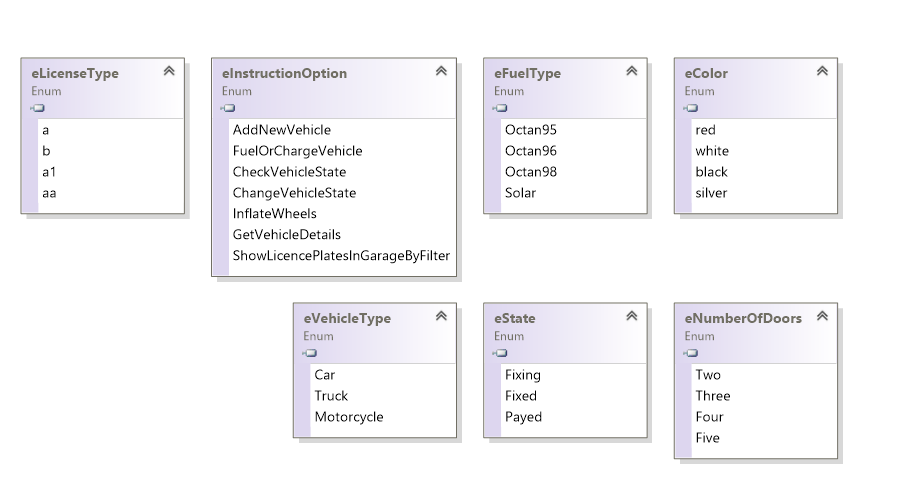




|  |  |
| --- | --- |
| **Short Explanation** | **Exception Class** |
| Thrown every time the garage class is asked to preform an operation on a vehicle that doesn't exist in the garage. | **VehicleNotInGarageException** |
| Thrown when a given value of an argument is not in the legal range. Has a MaxValue and MinValue field. | **ValueOutOfRangeException** |



|  |  |
| --- | --- |
| **Short Explanation** | **Enum** |
| Holds all Motorcycle license type. | **eLicenceType** |
| Holds all Legal fuel types. | **eFuleType** |
| Holds all Car color types. | **seColor** |
| Holds all possible vehicle states in a Garage. | **eState** |
| Holds all possible numbers of doors for a car. | **eNumberOfDoors** |
| Holds all possible vehicle types in the garage. (Every vehicle type added to the system must be added to this enum.) | **eVehicleType** |
| Holds all possible instructions that can be given to the system by the user. | **eInstructionOptions** |



**ConsoleUI**

|  |  |
| --- | --- |
| **Short Explanation** | **Class** |
| The entrance point to the program. Creates an instance of GarageRunner and calls it's Run() function. | **Program**  **(static)** |
| A class that prints all messages to the user. | **Printer**  **(static)** |
| Runs the main loop of the program-   1. Calls the GetInstructionFromUser() function of Communicator class. 2. Parses the instruction. 3. Calls the corresponding function in GarageManager Class. | **GarageRunner** |
| Holds a garage instance as a field and preforms operations requested by the user on it.  It is the mediator class between communicator (User) and garage which communicates with the GarageLogic classes. | **GarageManager** |
| A class responsible for communicating with the user and getting the inputs from the user. | **Communicator**  **(static)** |

