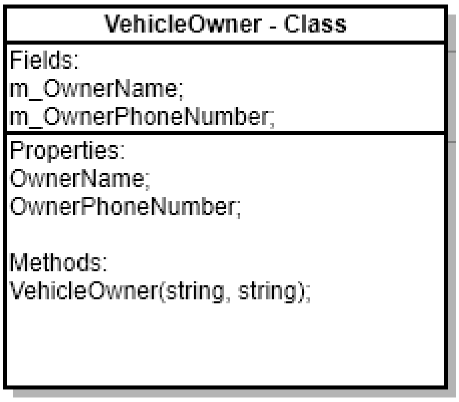
Roei Sagiv 205435555
Eden Slavin 315074906

Project: Ex03.GarageLogic		
Type	**Name**	**Explanation**
Class	Vehicle	Abstract class. Represents a Vehicle in the Garage with all the details that need to be to a vehicle with get and set for the properties and methods. Have abstract methods that each new type of vehicle extends from this class need to write them. Have methods fuelTank() and chargeBattery() that handles the vehicle.
Class	Wheel	A class inside Vehicle. Represents each wheel in the Vehicle with all the details that need to be in wheel with get and set for the properties and methods. Have a methods InfaltingAWheel(), InfaltingAWheelToMax(), ToStringWheel()
Class	VehicleOwner	Represents each owner of the Vehicle with all the details that need to be in owner with get and set for the properties. Have fields of Owner name and owner phone number.
Class	Battery	Represents battery of vehicle that is electric with all the details that need to be in battery with get and set for the properties. MaxBatteryInHours, CurrentBatteryInHours, BatteryPercentageRemain. The battery have the methods ChargeBattery() and ToString()
Class	FuelTank	Represents fuel tank of vehicle that is not electric with all the details that need to be in fuel tank with get and set for the properties. MaxFuelTankInLiters, CurrentFuelTankInLiters,FuelPercentageRemainInTank,FuelType. The fuel tank have the methods FuelTheTank() and ToString()
Class	Motorcycle	Implements Vehicle. Represents a Motorcycle typed Vehicle. Have specials fields for the motorcycle - EngineVolumeCC and TypeOfLicense. Can be electric or fueled.
Class	Car	Implements Vehicle. Represents a Car typed Vehicle. Have specials fields for the car- CarColor and CarDoorQuantity. Can be electric or fueled.
Class	Truck	Implements Vehicle. Represents a Car typed Vehicle. Have specials fields for the car- MaxCarryingWeight and IsContainHazardousMaterials. Can be electric only fueled.
Class	Garage	Represents a Garage contains dictionary VehicalsInGarage all the vehicles in the garage and dictionary VehicalStatusInGarage all the vehicles by the status. Contains all the methods for the functionals that the garage needs to run.
Class	VehicleTypeGenerator	Class that generates a vehicle. Ccreate each type of vehicle and initialize the vehicle.

Class	ValueOutOfRangeException	Extends From Exception. This exception thrown when the system get value from the user that is not in the range. Has fields MaxValue and MinValue
Enum	eVehicleType	Have all the options for vehicle type
Enum	eFuelTank	Have all the options for fuel tank
Enum	eMotorcycleLicenseType	Have all the options for motorcycle license
Enum	eCarClor	Have all the options for car colors
Enum	eCarDoorQuantity	Have all the options for car doors quantity
Enum	eVehicleStatusInGarage	Have all the options for the status of vehicle in garage

Roei Sagiv 205435555
Eden Slavin 315074906

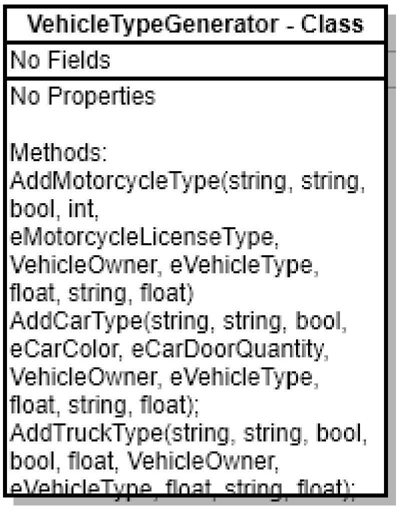


VehicleOwner - Class

Fields:
m_OwnerName;
m_OwnerPhoneNumber;

Properties:
OwnerName;
OwnerPhoneNumber;

Methods:
VehicleOwner(string, string);



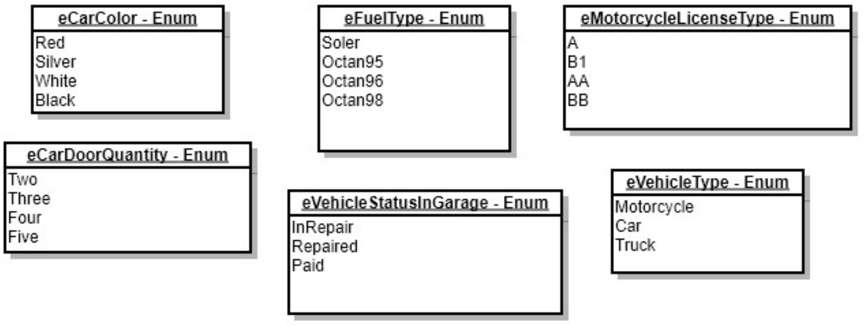
VehicleTypeGenerator - Class

No Fields

No Properties

Methods:
AddMotorcycleType(string, string, bool, int, eMotorcycleLicenseType, VehicleOwner, eVehicleType, float, string, float)
AddCarType(string, string, bool, eCarColor, eCarDoorQuantity, VehicleOwner, eVehicleType, float, string, float);
AddTruckType(string, string, bool, bool, float, VehicleOwner, eVehicleType, float, string, float);

Roei Sagiv 205435555
Eden Slavin 315074906



eCarColor - Enum
Red
Silver
White
Black

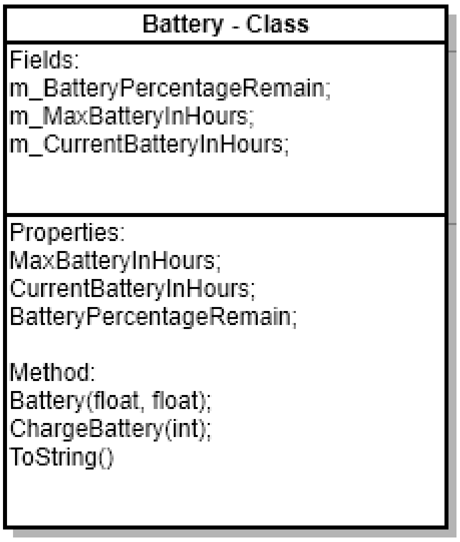
eFuelType - Enum
Soler
Octan95
Octan96
Octan98

eMotorcycleLicenseType - Enum
A
B1
AA
BB

eCarDoorQuantity - Enum
Two
Three
Four
Five

eVehicleStatusInGarage - Enum
InRepair
Repaired
Paid

eVehicleType - Enum
Motorcycle
Car
Truck



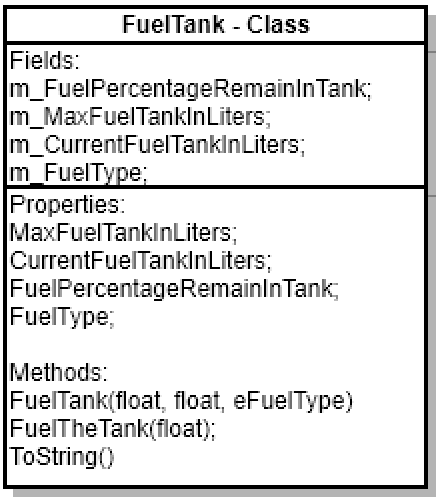
Battery - Class

Fields:
m_BatteryPercentageRemain;
m_MaxBatteryInHours;
m_CurrentBatteryInHours;

Properties:
MaxBatteryInHours;
CurrentBatteryInHours;
BatteryPercentageRemain;

Method:
Battery(float, float);
ChargeBattery(int);
ToString()

Roei Sagiv 205435555
Eden Slavin 315074906



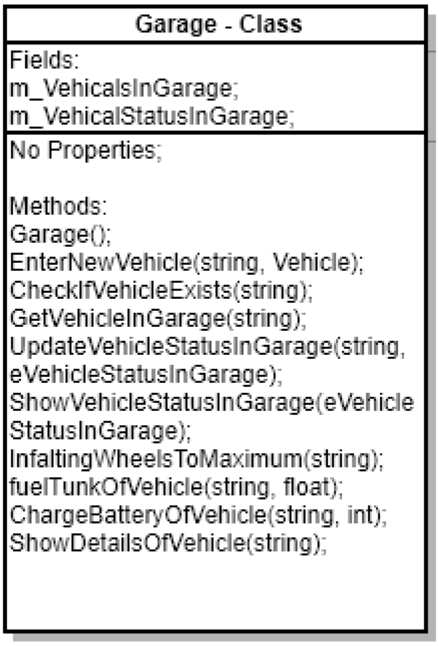
FuelTank - Class

Fields:
m_FuelPercentageRemainInTank;
m_MaxFuelTankInLiters;
m_CurrentFuelTankInLiters;
m_FuelType;

Properties:
MaxFuelTankInLiters;
CurrentFuelTankInLiters;
FuelPercentageRemainInTank;
FuelType;

Methods:
FuelTank(float, float, eFuelType)
FuelTheTank(float);
ToString()

Roei Sagiv 205435555
Eden Slavin 315074906

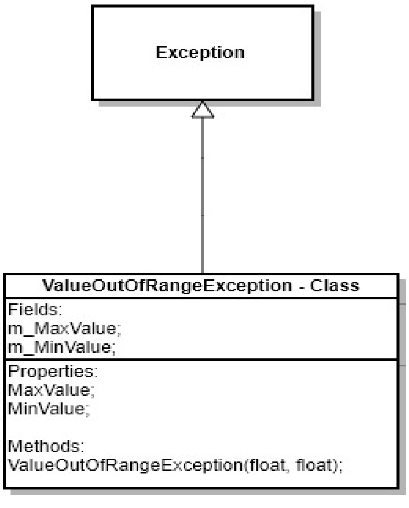


Garage - Class

Fields:
m_VehicalsInGarage;
m_VehicalStatusInGarage;

No Properties;

Methods:
Garage();
EnterNewVehicle(string, Vehicle);
CheckIfVehicleExists(string);
GetVehicleInGarage(string);
UpdateVehicleStatusInGarage(string, eVehicleStatusInGarage);
ShowVehicleStatusInGarage(eVehicleStatusInGarage);
InfaltingWheelsToMaximum(string);
fuelTunkOfVehicle(string, float);
ChargeBatteryOfVehicle(string, int);
ShowDetailsOfVehicle(string);



Exception

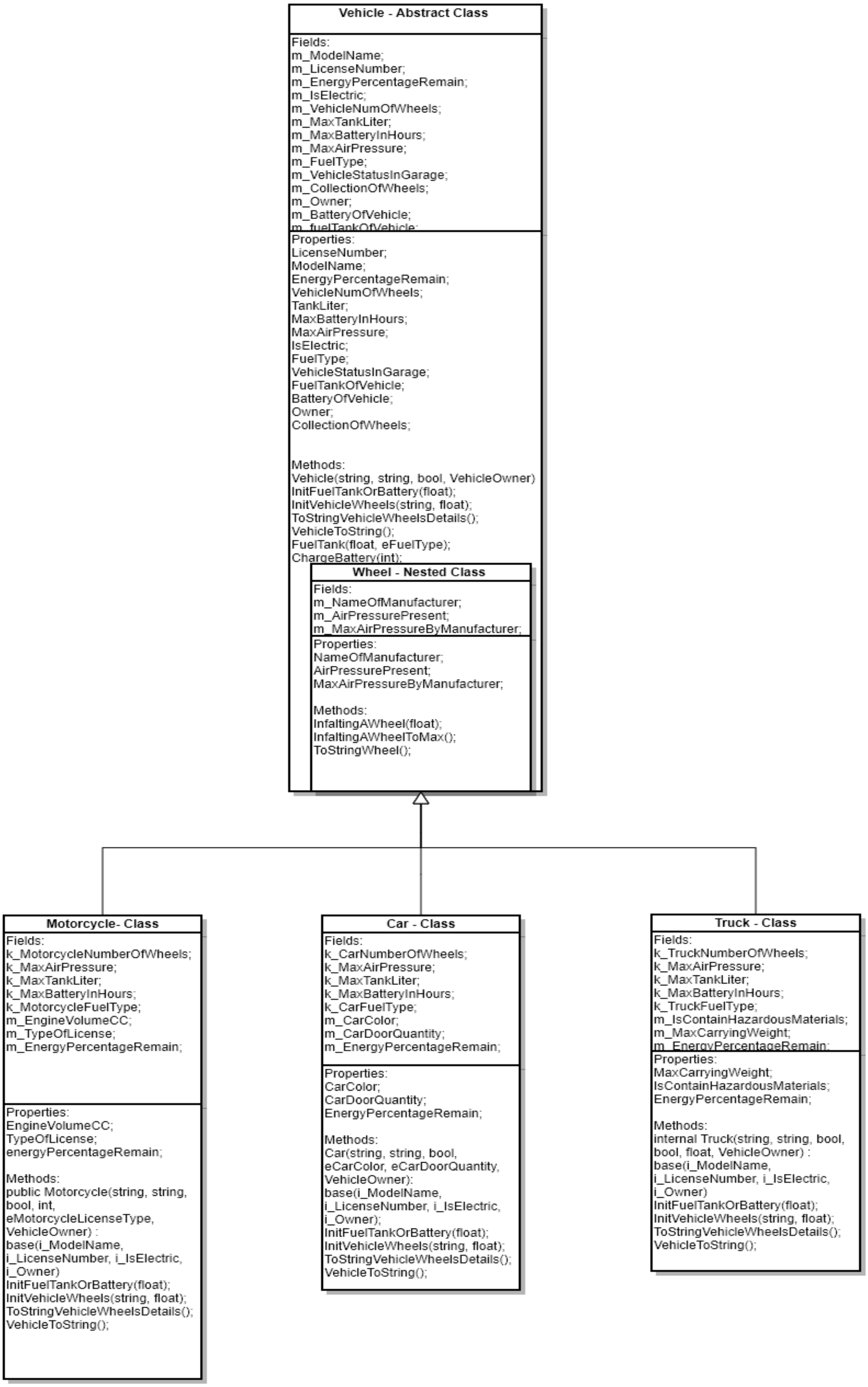
ValueOutOfRangeException - Class

Fields:
m_MaxValue;
m_MinValue;

Properties:
MaxValue;
MinValue;

Methods:
ValueOutOfRangeException(float, float);

Roei Sagiv 205435555
Eden Slavin 315074906



Vehicle - Abstract Class

Fields:
m_ModelName;
m_LicenseNumber;
m_EnergyPercentageRemain;
m_IsElectric;
m_VehicleNumOfWheels;
m_MaxTankLiter;
m_MaxBatteryInHours;
m_MaxAirPressure;
m_FuelType;
m_VehicleStatusInGarage;
m_CollectionOfWheels;
m_Owner;
m_BatteryOfVehicle;
m_fuelTankOfVehicle;

Properties:
LicenseNumber;
ModelName;
EnergyPercentageRemain;
VehicleNumOfWheels;
TankLiter;
MaxBatteryInHours;
MaxAirPressure;
IsElectric;
FuelType;
VehicleStatusInGarage;
FuelTankOfVehicle;
BatteryOfVehicle;
Owner;
CollectionOfWheels;

Methods:
Vehicle(string, string, bool, VehicleOwner)
InitFuelTankOrBattery(float);
InitVehicleWheels(string, float);
ToStringVehicleWheelsDetails();
VehicleToString();
FuelTank(float, eFuelType);
ChargeBattery(int);

Wheel - Nested Class

Fields:
m_NameOfManufacturer;
m_AirPressurePresent;
m_MaxAirPressureByManufacturer;

Properties:
NameOfManufacturer;
AirPressurePresent;
MaxAirPressureByManufacturer;

Methods:
InfaltingAWheel(float);
InfaltingAWheelToMax();
ToStringWheel();

Motorcycle- Class

Fields:
k_MotorcycleNumberOfWheels;
k_MaxAirPressure;
k_MaxTankLiter;
k_MaxBatteryInHours;
k_MotorcycleFuelType;
m_EngineVolumeCC;
m_TypeOfLicense;
m_EnergyPercentageRemain;

Properties:
EngineVolumeCC;
TypeOfLicense;
energyPercentageRemain;

Methods:
public Motorcycle(string, string, bool, int, eMotorcycleLicenseType, VehicleOwner) : base(i_ModelName, i_LicenseNumber, i_IsElectric, i_Owner)
InitFuelTankOrBattery(float);
InitVehicleWheels(string, float);
ToStringVehicleWheelsDetails();
VehicleToString();

Car - Class

Fields:
k_CarNumberOfWheels;
k_MaxAirPressure;
k_MaxTankLiter;
k_MaxBatteryInHours;
k_CarFuelType;
m_CarColor;
m_CarDoorQuantity;
m_EnergyPercentageRemain;

Properties:
CarColor;
CarDoorQuantity;
EnergyPercentageRemain;

Methods:
Car(string, string, bool, eCarColor, eCarDoorQuantity, VehicleOwner): base(i_ModelName, i_LicenseNumber, i_IsElectric, i_Owner);
InitFuelTankOrBattery(float);
InitVehicleWheels(string, float);
ToStringVehicleWheelsDetails();
VehicleToString();

Truck - Class

Fields:
k_TruckNumberOfWheels;
k_MaxAirPressure;
k_MaxTankLiter;
k_MaxBatteryInHours;
k_TruckFuelType;
m_IsContainHazardousMaterials;
m_MaxCarryingWeight;
m_EnergyPercentageRemain;

Properties:
MaxCarryingWeight;
IsContainHazardousMaterials;
EnergyPercentageRemain;

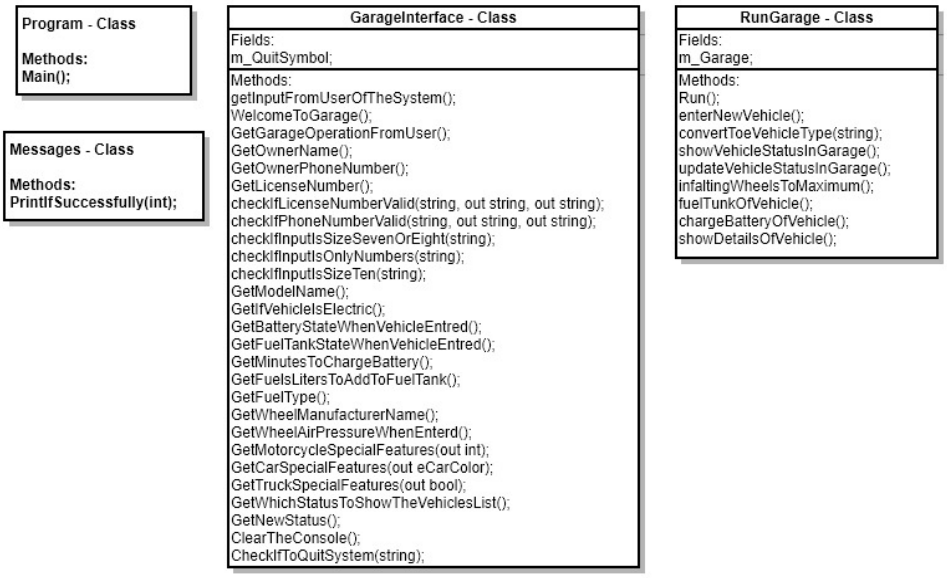
Methods:
internal Truck(string, string, bool, bool, float, VehicleOwner) : base(i_ModelName, i_LicenseNumber, i_IsElectric, i_Owner)
InitFuelTankOrBattery(float);
InitVehicleWheels(string, float);
ToStringVehicleWheelsDetails();
VehicleToString();

Roei Sagiv 205435555
Eden Slavin 315074906

Project: Ex03.ConsoleUI		
Type	**Name**	**Explanation**
Class	Program	This class is the entry point to our program. Have a main that creates RunGarage and start run him with the method Run().
Class	GarageInterface	Class that connected with the user of the system. Have methods that ask the user questions and get from the user the input.

Roei Sagiv 205435555
Eden Slavin 315074906

Class	RunGarage	Run all the garage with all the methods he can do. Ask the user -> get the input -> and run the function by the operation from the user. If the methods need, they catch the exceptions and print the user the errors.
Class	Messages	Have the methods that the user gets after the system successfully the operation that the user asks the system to do.



Program - Class

Methods:
Main();

Messages - Class

Methods:
PrintIfSuccessfully(int);

GarageInterface - Class

Fields:
m_QuitSymbol;

Methods:
getInputFromUserOfTheSystem();
WelcomeToGarage();
GetGarageOperationFromUser();
GetOwnerName();
GetOwnerPhoneNumber();
GetLicenseNumber();
checkIfLicenseNumberValid(string, out string, out string);
checkIfPhoneNumberValid(string, out string, out string);
checkIfInputIsSizeSevenOrEight(string);
checkIfInputIsOnlyNumbers(string);
checkIfInputIsSizeTen(string);
GetModelName();
GetIfVehicleIsElectric();
GetBatteryStateWhenVehicleEntred();
GetFuelTankStateWhenVehicleEntred();
GetMinutesToChargeBattery();
GetFuelsLitersToAddToFuelTank();
GetFuelType();
GetWheelManufacturerName();
GetWheelAirPressureWhenEnterd();
GetMotorcycleSpecialFeatures(out int);
GetCarSpecialFeatures(out eCarColor);
GetTruckSpecialFeatures(out bool);
GetWhichStatusToShowTheVehiclesList();
GetNewStatus();
ClearTheConsole();
CheckIfToQuitSystem(string);

RunGarage - Class

Fields:
m_Garage;

Methods:
Run();
enterNewVehicle();
convertToeVehicleType(string);
showVehicleStatusInGarage();
updateVehicleStatusInGarage();
infaltingWheelsToMaximum();
fuelTunkOfVehicle();
chargeBatteryOfVehicle();
showDetailsOfVehicle();