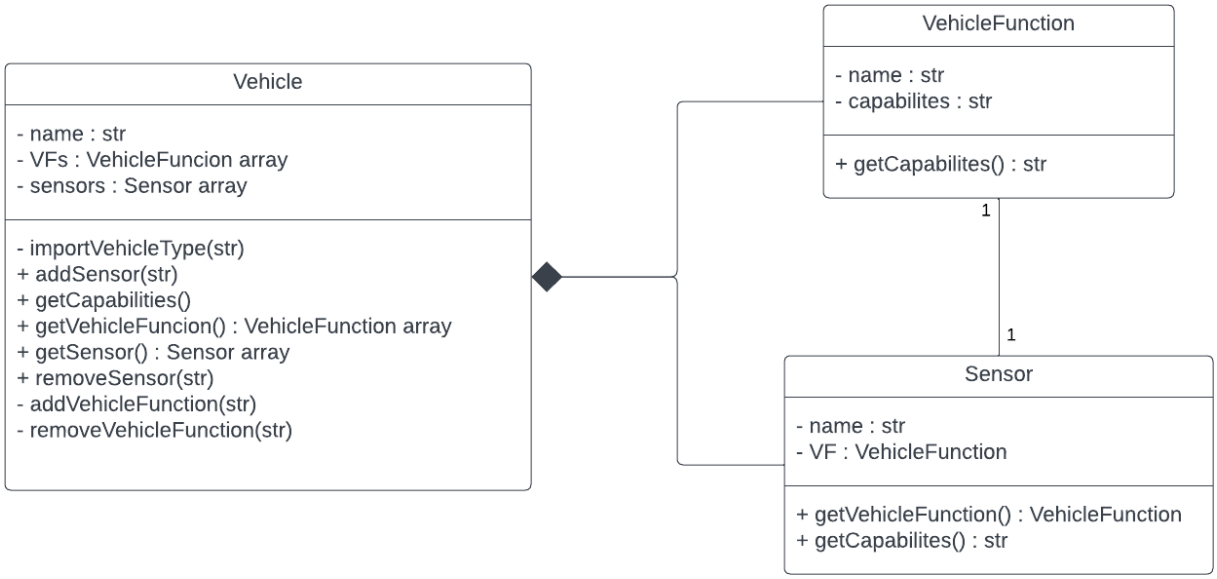

In order to meet Oscar's requirement we can create three classes: **Vehicle**, **Sensor** and **VehicleFunction**.

- **VehicleFunction** contains the functions that the vehicle is able to perform. Each function is identified by its name (e.g. overtaking) and the corresponding capability (i.e. "The vehicle is capable of overtaking"). This object will be used by Oscar during the tests.
  For example, consider the test case:
  "Drive in an highway, set a speed higher than the vehicle in front of you -> the vehicle shall perform the overtaking".
  If the vehicle has the overtaking VehicleFunction, it will pass the test, otherwise it will not.
  The method getCapabilities() gives as output the description of the VehicleFunction under test.
- **Sensor** is the class that includes all the features of each sensor, in this case the name (e.g. radar) and the corresponding VehicleFunction (e.g. overtaking).
  The methods included in this class are getVehicleFunction(), which provides the VehicleFunction object related to a sensor, and getCapabilities() which describes its capabilities.
- **Vehicle** is the class that represents the whole car object. Its objects are defined by their name (the vehicle line name, e.g. Elite), the **Sensor** and **Vehicle Function** arrays.
  The method importVehicleType(str) identifies the "base" vehicle functions defined by the car model from which the vehicle derives (in this case Ant). Since the only parameters that the customer can choose are the vehicle line and the sensor list, the vehicle type is pre-defined in a database that maps it directly to the chosen vehicle line. The same database includes also the list of sensors allowed for each vehicle line, and the vehicle functions connected to each sensor and vehicle type. This solution makes the system easy to scale. In fact, it is possible ~~in order~~ to add new sensors or vehicle lines and types to it by simply adding them to the database, leaving the code unchanged.
  The methods addSensor(str) and removeSensor(str) give the possibility to add/remove a sensor from a vehicle object already created. These functions recall addVehicleFunction(str) and removeVehicleFunction(str) which add/remove the VehicleFunction related to the sensor
  The last three methods are getSensor(), getVehicleFunction() and getCapabilities() and they are respectively getters of: the array of Sensor objects, the array of VehicleFunction objects, and the list of capabilities of the vehicle.