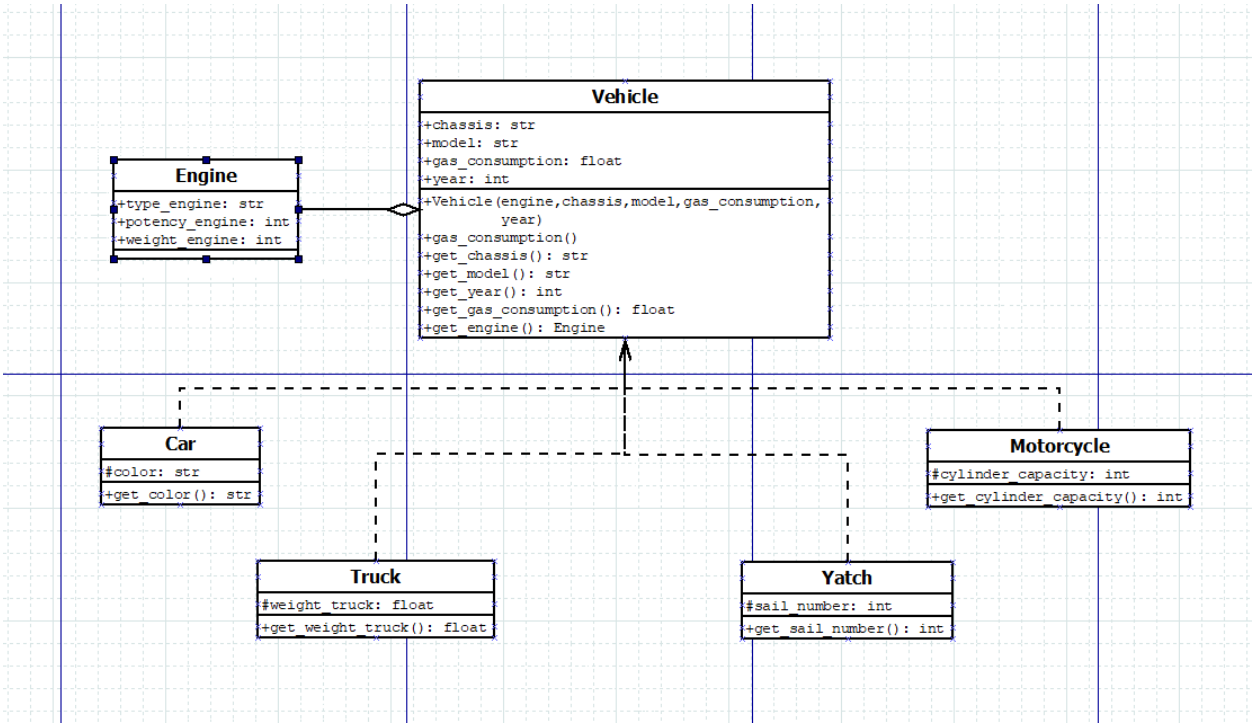
UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Report: workshop 1

Cristian Andres Gamez Nuñez

Advanced programming

Class Diagram:



In this program the object-oriented programming paradigm was used, object-oriented programming is used to organize and structure the code efficiently. The classes and the relationships of these classes (aggregation, inheritance) allow easy extension and modification of the program, as well as the reuse of the code, which is an advantage that this paradigm gives us.

The classes in this program are the following:

Engine: This class represents a vehicle engine. It has attributes such as name, type_, potency, and weight. It is used to create engine instances that are then assigned to vehicles. The name attribute is used to identify the engine, since it will be stored in a dictionary and the name would be the key to each engine

Vehicle: It is an class that represents a generic vehicle. Contains common attributes such as chassis, model, year, engine, and gas_consumption. It also has methods to obtain the information about these attributes, which are getter methods. For the chassis attribute, a validation is done to verify that the data is either "A" or "B", if this is different it will throw an error in the console. A

validation is also done for the year attribute and if it is less than 1990 it will throw an error in the console

Specific vehicle classes (Car, Truck, Yatch, Motorcycle): These classes inherit from the Vehicle class and represent specific types of vehicles. Each has its own set of additional attributes and methods to access them. For example, Car has a color attribute, Truck has weight_truck, Yatch has sail_number, and Motorcycle has cylinder_capacity.

The functions found in this program are the following:

create_engine(): Allows the user to create a new engine. Prompts the user to enter details such as engine name, type, power, and weight, then creates an Engine instance and adds it to an engines dictionary

create_vehicle(vehicle_type): This function creates a new vehicle of the specified type (car, truck, yatch, motorcycle). Asks the user to enter specific details of the vehicle type, such as color for a car, maximum weight for a truck, sail number for a yacht, etc., and creates an instance of the corresponding class (Car, Truck, Yatch, Motorcycle) with the specified engine.

menu(): This function displays an interactive menu to the user, where they can select different options, such as create an engine, create a vehicle, show all created engines, show all created vehicles, or exit the program.

Program Flow:

1. The program starts by calling the menu() function.
2. The user can select various options from the menu.
3. Depending on the selected option, the corresponding functions (create_engine(), create_vehicle(), etc.) are invoked.
4. After completing the selected action, the user can choose another option or exit the program.