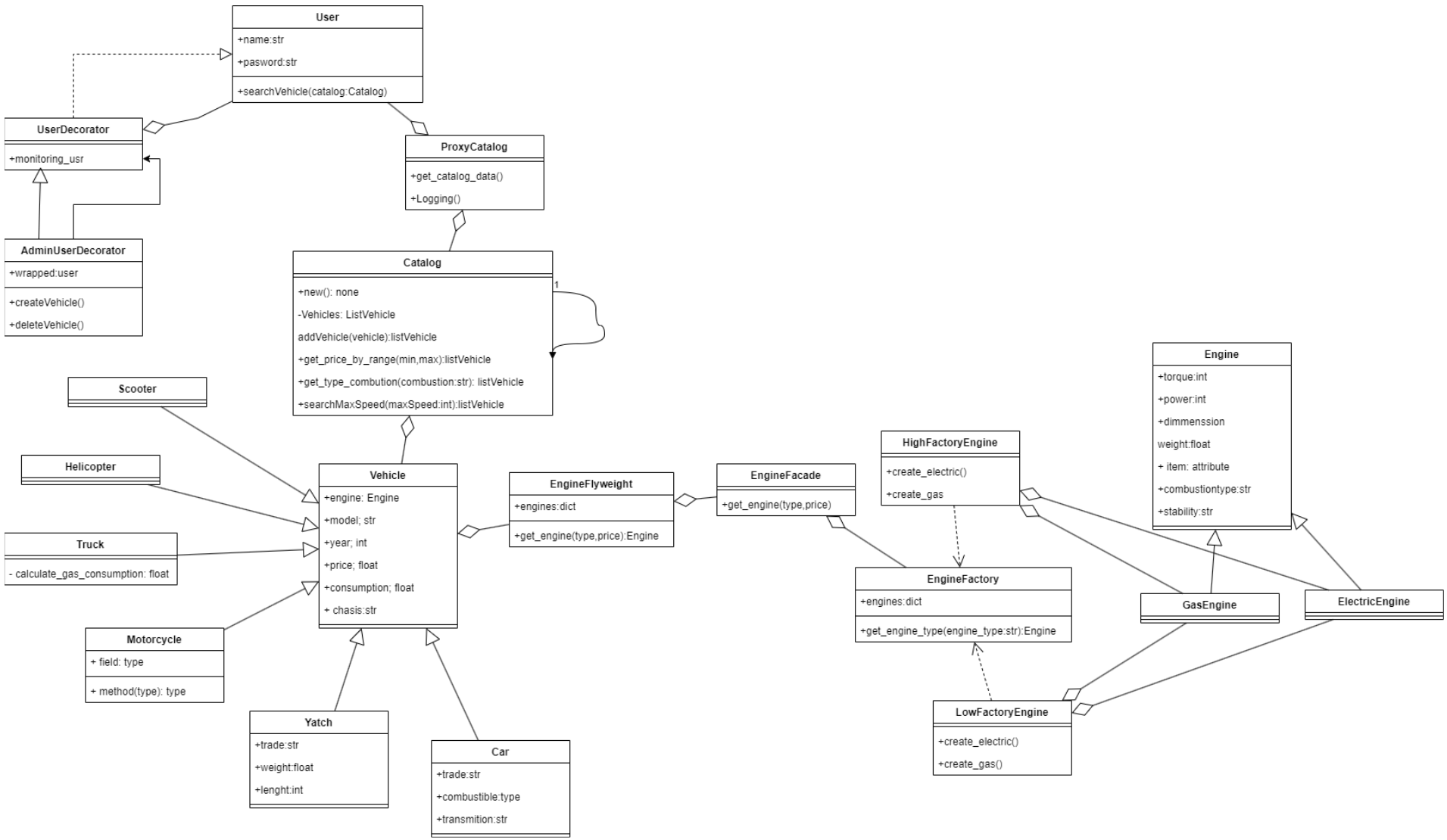
Technical report workshop 3

This report is meant to show the technical desitions and strategies used during workshop 3

Decision making

- Since it is meant to use less memory it will be tried to implement flyweight pattern
- Facade pattern is seen to be the easiest way to approach the subsystem interaction
- Since user is meant to be monitored, and also the performance of the application it is gonna be implemented a proxy pattern
- Decided to create a base user class to later use decorators on it
- Flyweight implemented for engines to reduce memory consumption
- Implemented façade on the engines part since we want to make the engines its own subsysted that at the same time it helps with encapsulation
- Engine facade only interacts with the engine flyweight and the flyweight interacts directly with the vehicle, this facade and flyweight helps with encapsulation and memory consumption at the same time in the past of the engines
- Used decorators to represent the admin user and the normal user that interacts with the catalog
- Vehicle part is still the same but since it only interacts with the façade of engine and with the catalog it is more encapsulated than before

Class diagram:


User
+name:str
+pasword:str
+searchVehicle(catalog:Catalog)
UserDecorator
+monitoring_usr
ProxyCatalog
+get_catalog_data()
+Logging()
AdminUserDecorator
+wrapped:user
+createVehicle()
+deleteVehicle()
Catalog
+new(): none
-Vehicles: ListVehicle
addVehicle(vehicle):listVehicle
+get_price_by_range(min,max):listVehicle
+get_type_combution(combustion:str): listVehicle
+searchMaxSpeed(maxSpeed:int):listVehicle
1
Scooter
Helicopter
Truck
- calculate_gas_consumption: float
Motorcycle
+ field: type
+ method(type): type
Vehicle
+engine: Engine
+model; str
+year; int
+price; float
+consumption; float
+ chasis:str
Yatch
+trade:str
+weight:float
+lenght:int
Car
+trade:str
+combustible:type
+transmition:str
EngineFlyweight
+engines:dict
+get_engine(type,price):Engine
EngineFacade
+get_engine(type,price)
HighFactoryEngine
+create_electric()
+create_gas
EngineFactory
+engines:dict
+get_engine_type(engine_type:str):Engine
LowFactoryEngine
+create_electric()
+create_gas()
Engine
+torque:int
+power:int
+dimmenssion
weight:float
+ item: attribute
+combustiontype:str
+stability:str
GasEngine
ElectricEngine