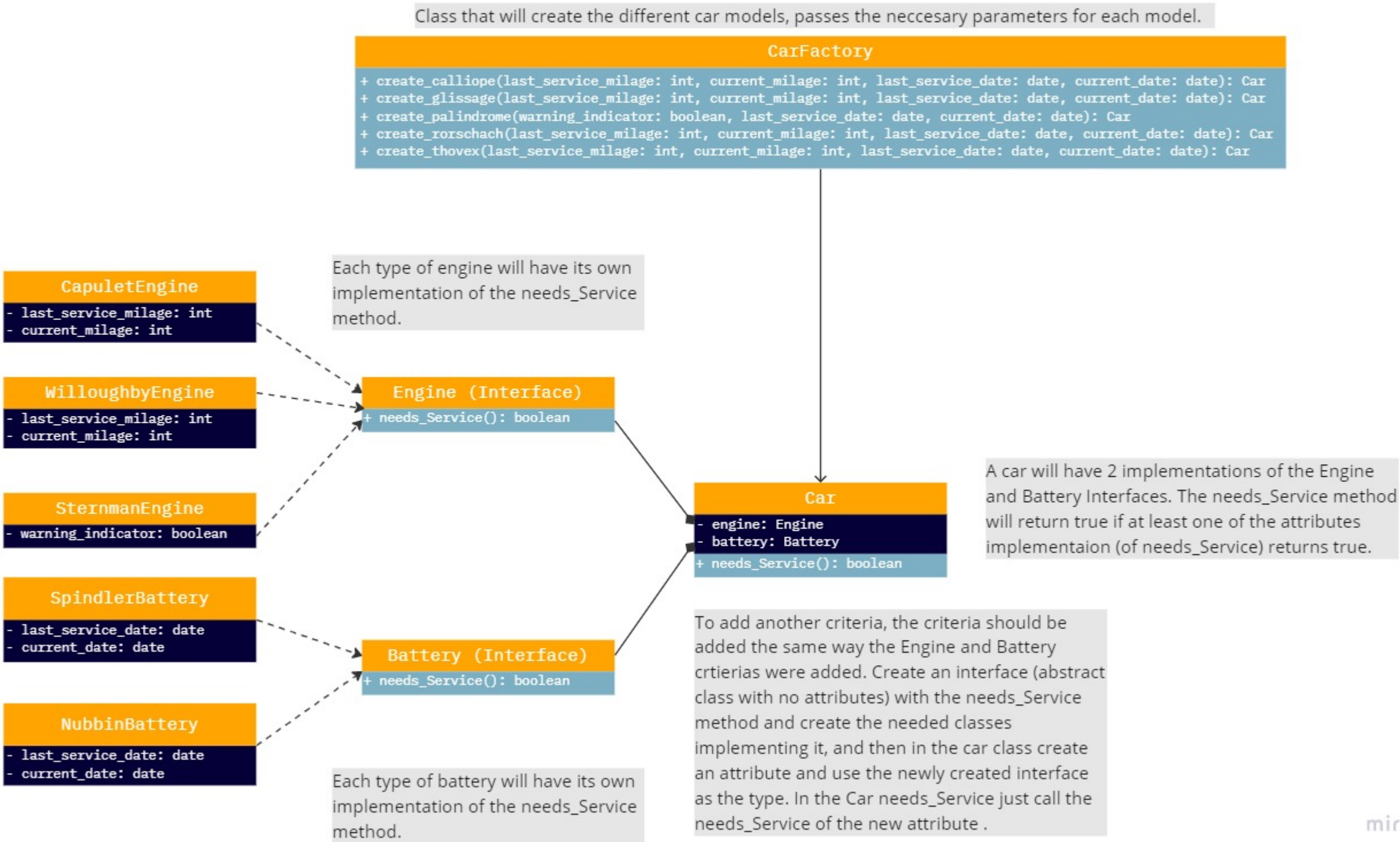
Class that will create the different car models, passes the neccesary parameters for each model.
CarFactory
+ create_calliope(last_service_milage: int, current_milage: int, last_service_date: date, current_date: date): Car
+ create_glissage(last_service_milage: int, current_milage: int, last_service_date: date, current_date: date): Car
+ create_palindrome(warning_indicator: boolean, last_service_date: date, current_date: date): Car
+ create_rorschach(last_service_milage: int, current_milage: int, last_service_date: date, current_date: date): Car
+ create_thovex(last_service_milage: int, current_milage: int, last_service_date: date, current_date: date): Car
Each type of engine will have its own implementation of the needs_Service method.
CapuletEngine
- last_service_milage: int
- current_milage: int
WilloughbyEngine
- last_service_milage: int
- current_milage: int
SternmanEngine
- warning_indicator: boolean
Engine (Interface)
+ needs_Service(): boolean
Car
- engine: Engine
- battery: Battery
+ needs_Service(): boolean
A car will have 2 implementations of the Engine and Battery Interfaces. The needs_Service method will return true if at least one of the attributes implementaion (of needs_Service) returns true.
SpindlerBattery
- last_service_date: date
- current_date: date
NubbinBattery
- last_service_date: date
- current_date: date
Battery (Interface)
+ needs_Service(): boolean
To add another criteria, the criteria should be added the same way the Engine and Battery crtierias were added. Create an interface (abstract class with no attributes) with the needs_Service method and create the needed classes implementing it, and then in the car class create an attribute and use the newly created interface as the type. In the Car needs_Service just call the needs_Service of the new attribute .
Each type of battery will have its own implementation of the needs_Service method.
miro