Лютенко, КС34, варіант 1

Вступ

Facade - структурний шаблон проектування, що надає спрощений інтерфейс до бібліотеки чи фреймворку, чи будь-яким іншим складним сетів класів.

Уявімо, що вам необхідно ваш код працював з різноманітною множиною об’єктів, що належать складдній бібліотеці чи фреймворку. Зазвичай, потрібно проініціалізувати усі ці об’єкти, відслідковувати залежності, використовувати велику кільскість методів і тому інше.

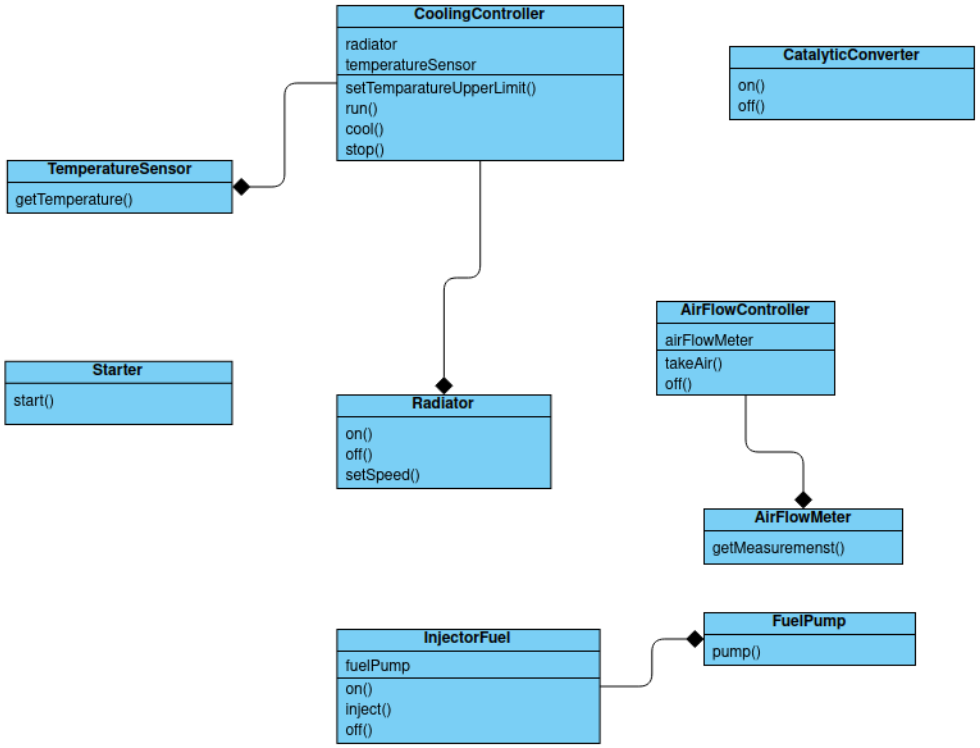
У результаті логіка додатку ваших класів буде тісно пов’язана з деталями реалізації сторонніх класів, що ускладнить її розуміння та підтримку.

Вирішення

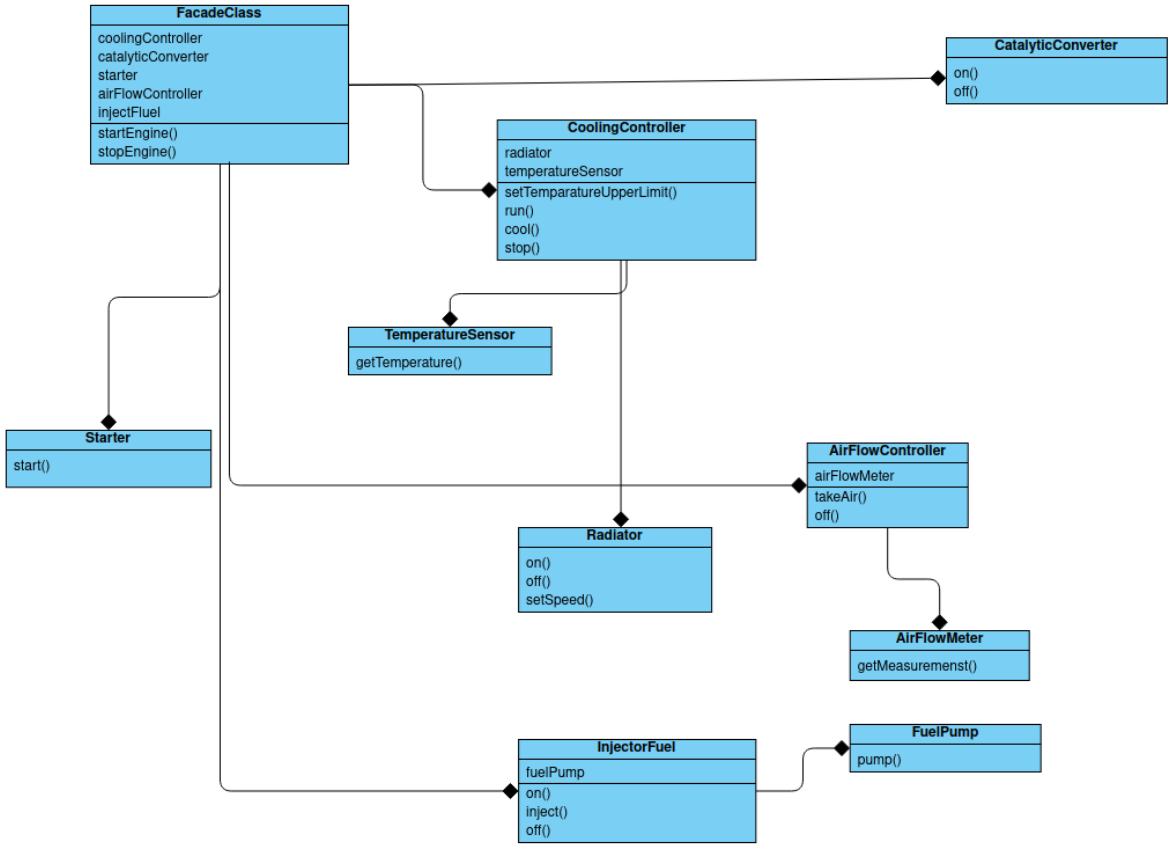
Facade-клас має надати спрощений інтерфейс до складних підсистем додатку, що містять у собі багато змінних та незалежних систем. Також facade може надавати зменшену чи обмежену функціональність в порівнянні з тим, коли ви працюєте напряму з підсистема додатка. Але він включає лише ті методи, що лише необхідні користувачу для взаємодії.

Problem

Уявімо такий випадок, коли в нас є legacy-система, що відображає імітацію роботи двигуна автомобіля, а точніше усієї його інженерної частини. В діаграмі класів є такі класи, як: охолоджувальна система, датчик температури, контролер повітря, паливний насос тощо. Ось ця діаграма:



З шаблоном проектування facade маємо те, що під одним логічним іменем буде доступні інші об’єкти класів.



Тепер, щоби користувачу розпочати працювати з бібліотеки непотрібно прописувати ініціалізація таким чином:

```
airFlowController.takeAir()
fuelInjector.on()
fuelInjector.inject()
starter.start()
coolingController.setTemperatureUpperLimit(DEFAULT_COOLING_TEMP)
coolingController.run()
catalyticConverter.on()
```

Відповідно й непотрібно прописувати закінчення роботи:

```
fuelInjector.off()
catalyticConverter.off()
coolingController.cool(MAX_ALLOWED_TEMP)
coolingController.stop()
airFlowController.off()
```

Замість цього все, шо потрібно користувачу, щоб почати/закінчити роботу з даною бібліотекою, що імітує роботи двигуна автомобіля, робимо так:

```
        facade.startEngine();
        // ...
        facade.stopEngine();
```

Ось сам приклад facade-класу:

```
public class CarEngineFacade {
        private static int DEFAULT_COOLING_TEMP = 90;
        private static int MAX_ALLOWED_TEMP = 50;
        private FuelInjector fuelInjector = new FuelInjector();
        private AirFlowController airFlowController = new AirFlowController();
        private Starter starter = new Starter();
        private CoolingController coolingController = new CoolingController();
        private CatalyticConverter catalyticConverter = new
CatalyticConverter();

        public void startEngine() {
        fuelInjector.on();
        airFlowController.takeAir();
        fuelInjector.on();
        fuelInjector.inject();
        starter.start();

coolingController.setTemperatureUpperLimit(DEFAULT_COOLING_TEMP);
        coolingController.run();
        catalyticConverter.on();
        }

        public void stopEngine() {
        fuelInjector.off();
        catalyticConverter.off();
        coolingController.cool(MAX_ALLOWED_TEMP);
        coolingController.stop();
        airFlowController.off();
        }
}
```

Висновок

Facade-патерн скриває за спрощеним інтерфейсом складність системи чи різноманість підсистем, з яких вона скаладається.

Посилання гітхаб репозиторій з фрагментом коду facade-класу:
https://github.com/lutenkoNikita/pic/