Юрин Михаил

ПИ-15-1

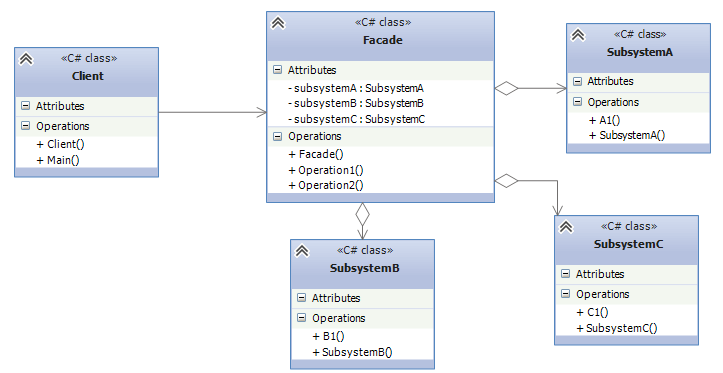
Паттерн «Facade»

***1. Назначение и цель применение паттерна***

Фасад — это простой интерфейс работы со сложной подсистемой, содержащей множество классов. Фасад может иметь урезанный интерфейс, не имеющий 100% функциональности, которую можно достичь, используя сложную подсистему напрямую. Но он предоставляет именно те фичи, которые нужны клиенту, и скрывает все остальное.

Фасад полезен, если вы используете какую-то сложную библиотеку с множеством подвижных частей, но вам нужна только часть её возможностей.

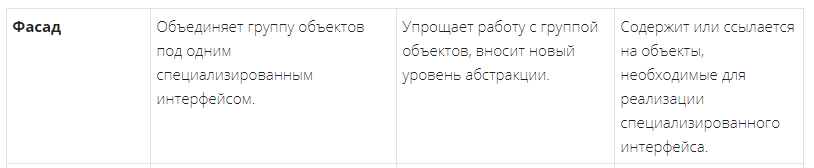
***2. UML представление паттерна***

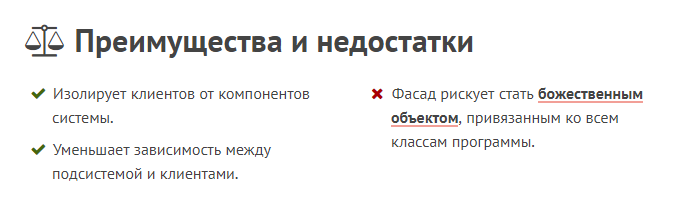


***3. Области применения***

1. Когда имеется сложная система, и необходимо упростить с ней работу. Фасад позволит определить одну точку взаимодействия между клиентом и системой.
2. Когда надо уменьшить количество зависимостей между клиентом и сложной системой. Фасадные объекты позволяют отделить, изолировать компоненты системы от клиента и развивать и работать с ними независимо.
3. Когда нужно определить подсистемы компонентов в сложной системе. Создание фасадов для компонентов каждой отдельной подсистемы позволит упростить взаимодействие между ними и повысить их независимость друг от друга.

***4. Особенности паттерна***





***5. Пример реализации паттерна C#***

Вариант 1. https://habrahabr.ru/post/87110/  
Вариант 2. <https://metanit.com/sharp/patterns/4.3.php>  
Вариант 3. <http://andrey.moveax.ru/post/patterns-oop-structural-facade>

***6. Список источников***

1. <https://habrahabr.ru/post/87110/>

2. <https://refactoring.guru/ru/design-patterns/facade>

3. <http://largescalejs.ru/the-facade-pattern/>

4. <https://metanit.com/sharp/patterns/4.3.php>

5. http://andrey.moveax.ru/post/patterns-oop-structural-facade