Java Design Patterns

Facade

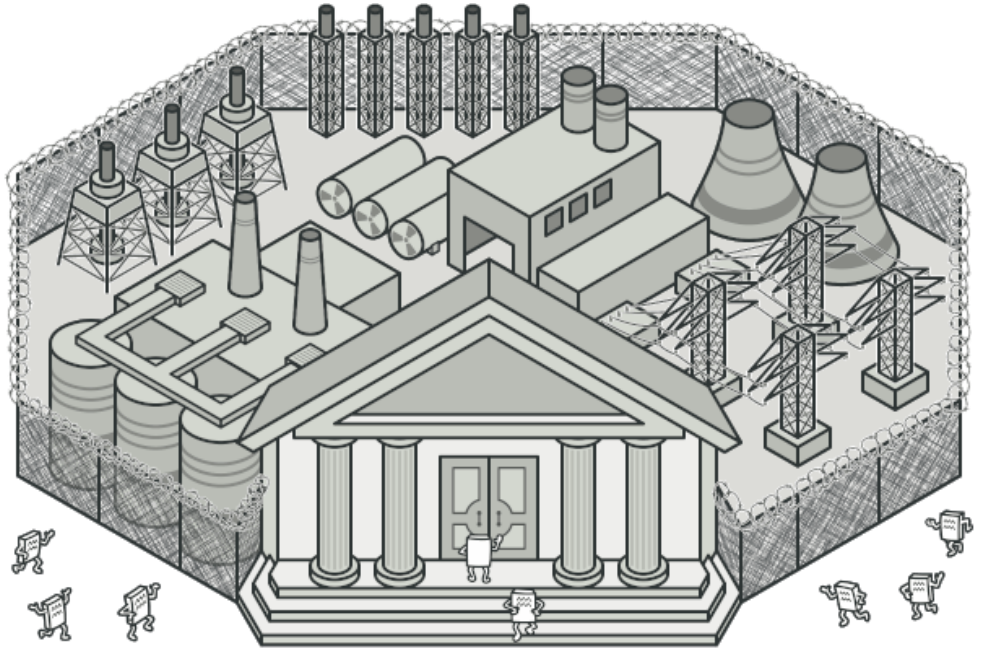
Тема

Facade

Суть паттерна

Фасад

Фасад — это структурный паттерн проектирования, который предоставляет простой интерфейс к сложной системе классов, библиотеке или фреймворку.



Проблема

Постановка задачи

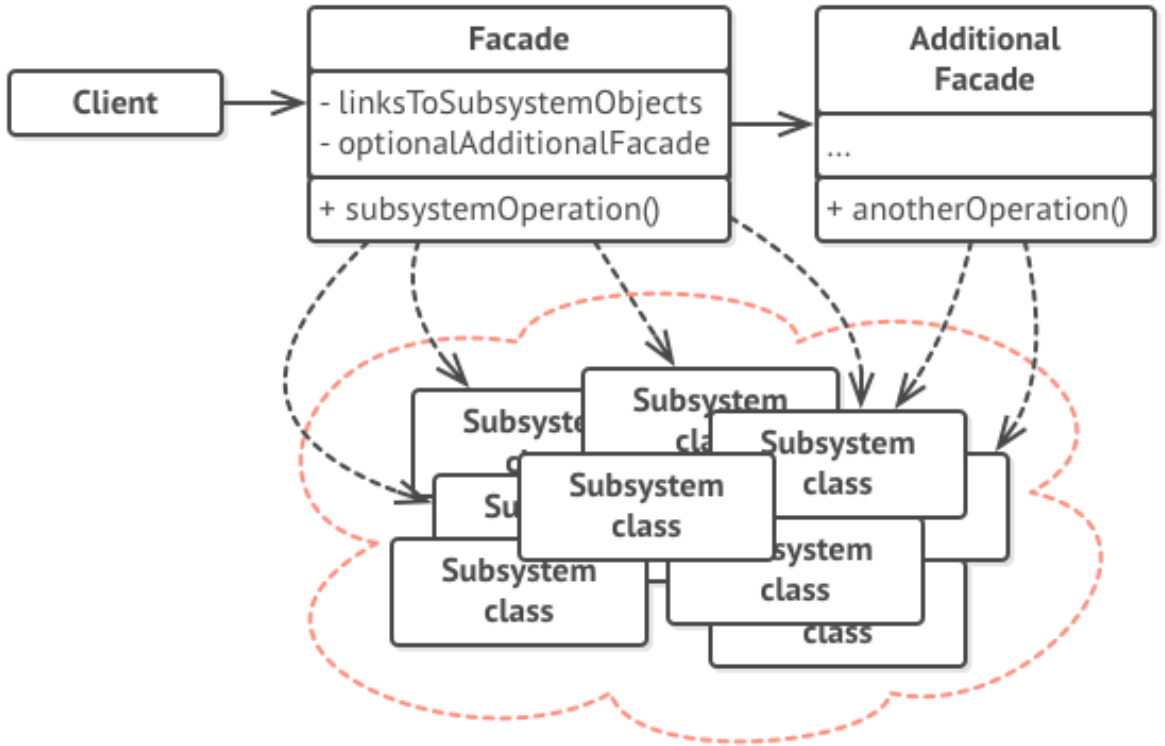
Вашему коду приходится работать с большим количеством объектов некой сложной библиотеки или фреймворка. Вы должны самостоятельно инициализировать эти объекты, следить за правильным порядком зависимостей и так далее.
В результате, бизнес-логика ваших классов тесно переплетается с деталями реализации сторонних классов. Такой код довольно сложно понимать и поддерживать.

Решение задачи

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.

Структура паттерна

1. Фасад предоставляет быстрый доступ к определённой функциональности подсистемы. Он «знает», каким классам нужно переадресовать запрос, и какие данные для этого нужны.
2. Дополнительный фасад можно ввести, чтобы не захламлять единственный фасад разнородной функциональностью. Он может использоваться как клиентом, так и другими фасадами.
3. Сложная подсистема состоит из множества разнообразных классов. Для того чтобы заставить их что-то делать, нужно знать подробности устройства подсистемы, порядок инициализации объектов и так далее.
4. Сложная подсистема состоит из множества разнообразных классов. Для того чтобы заставить их что-то делать, нужно знать подробности устройства подсистемы, порядок инициализации объектов и так далее.



Применимость

Применение паттерна

1. Когда вам нужно представить простой или урезанный интерфейс к сложной подсистеме.
2. Когда вы хотите разложить подсистему на отдельные слои.

Шаги реализации

Алгоритм реализации паттерна

1. Определите можно ли создать более простой интерфейс, чем тот, который предоставляет сложная подсистема. Вы на правильном пути, если этот интерфейс избавит клиента от знания о подробностях подсистемы.
2. Создайте класс фасада, реализующий этот интерфейс. Он должен переадресовывать вызовы клиента нужным объектам подсистемы. Фасад должен будет позаботиться о том, чтобы правильно инициализировать объекты подсистемы.
3. Вы получите максимум пользы, если клиент будет работать только с фасадом. В этом случае, изменения в подсистеме будут затрагивать только код фасада, а клиентский код останется рабочим.
4. Если ответственность фасада начинает размываться, подумайте о введении дополнительных фасадов.

Преимущества и недостатки

Плюсы и недостатки

Плюсы:

- Изолирует клиентов от компонентов системы.
- Уменьшает зависимость между подсистемой и клиентами.

Минусы:

- Фасад рискует стать божественным объектом, привязанным ко всем классам программы.

Отношения с другими паттернами

Отношение с другими паттернами

- Фасад задаёт новый интерфейс, тогда как Адаптер повторно использует старый. *Адаптер* оборачивает только один класс, а *Фасад* оборачивает целую подсистему. Кроме того, *Адаптер* позволяет двум существующим интерфейсам работать сообща, вместо того, чтобы задать полностью новый.
- Абстрактная фабрика может быть использована вместо Фасада для того, чтобы скрыть платформо-зависимые классы.
- Легковес показывает, как создавать много мелких объектов, а Фасад показывает, как создать один объект, который отображает целую подсистему.
- Посредник и Фасад похожи тем, что пытаются организовать работу множества существующих классов.
  - *Фасад* создаёт упрощённый интерфейс к подсистеме, не внося в неё никакой добавочной функциональности. Сама подсистема не знает о существовании *Фасада*. Классы подсистемы общаются друг с другом напрямую.
  - *Посредник* централизует общение между компонентами системы. Компоненты системы знают только о существовании *Посредника*, у них нет прямого доступа к другим компонентам.
- Фасад можно сделать Одиночкой, так как обычно нужен только один объект-фасад.
- Фасад похож на Заместитель тем, что замещает сложную подсистему и может сам её инициализировать. Но в отличие от *Фасада*, *Заместитель* имеет тот же интерфейс, что его служебный объект, благодаря чему их можно взаимозаменять.

Информационный видеосервис для разработчиков программного обеспечения

ITVDN

IT VIDEO DEVELOPERS NETWORK

TestProvider PROMETRIC TEST CENTER CyberBionic systematics Microsoft Partner Silver Learning Windows Azure Cloud Partner EBA European Business Association