Эссе «GOF. Паттерн Facade»

Паттерны проектирования GoF

Паттерны проектирования, представленные в книге "Банды четырех" (Gang of Four, GoF), являются фундаментальными инструментами для разработки гибких и поддерживаемых программных систем.

Группа паттернов GoF включает 23 паттерна, которые делятся на три основные категории: порождающие, структурные и поведенческие. Каждый паттерн решает конкретную проблему и предоставляет абстрактное описание решения, которое можно адаптировать к конкретному контексту. Рассмотрим несколько примеров паттернов из каждой категории.

***Порождающие паттерны*:**

Фабричный метод (Factory Method): Позволяет создавать объекты определенного типа, но делегирует процесс создания подклассам. Это способствует гибкости и устраняет необходимость привязывать код к конкретным классам.

Абстрактная фабрика (Abstract Factory): Предоставляет интерфейс для создания семейств связанных объектов без указания их конкретных классов. Это позволяет создавать объекты согласованно и с учетом зависимостей между ними.

Структурные паттерны:

Адаптер (Adapter): Позволяет интерфейсу одного класса работать с интерфейсом другого класса, делая их несовместимые интерфейсы совместимыми. Это обеспечивает переиспользование существующего кода и интеграцию различных компонентов.

Фасад (Facade): Предоставляет унифицированный интерфейс к группе интерфейсов в подсистеме, упрощая взаимодействие клиентов с этой подсистемой и скрывая ее сложность.

Поведенческие паттерны:

Наблюдатель (Observer): Определяет зависимость между объектами так, чтобы при изменении состояния одного объекта все его зависимые объекты уведомлялись и обновлялись автоматически. Это способствует слабой связности между компонентами системы.

Стратегия (Strategy): Позволяет определять семейство алгоритмов, инкапсулировать их и делать их взаимозаменяемыми. Это обеспечивает гибкость выбора алгоритма во время выполнения.

Паттерн Facade

Паттерн Фасад (Facade) - это структурный паттерн проектирования, который предоставляет унифицированный интерфейс к группе интерфейсов в подсистеме. Он

позволяет клиентам взаимодействовать с подсистемой через одну упрощенную точку входа, скрывая детали реализации и сложность работы с этой подсистемой.

Главная цель паттерна Фасад - упростить взаимодействие клиента с подсистемой и сделать код клиента более независимым от деталей реализации. В результате улучшается структура кода, уменьшается связность и повышается читаемость.

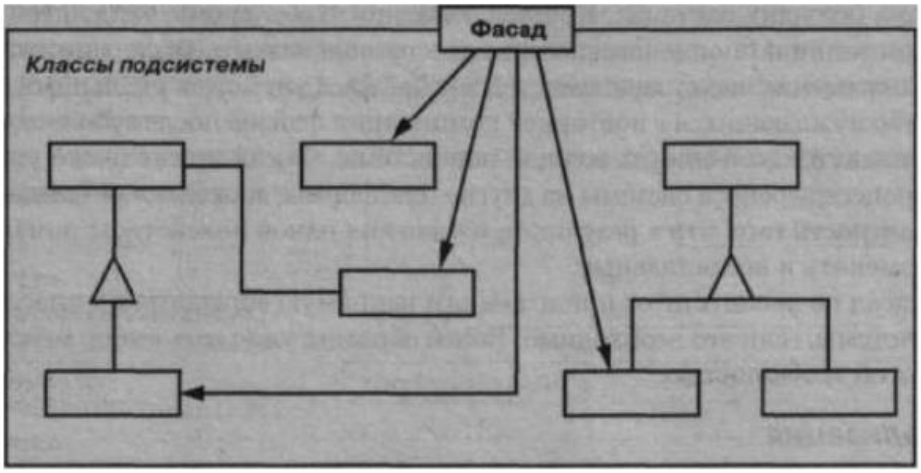
Применимость:

Используйте паттерн фасад, когда:

- хотите предоставить простой интерфейс к сложной подсистеме. Часто подсистемы усложняются по мере развития. Применение большинства паттернов приводит к появлению меньших классов, но в большем количестве. Такую подсистему проще повторно использовать и настраивать под конкретные нужды, но вместе с тем применять подсистему без настройки становится труднее. Фасад предлагает некоторый вид системы по умолчанию, устраивающий большинство клиентов. И лишь те объекты, которым нужны более широкие возможности настройки, могут обратиться напрямую к тому, что находится за фасадом;

- между клиентами и классами реализации абстракции существует много зависимостей. Фасад позволит отделить подсистему как от клиентов, так и от других подсистем, что, в свою очередь, способствует повышению степе ни независимости и переносимости;

- а вы хотите разложить подсистему на отдельные слои. Используйте фасад для определения точки входа на каждый уровень подсистемы. Если подсистемы зависят друг от друга, то зависимость можно упростить, разрешив подсистемам обмениваться информацией только через фасады.



Участники:

- Facade (Compiler) фасад:
  1) «знает», каким классам подсистемы адресовать запрос
  2) делегирует запросы клиентов подходящим объектам внутри подсистемы;

- Классы подсистемы (Scanner, Parser, Program Node и т.д.):
  1) реализуют функциональность подсистемы
  2) выполняют работу, порученную объектом Facade
  3) ничего не «знают» о существовании фасада, то есть не хранят ссылок на него.

Отношения:

Клиенты общаются с подсистемой, посылая запросы фасаду. Он переадресует их подходящим объектам внутри подсистемы. Хотя основную работу выполняют именно объекты подсистемы, фасаду, возможно, придется преобразовать свой интерфейс в интерфейсы подсистемы.

Клиенты, пользующиеся фасадом, не имеют прямого доступа к. объектам подсистемы.

Преимущества:

Паттерн Фасад предоставляет простой и упрощенный интерфейс к сложной системе или подсистеме, что приносит следующие преимущества:

1. Упрощение интерфейса: Скрывает детали сложности системы от клиентов.

2. Сокрытие деталей реализации: Позволяет внесение изменений в систему без влияния на клиентский код.

3. Уменьшение связности: Снижает зависимость клиента от множества классов системы.

4. Улучшение читаемости кода: Делает код более читаемым и понятным для разработчиков.

5. Облегчение тестирования: Упрощает создание юнит-тестов для подсистемы.

6. Повышение безопасности: Может обеспечивать контроль доступа.

7. Легкость интеграции: Упрощает интеграцию различных компонентов и сервисов.