Черник Матвей Денисович 212 группа

«The Gang of Four» и паттерн Facade
«The Gang of Four» (GoF) и паттерн «Facade» — два основополагающих элемента в области проектирования и архитектуры программного обеспечения. «The Gang of Four» относится к Эриху Гамме, Ричарду Хелму, Ральфу Джонсону и Джону Влиссидесу, которые в 1994 году написали влиятельную книгу «Design Patterns: Elements of Reusable Object-Oriented Software». В этой книге была представлена концепция паттернов (шаблонов) проектирования, включая паттерн Facade, который с тех пор стал фундаментальным инструментом в наборе инструментов разработки программного обеспечения. Покажем значение паттернов проектирования GoF и Facade, изучая их происхождение, принципы и реальное применение.

Банда четырех (GoF):
«The Gang of Four», или GoF, сыграла ключевую роль в формировании области разработки программного обеспечения. Их книгу Design Patterns часто считают плодотворной работой в этой дисциплине. Авторов вдохновила потребность в общем словаре и структуре для обсуждения и документирования повторяющихся проблем проектирования и их решений. С тех пор их работа стала краеугольным камнем в разработке объектно-ориентированного программного обеспечения.

Происхождение:
Истоки GoF можно проследить до середины 1990-х годов, когда создатели вместе работали в Object Technology International, Inc. (OTI), небольшой консалтинговой компании по программному обеспечению. В ходе сотрудничества они осознали необходимость каталогизировать и формализовать повторяющиеся решения распространенных проблем проектирования программного обеспечения. Их опыт привел их к формулировке 23 паттернов проектирования, каждый из которых стандартизированным образом решает конкретную проблему.

Принципы:

Абстракция: паттерны проектированя способствуют абстракции сложных систем на управляемые компоненты. Это делает программное обеспечение более модульным, удобным в сопровождении и понятным.

Инкапсуляция: они подчеркивают инкапсуляцию и сокрытие деталей реализации системы. Такое разделение задач повышает гибкость и уменьшает зависимости.

Наследование и композиция: использование композиции вместо наследования при построении объектно-ориентированных систем повышает гибкость и снижает риск жесткой связи.

Гибкость и расширяемость: паттерны проектирования способствуют созданию систем, которые могут развиваться и адаптироваться к меняющимся требованиям с минимальными нарушениями.

Повторное использование: Предоставляя проверенные решения повторяющихся проблем, паттерныч проектирования облегчают повторное использование кода, экономя время и усилия при разработке программного обеспечения.

Facade паттерн:
Одним из 23 шаблонов проектирования, представленных GoF, является паттерн Facade.

Паттерн «Facade» относится к категории шаблонов структурного проектирования и служит шлюзом к сложной подсистеме, предоставляя клиентам упрощенный интерфейс. Такое упрощение помогает управлять тонкостями системы, предоставляя точку входа высокого уровня.

Паттерн «Facade» предоставляет унифицированный интерфейс вместо набора интерфейсов некоторой подсистемы. Facade определяет интерфейс более высокого уровня, который упрощает использование подсистемы.
Проще говоря, «Facade» — есть ни что иное, как некоторый объект, аккумулирующий в себе высокоуровневый набор операций для работы с некоторой сложной подсистемой. Facade агрегирует классы, реализующие функциональность этой подсистемы, но не скрывает их

Принципы:
Паттерн «Facade» воплощает в себе несколько основных принципов:

Простота: он предлагает упрощенный и унифицированный интерфейс для набора интерфейсов внутри подсистемы, что упрощает взаимодействие клиентов с системой.

Абстракция: инкапсулируя сложности подсистемы, он защищает клиентов от деталей того, как работает подсистема, обеспечивая более высокий уровень абстракции.

Развязка: Facade отделяет клиентов от подсистемы, уменьшая зависимости и делая систему более удобной в обслуживании и гибкой.

Удобство использования: повышает удобство использования сложной системы, предоставляя более простой и интуитивно понятный интерфейс, сокращая время обучения пользователей.

Реальные приложения:
Паттерн «Facade» находит применение в различных областях, включая библиотеки программного обеспечения, платформы и системы. Некоторые примеры из реальной жизни включают в себя:

API операционной системы. Операционные системы часто предоставляют фасадные интерфейсы для управления сложными задачами, такими как файловый ввод-вывод, управление памятью и сетевое взаимодействие.

Графические библиотеки. Графические библиотеки, такие как OpenGL, используют паттерн Facade для упрощения взаимодействия со сложным графическим оборудованием и программным обеспечением.

Сложные программные системы. Большие программные системы могут использовать паттерн Facade для предоставления удобного интерфейса, одновременно выполняя сложную обработку за кулисами.

Использование паттерна Facade включает в себя несколько шагов и соображений:

Шаг 1. Определите сложную подсистему
Начните с определения сложной подсистемы вашего приложения. Эта подсистема обычно состоит из нескольких классов или компонентов, которые работают вместе для достижения определенной функциональности. Вы должны понимать, что клиентский код, напрямую взаимодействующий с этими классами подсистемы, может стать запутанным и сложным в обслуживании.

Шаг 2. Создайте паттерн Facade
Спроектируйте класс Facade, который будет служить упрощенной точкой входа в подсистему. Класс Facade должен инкапсулировать сложные взаимодействия с подсистемой и предоставлять клиентам понятный высокоуровневый интерфейс. Вот несколько рекомендаций по проектированию класса Facade:

Определите методы: определите, какие операции или функции подсистемы должны быть доступны через Facade. Создайте методы в классе Facade, соответствующие этим функциям.

Вызовы делегатов. Внутри каждого метода Facade делегируйте вызовы соответствующим классам подсистемы. Facade должен координировать взаимодействие с подсистемой.

Скрыть сложность: убедитесь, что Facade скрывает сложности подсистемы. Клиентам не нужно знать внутреннюю работу подсистемы.

Шаг 3. Реализация Façade's методов
Реализуйте методы класса Facade, делегируя задачи соответствующим классам подсистемы. В зависимости от сложности подсистемы это может включать вызов методов из нескольких классов подсистемы и организацию их взаимодействия.

Шаг 4. Интегрируйте Facade в клиентский код
Клиенты должны взаимодействовать с подсистемой через Facade, а не напрямую с классами подсистемы. Измените клиентский код, чтобы использовать методы Facade для доступа к функциональности подсистемы. Это гарантирует, что клиенты получат выгоду от упрощенного интерфейса, предоставляемого Facade.

Шаг 5: Создайте экземпляр Facade
В своем приложении создайте экземпляр класса Facade. Facade должен управлять созданием экземпляров и жизненным циклом классов подсистемы, гарантируя, что клиентам не придется иметь дело с этой сложностью.

Шаг 6. Тестирование и доработка
Тщательно протестируйте взаимодействие между клиентским кодом и Facade. Убедитесь, что желаемая функциональность достигнута, не раскрывая тонкостей подсистемы. При необходимости уточните методы Facade или взаимодействие с подсистемой, чтобы оптимизировать производительность, удобство обслуживания или удобство использования.

Шаг 7: Сохраняйте независимость
Помните, что классы подсистемы должны оставаться независимыми от Facade. Это означает, что вы можете изменять или расширять подсистему, не затрагивая Facade или клиентский код. Поддерживайте разделение задач, чтобы сохранить модульность и гибкость кодовой базы.

Шаг 8: Документируйте Facade
Предоставьте четкую документацию для класса Facade, описывающую доступные методы, их параметры и ожидаемое поведение. Эта документация помогает другим разработчикам понять, как правильно использовать Facade.

Шаг 9: Обработка исключений

Рассмотрим, как обрабатывать исключения и ошибки, которые могут возникнуть в подсистеме. Facade должен обеспечивать соответствующую обработку ошибок и обратную связь с клиентами, одновременно защищая их от исключений низкого уровня.

Шаг 10: Мониторинг и обновление
По мере развития вашего приложения периодически проверяйте использование паттерна Facade, чтобы убедиться, что он продолжает соответствовать вашим потребностям. Возможно, вам придется обновить Facade или расширить его функциональность по мере изменения требований вашего приложения.

Шаг 11: Осознайте преимущества
Используя паттерна Facade, вы упростите клиентский код, обеспечите разделение, повысите удобство сопровождения и улучшите читаемость кода.

«Gang of Four» и паттерн проектирования «Facade» являются столпами в мире проектирования и архитектуры программного обеспечения. Новаторская работа над паттернами проектирования предоставила стандартизированный словарь и структуру для решения повторяющихся проблем проектирования. Среди этих паттернов паттерн Facade играет жизненно важную роль в упрощении сложных систем, развитии абстракции и повышении удобства сопровождения и гибкости. Понимание принципов и применения шаблонов GoF и Facade необходимо любому инженеру-программисту, стремящемуся создавать элегантные, удобные в обслуживании и эффективные системы.