Информационных Технологий, Механики и Оптики
ПИиКТ

Лабораторная работа 2
по дисциплине
«Архитектура программных систем»

Выполнил: Мансуров Б.Б.
Группа: P33113
Преподаватель: Перл И.А.

Санкт-Петербург
2020г

Задание

Из списка шаблонов проектирования GoF и GRASP выбрать 3-4 шаблона и для каждого из них придумать 2-3 сценария, для решения которых могу применены выбранные шаблоны.
Сделать предположение о возможных ограничениях, к которым можем привести использование шаблона в каждом описанном случае.
Обязательно выбрать шаблоны из обоих списков.

Что такое Паттерн?

Паттерн проектирования - это часто встречающееся решение определенной проблемы при проектирование архитектуры программ.

Зачем знать Паттерны?

Мы можем вполне успешно работать, не зная ни одного паттерна. Я думаю мы уже не раз реализовали какой - то паттерн, даже не зная об этом.

Но осознанное владение инструментом как раз и отличает профессионала от любителя. Мы можем забить дрелью гвоздь. Но профессионал знает, что главная фишка дрели не забивать гвозди.

Проверенные решения - тратим меньше времени, используя готовые решения, вместо изобретения велосипеда.

Общий программистский язык - чтобы объяснить просто произносим названия паттерна.

Стандартизация кода - делаем меньше просчета при проектировании, используя унифицированные решения, так как все скрытые проблемы были давно найдены.

Выполнение

Абстрактная фабрика (GoF) - это порождающий паттерн проектирования, который позволяет создавать семейства связанных объектов, не привязываясь к конкретным классам создаваемых объектов.

Применимость

1. Когда бизнес - логика программы должна работать с разными видами связанных друг с другом продуктов, не завися от конкретных классов (объектов).

Абстрактная фабрика скрывает от клиента (клиентского кода) подробности того, как и какие объекты будут созданы Но клиентский код может работать со всеми типами создаваемых объектов, поскольку клиент использует общий интерфейс этих классов.

2. Когда в программе нужно добавить новый тип объектов.

В хорошей программе каждый класс отвечает только за одну вещь. Поэтому имеет смысл вынести всю логику создания продуктов в отдельную иерархию классов, применив абстрактную фабрику.

Преимущества и недостатки

- Гарантирует сочетаемость создаваемых объектов (классов).
- Упрощает добавление новых объектов.
- Реализует принцип открытости / закрытости.
- Избавляет клиентский код от привязки к конкретным реализациям.

- Усложняет код программы из - за введения абстракций и множество дополнительных классов.

Мост (GoF) - это структурный паттерн проектирования, который разделяет один или несколько классов на две отдельные иерархии — абстракцию и реализацию, позволяя изменять их независимо друг от друга.

Применимость

1. Когда надо разделить монолитный класс, который содержит несколько различных реализаций какой - то функциональности. Например если

класс работает с разными базами данных, или графический интерфейс работает с разными операционными системами.

Чем больше класс, тем труднее разобраться в нем, и тем больше затягивает разработку. Кроме того изменения вносимые в одну из реализаций приводят к редактированию всего класса, что может привести к внесению разных ошибок случайных и неслучайных.

Паттерн Мост позволяет разделить монолитный класс на отдельные иерархий. После этого можно изменять их независимо друг от друга. Это упрощает разработку и понимания кода.

2. Когда класс нужно расширять в двух независимых направлениях или плоскостях.

Мост позволяет заменить реализацию даже во время выполнения программы, так как можно заменить конкретную реализацию в абстракции.

Преимущества и недостатки

- Позволяет создавать платформо - независимые программы.
- Тут тоже реализуется принцип открытости / закрытости.
- Скрывать лишние и ненужные детали реализации от клиента.

- Тут тоже усложняется код программы из - за введения дополнительных классов.

Controller (GRASP) - это объект прослойка между частями программы, системы и тд. Например это объект прослойке между UI логикой и предметной логикой приложения. Контроллер используется для определений точек входа на каждый уровень подсистемы.

Применимость

1. Когда нам нужно предоставить простой или урезанный интерфейс к сложной подсистеме.

Часто подсистемы усложняются по мере развития программы. Применение большинства паттернов приводит к появлению меньших классов, но в бóльшем количестве. Такую подсистему проще повторно использовать, настраивая её каждый раз под конкретные задачи, но вместе с тем, применять подсистему без настройки становится труднее.Контроллер предлагает определённый вид системы по умолчанию, устраивающий большинство клиентов.

2. Когда надо разложить систему на отдельные слои.

Если системы или подсистемы зависят друг от друга, то зависимость можно упростить, разрешив подсистемам обмениваться информацией только через контроллер.

Преимущества и недостатки

- Изолирует клиентов от компонентов сложной системы.

- Помещение всей ответственности может привести к его возрастанию и усложнение поддержки системы. Контроллер может стать божественным объектом который привязан ко всем классам системы, который может делать все что угодно.

Полиморфизм (GRASP) - если говорить кратко полиморфизм - это способность объектов использовать методы производного класса реализующий определенный интерфейс или наследующий класс.

Он дает возможность трактовать однообразно разные объекты с одинаковым интерфейсом (спецификацией).

Применимость

1. Когда программа должна обрабатывать разнообразные запросы несколькими способами, но заранее неизвестно, какие конкретно запросы будут приходить и какие обработчики для них понадобятся.

Например Допустим у нас есть интерфейс report его реализует два интерфейса generatePDF, generateExcel. И в зависимости от формата отчета мы вызовем функцию generate() PDF или Excel.

2. Когда вам нужно использовать разные вариации какого-то алгоритма внутри одного объекта.

3. Когда у вас есть множество похожих классов, отличающихся только некоторым поведением.

…

Преимущества и недостатки

- Можно сказать реализуется принцип единственной ответственности.
- Убираем прямую зависимость между объектами.

- Про минусы ничего не знаю (может быть система будет работать более медленно).

Вывод

Ну как вы сами объяснили надо подгонять паттерны под задачи а не задачи под паттерны. Но это сложно, для этого нужен опыт. Так как я новичок то если мне дали молоток то мне кажется что все предметы вокруг меня начинают напоминать гвозди. Вникнув в паттерн, новичок и иногда опытные программисты пытаются применить паттерн даже там, где можно было обойтись без паттерна.

Паттерны GoF - можно сказать реализация определенных задач.

А вот паттерны GRASP это нечто другое это как принципы SOLID. Я бы сказал многие паттерны GRASP вещи без которых не обойтись. Это как фундамент любого программирования.

На примере Полиморфизма - решает проблему альтернативных поведений на основе типа. Паттерны как Chain of Responsibility, Strategy, Command ... , все они по сути основываются на полиморфизме.

Можно сказать что GRASP это основа для всех паттернов GoF.

Многие паттерны пересекаются и надо всех их использовать с умом и балансировать их с какими - то костылями под свою задачу, чтобы получить действительно красивые, устойчивые и гибкие архитектуры и приложения.