ООП

Семестр 2. Лекция 5.
Паттерны проектирования

Кафедра ИВТ и ПМ
ЗабГУ

2018

План

Прошлые темы

Шаблоны проектирования

Outline

SOLID

- **S.** Принцип единственной ответственности (The Single Responsibility Principle, SRP)
- **O.** Принцип открытости/закрытости (The Open Closed Principle, OCP)
- **L.** Принцип подстановки Барбары Лисков (The Liskov Substitution Principle, LSP)
- **I.** Принцип разделения интерфейса (The Interface Segregation Principle, ISP)
- **D.** Принцип инверсии зависимостей (The Dependency Inversion Principle, DIP)

Outline

Шаблоны проектирования

Шаблон проектирования или **паттерн** (design pattern) в разработке программного обеспечения — повторяемая архитектурная конструкция, представляющая собой решение проблемы проектирования в рамках некоторого часто возникающего контекста.

Шаблоны проектирования

Алгоритмы по своей сути также являются шаблонами, но не проектирования, а вычисления, так как решают вычислительные задачи.

Достоинства и недостатки

Достоинства

- снижении сложности разработки за счёт готовых абстракций для решения целого класса проблем
- унификация деталей решений: модулей, элементов проекта

Недостатки

- слепое следование выбранному шаблону может привести к усложнению программы
- необоснованное применение применение шаблона

Антипаттерн

Антипаттерн (anti-pattern) — это распространённый подход к решению класса часто встречающихся проблем, являющийся неэффективным, рискованным или непродуктивным

Шаблоны проектирования

- Основные шаблоны (Fundamental)
- Порождающие шаблоны (Creational)
- Структурные шаблоны (Structural)
- Поведенческие шаблоны (Behavioral)

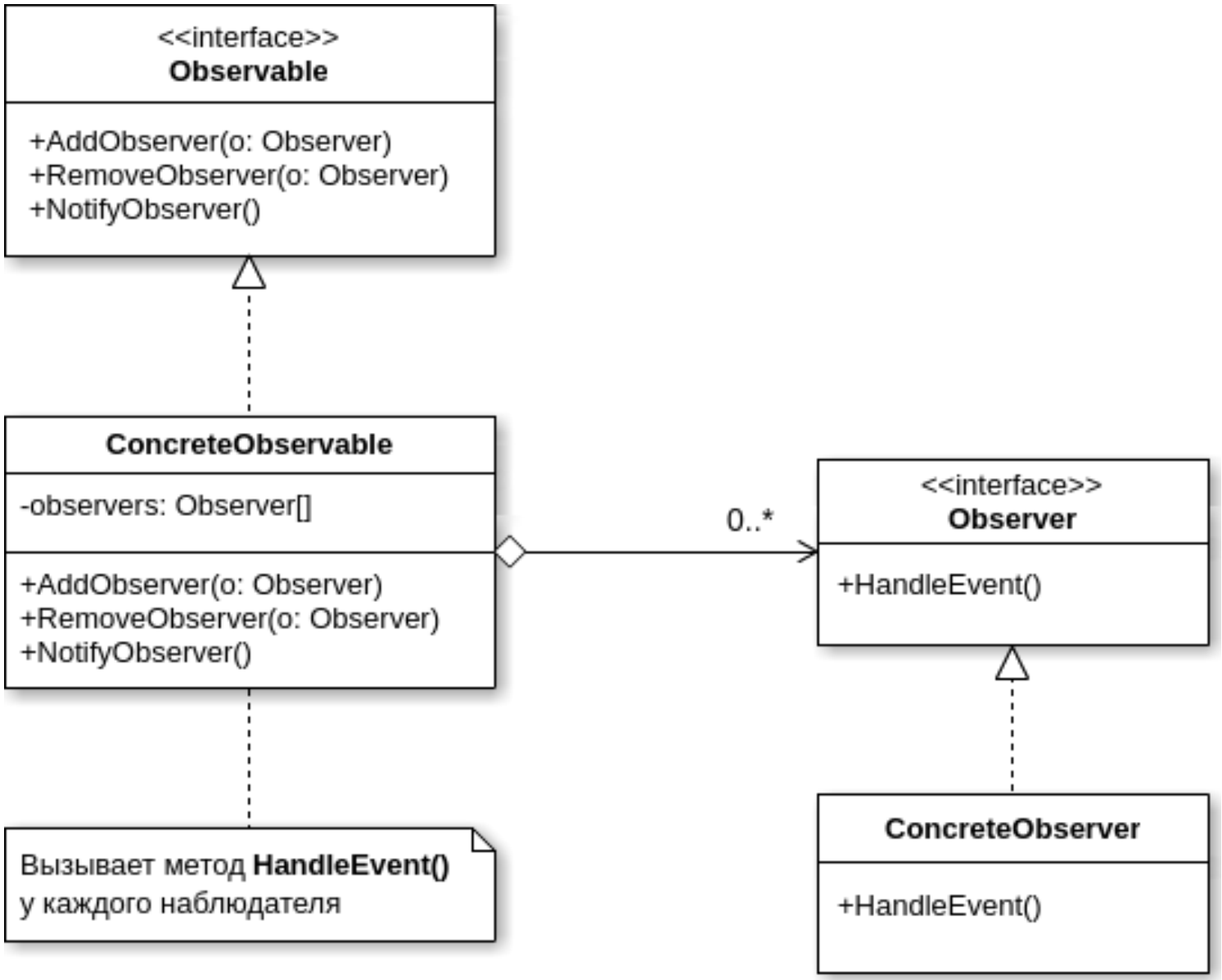
Наблюдатель

Наблюдатель (Observer) — поведенческий шаблон проектирования. Также известен как «подчинённые» (Dependents).

Создает механизм у класса, который позволяет получать экземпляру объекта этого класса оповещения от других объектов об изменении их состояния, тем самым наблюдая за ними

wikipedia - Пример

Наблюдатель



Наблюдатель

- **Observable** — интерфейс, определяющий методы для добавления, удаления и оповещения наблюдателей;
- **Observer** — интерфейс, с помощью которого наблюдатель получает оповещение;
- **ConcreteObservable** — конкретный класс, который реализует интерфейс Observable;
- **ConcreteObserver** — конкретный класс, который реализует интерфейс Observer.

Когда применять?

Если система обладает следующими свойствами:

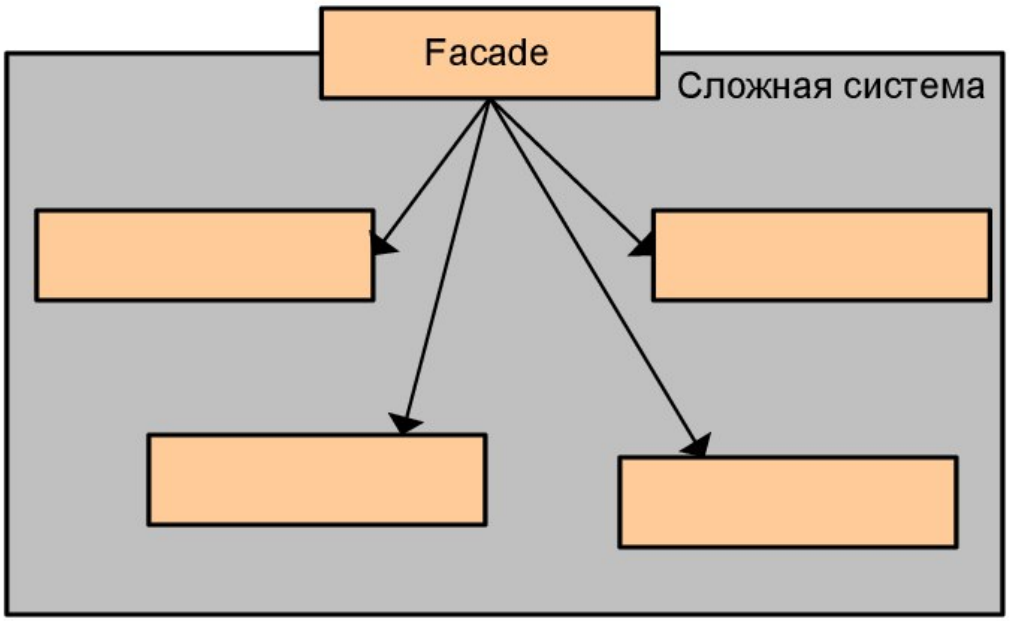
- существует, как минимум, один объект, рассылающий сообщения; имеется не менее одного получателя сообщений, причём их
- количество и состав могут изменяться во время работы приложения;
- нет надобности очень сильно связывать взаимодействующие объекты, что полезно для повторного использования.

Фасад

Шаблон **фасад** (Facade) — структурный шаблон проектирования, позволяющий скрыть сложность системы путём сведения всех возможных внешних вызовов к одному объекту, делегирующему их соответствующим объектам системы.

Пример

Фасад



Фасад
Facade

Тип: Структурный

Что это:

Предоставляет единый интерфейс к группе интерфейсов подсистемы. Определяет высокоуровневый интерфейс, делая подсистему проще для использования.

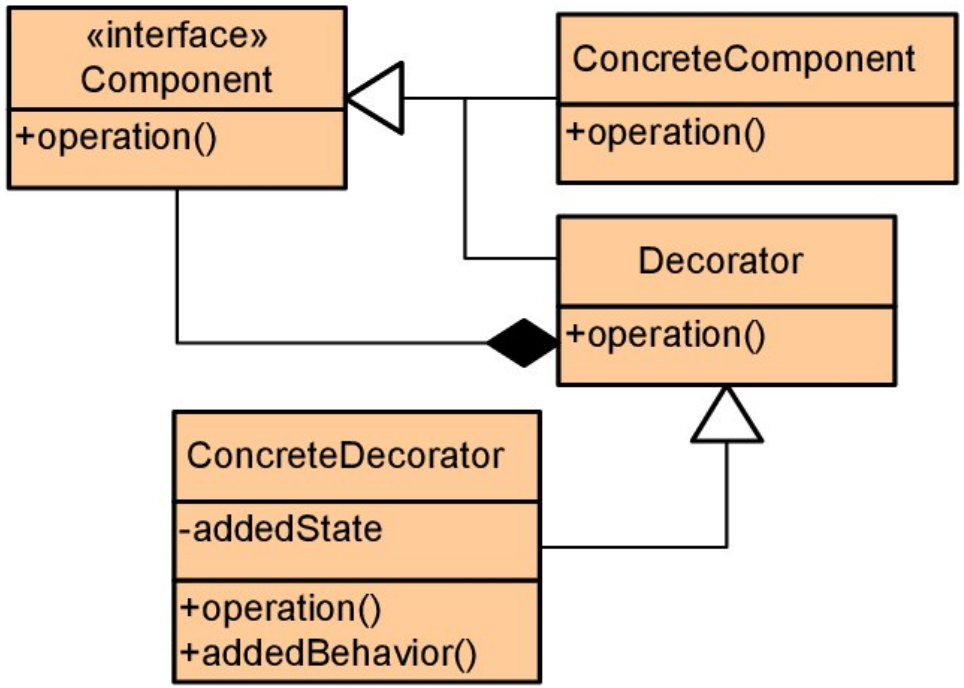
Декоратор

Декоратор (Decorator) — структурный шаблон проектирования, предназначенный для динамического подключения дополнительного поведения к объекту.

Шаблон Декоратор предоставляет гибкую альтернативу практике создания подклассов с целью расширения функциональности.

Пример

Декоратор



Декоратор
Decorator

Тип: Структурный

Что это:

Динамически предоставляет объекту дополнительные возможности. Представляет собой гибкую альтернативу наследованию для расширения функциональности.

Абстрактная фабрика

Абстрактная фабрика (Abstract factory) — порождающий шаблон проектирования, предоставляет интерфейс для создания семейств взаимосвязанных или взаимозависимых объектов, не специфицируя их конкретных классов.

Пример

Абстрактная фабрика

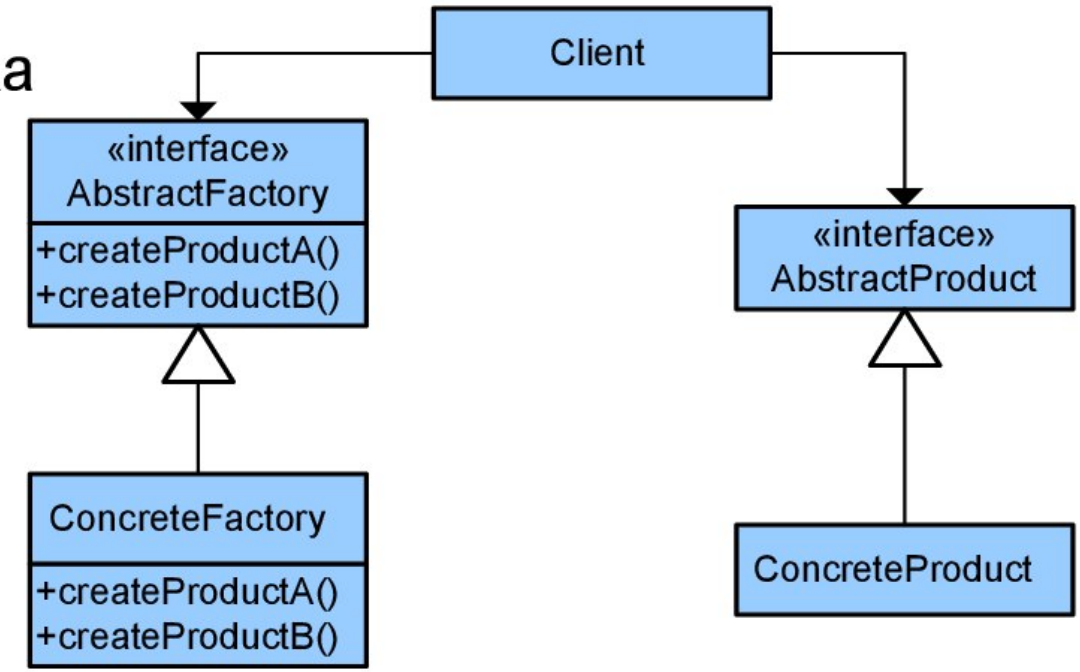
Абстрактная фабрика

Abstract factory

Тип: Порождающий

Что это:

Предоставляет интерфейс для создания групп связанных или зависимых объектов, не указывая их конкретный класс.



Когда применять?

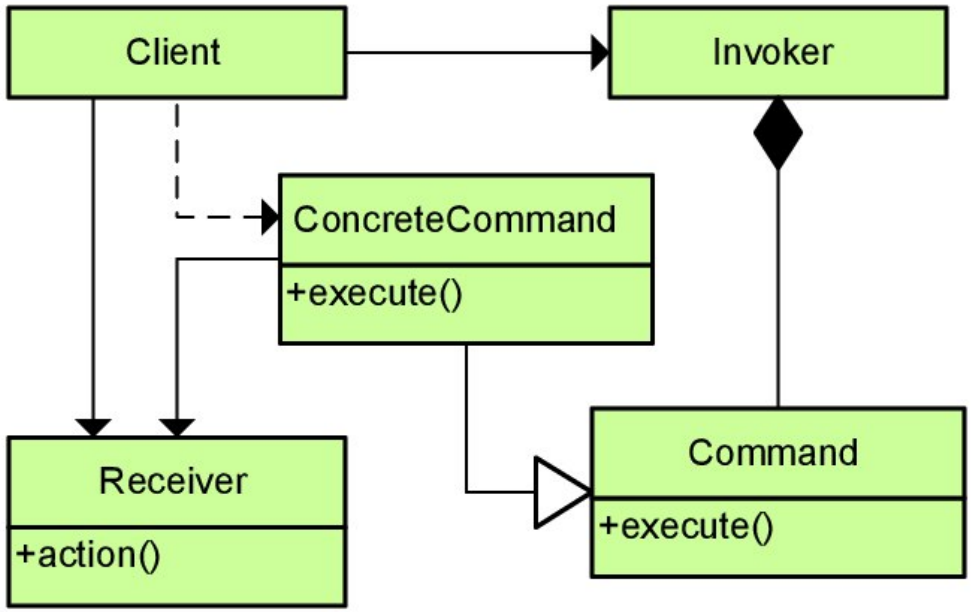
- Когда программа должна быть независимой от процесса и типов создаваемых новых объектов.
- Когда необходимо создать семейства или группы взаимосвязанных объектов исключая возможность одновременного использования объектов из разных этих наборов в одном контексте

Команда

Команда (Command, Action) — поведенческий шаблон проектирования, используемый при объектно-ориентированном программировании, представляющий действие. Объект команды заключает в себе само действие и его параметры.

Пример

Команда



Команда
Command

Тип: Поведенческий

Что это:

Инкапсулирует запрос в виде объекта, позволяя передавать их клиентам в качестве параметров, ставить в очередь, логировать а также поддерживает отмену операций.

Ссылки и литература

1. Фримен Эр., Фримен Эл., Бейтс Б., Сьерра К. "Паттерны проектирования"
2. habrahabr.ru/post/210288 - Шпаргалка по шаблонам проектирования
   См. сылку на pdf в конце статьи.
3. Приёмы объектно-ориентированного проектирования. Паттерны проектирования. «Банда четырёх» (GoF): Эрих Гамма, Ричард Хелм, Ральф Джонсон, Джон Влиссидес.

Ссылки и литература

1. Г. Буч. Объектно-ориентированный анализ и проектирование с примерами приложений. 720 с. 2010 г. 700 страниц. Теория. Примеры на C++. Картинки! Вторая половина книги - примеры OOA и OOD с UML диаграммами.
2. MSDN - Microsoft Developer Network
3. Qt 5.X. Профессиональное программирование на C++. Макс Шлее. 2015 и более поздние издания г. 928 с. Книга периодически обновляется с выходом новых версий фреймворка Qt.
4. www.stackowerflow.com - система вопросов и ответов
5. draw.io — создание диаграмм.

Материалы курса

Слайды, вопросы к экзамену, задания, примеры

github.com/VetrovSV/OOP