Паттерны = Шаблоны проектирования

МКД 03.01 «Технология разработки программного обеспечения»

Определения

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

Шаблон – не законченный образец, не код; это пример решения задачи, который можно использовать в различных ситуациях.

ОО шаблоны показывают отношения и взаимодействия между классами или объектами, без определения того, какие конечные классы или объекты приложения будут использоваться.

Определения

Идиомы – «низкоуровневые» шаблоны, учитывающие специфику конкретного языка программирования. Они не универсальные.

Архитектурные шаблоны – «наивысший» уровень, охватывают архитектуру всей программной системы.

Алгоритмы – шаблоны вычисления, так как решают вычислительные задачи.

История

- 1970-е гг. архитектор Кристофер Александр, набор шаблонов проектирования, не оценили ☹.
- 1987 г. Кент Бэк (Kent Beck) и Вард Каннингем (Ward Cunningham), шаблоны разработки ПО на языке Smalltalk.
- 1988 г. Эрих Гамма (Erich Gamma), докторская диссертация о применении шаблонов к разработке ПО.
- 1989-1991 гг. Джеймс Коплин (James Coplien), идиомы для программирования на C++, книга «Advanced C++ Idioms».
- 1991 г. «Банда четырёх» (*Gang of Four*, *GoF*): Эрих Гамма, Ричард Хелм (Richard Helm), Ральф Джонсон (Ralph Johnson), Джон Влиссидс (John Vlissides), книга «Design Patterns — Elements of Reusable Object-Oriented Software», описаны 23 шаблона проектирования.

Плюсы

- Снижение сложности разработки за счёт готовых абстракций для решения целого класса проблем.
- Облегчение коммуникации между разработчиками, позволяя ссылаться на известные шаблоны.
- Снижение количества ошибок за счет унификации деталей решений (модулей, элементов проекта, …).
- Возможность многократно использовать удачное решение (≈ как использование готовых библиотек кода).
- Возможность выбрать наиболее подходящий вариант проектирования из набора шаблонов.

Минусы

- Слепое следование шаблону может привести к усложнению программы.
- Использование шаблона без особых оснований.
- Шаблоны проектирования в ООП – идиоматическое воспроизведение *элементов* функциональных языков.
- Питер Норвиг: «16 из 23 шаблонов *GoF* в Lisp реализуются существенно проще, чем в C++, либо оказываются незаметны».
- Пол Грэхэм: идея шаблонов проектирования = анти-паттерн, сигнал, что система не обладает достаточным уровнем абстракции, и необходима её тщательная переработка.
- Шаблон = «*готовое решение, но не прямое обращение к библиотеке*» = отказ от повторного использования в пользу дублирования.

Типы шаблонов проектирования: Основные (Fundamental)

Название	Оригинал. название	Описание	Описан в Design Patterns
Шаблон делегирования	Delegation pattern	Объект внешне выражает некоторое поведение, но в реальности передаёт ответственность за выполнение этого поведения связанному объекту	Н/Д
Шаблон функциональ-ного дизайна	Functional design	Гарантирует, что каждый модуль компьютерной программы имеет только одну обязанность и исполняет её с минимумом побочных эффектов на другие части программы	Н/Д
Неизменяемый интерфейс	Immutable interface	Создание неизменяемого объекта	Н/Д
Интерфейс	Interface	Общий метод для структурирования компьютерных программ для того, чтобы их было проще понять	Н/Д
Интерфейс-маркер	Marker interface	В качестве атрибута применяется наличие или отсутствие реализации интерфейса-маркера.	Н/Д
Контейнер свойств	Property Container	Позволяет добавлять дополнительные свойства для класса в контейнер (внутри класса), вместо расширения класса новыми свойствами	Н/Д
Event Channel	Event Channel	Расширяет шаблон Publish/Subscribe, создавая централизованный канал для событий. Использует объект-представитель для подписки и объект-представитель для публикации события в канале. Представитель существует отдельно от реального издателя или подписчика. Подписчик может получать опубликованные события от более чем одного объекта, даже если он зарегистрирован только на одном канале	Н/Д

Типы шаблонов проектирования: Порождающие (Creational)

Абстрагируют процесс инстанцирования. Позволяют сделать систему независимой от способа создания, композиции и представления объектов. Шаблон, порождающий классы, использует наследование, чтобы изменять инстанцируемый класс, а шаблон, порождающий объекты, делегирует инстанцирование другому объекту

Название	Описание
Абстрактная фабрика (Abstract factory)	Класс, который представляет собой интерфейс для создания компонентов системы
Строитель (Builder)	Класс, который представляет собой интерфейс для создания сложного объекта
Фабричный метод (Factory method)	Определяет интерфейс для создания объекта, но оставляет подклассам решение о том, какой класс инстанцировать
Отложенная инициали-зация (Lazy initialization)	Объект, инициализируемый во время первого обращения к нему
Пул одиночек (Multiton)	Гарантирует, что класс имеет поименованные экземпляры объекта и обеспечивает глобальную точку доступа к ним
Объектный пул (Object pool)	Класс, который представляет собой интерфейс для работы с набором инициализированных и готовых к использованию объектов
Прототип (Prototype)	Определяет интерфейс создания объекта через клонирование другого объекта вместо создания через конструктор
Получение ресурса есть инициализация (Resour-ce acquisition is initialization)	Получение некоторого ресурса совмещается с инициализацией, а освобождение — с уничтожением объекта
Одиночка (Singleton)	Класс, который может иметь только один экземпляр

Типы шаблонов проектирования: Структурные (Structural)

Определяют различные сложные структуры, которые изменяют интерфейс уже существующих объектов или его реализацию, позволяя облегчить разработку и оптимизировать программу

Название	Описание
Адаптер (Adapter / Wrapper)	Объект, обеспечивающий взаимодействие двух других объектов, один из которых использует, а другой предоставляет несовместимый с первым интерфейс
Мост (Bridge)	Структура, позволяющая изменять интерфейс обращения и интерфейс реализации класса независимо Да Компоновщик Composite Объект, который объединяет в себе объекты, подобные ему самому
Декоратор или Обёртка (Wrapper/ Decorator)	Класс, расширяющий функциональность другого класса без использования наследования
Фасад (Facade)	Объект, который абстрагирует работу с несколькими классами, объединяя их в единое целое
Единая точка входа (Front Controller)	Обеспечивает унифицированный интерфейс для интерфейсов в подсистеме. Front Controller определяет высокоуровневый интерфейс, упрощающий использование подсистемы
Приспособленец (Flyweight)	Это объект, представляющий себя как уникальный экземпляр в разных местах программы, но по факту не являющийся таковым
Заместитель (Proxy)	Объект, который является посредником между двумя другими объектами, и который реализует/ограничивает доступ к объекту, к которому обращаются через него

Типы шаблонов проектирования: Поведенческие (Behavioral)

определяют взаимодействие между объектами, увеличивая таким образом его гибкость

Название	Описание
Цепочка обязанностей (Chain of responsibility)	Предназначен для организации в системе уровней ответственности
Команда (Command, Action, Transaction)	Представляет действие. Объект команды заключает в себе само действие и его параметры
Интерпретатор (Interpreter)	Решает часто встречающуюся, но подверженную изменениям, задачу
Итератор (Iterator, Cursor)	Представляет собой объект, позволяющий получить последовательный доступ к элементам объекта-агрегата без использования описаний каждого из объектов, входящих в состав агрегации
Посредник (Mediator)	Обеспечивает взаимодействие множества объектов, формируя при этом слабую связанность и избавляя объекты от необходимости явно ссылаться друг на друга
Хранитель (Memento, Token)	Позволяет не нарушая инкапсуляцию зафиксировать и сохранить внутренние состояния объекта так, чтобы позднее восстановить его в этих состояниях
Null Object	Предотвращает нулевые указатели, предоставляя объект «по умолчанию»

Типы шаблонов проектирования: Поведенческие (Behavioral)

Название	Описание
Наблюдатель (Observer, Publish-Subscribe,Listener)	Определяет зависимость типа «один ко многим» между объектами таким образом, что при изменении состояния одного объекта все зависящие от него оповещаются об этом событии
Слуга (Servant)	Используется для обеспечения общей функциональности группе классов
Спецификация (Specification)	Служит для связывания бизнес-логики
Состояние (State, Objects for States)	Используется в тех случаях, когда во время выполнения программы объект должен менять свое поведение в зависимости от своего состояния
Стратегия (Strategy)	Предназначен для определения семейства алгоритмов, инкапсуляции каждого из них и обеспечения их взаимозаменяемости
Шаблонный метод (Template method)	Определяет основу алгоритма и позволяет наследникам переопределять некоторые шаги алгоритма, не изменяя его структуру в целом
Посетитель (Visitor)	Описывает операцию, которая выполняется над объектами других классов. При изменении класса Visitor нет необходимости изменять обслуживаемые классы
Одноразовый посетитель (Single-serving visitor)	Оптимизирует реализацию шаблона посетитель, который инициализируется, единожды используется, и затем удаляется
Иерархический посетитель (Hierarchical visitor)	Предоставляет способ обхода всех вершин иерархической структуры данных (напр. древовидной)

Типы шаблонов проектирования: Частные

Шаблоны параллельного программирования - используются для более эффективного написания многопоточных программ, и предоставляет готовые решения проблем синхронизации.

Active Object Служит для отделения потока выполнения метода от потока, в котором он был вызван. Использует шаблоны асинхронный вызов методов и планировщик

Balking Служит для выполнения действия над объектом только тогда, когда тот находится в корректном состоянии.

Обмен сообщениями (Messaging design pattern, MDP) Позволяет компонентам и приложениям обмениваться информацией (сообщениями)

Шаблоны архитектуры системы

- Model-View-Controller (MVC)
- Модель-представление-контроллер
- Model-View-Presenter
- Model-View-View Model
- Presentation-Abstraction-Control

Типы шаблонов проектирования: Прочие

Carrier Rider Mapper описывают предоставление доступа к хранимой информации

Аналитические шаблоны описывают основной подход для составления требований для программного обеспечения (requirement analysis) до начала самого процесса программной разработки

Коммуникационные шаблоны описывают процесс общения между отдельными участниками/сотрудниками организации

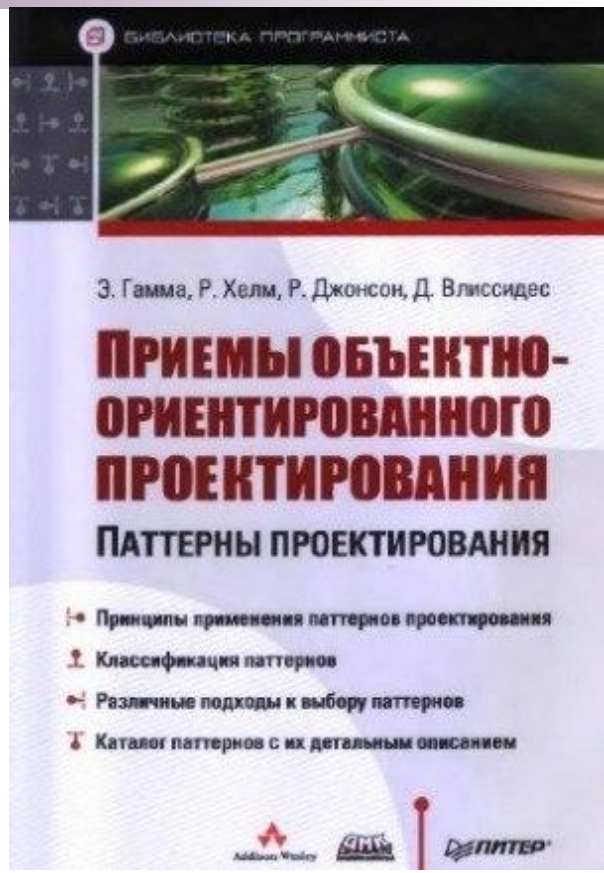
Организационные шаблоны описывают организационную иерархию предприятия/фирмы

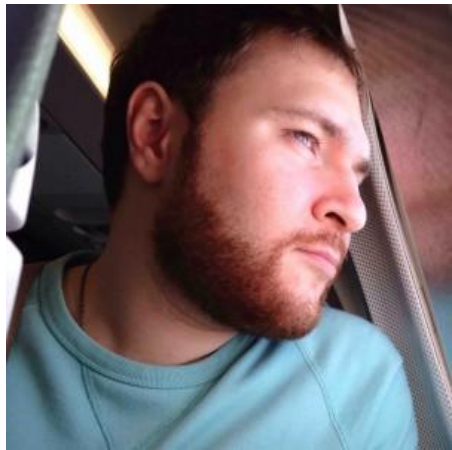
Анти-паттерны (Anti-Design-Patterns) описывают, как не следует поступать при разработке программ, показывая характерные ошибки в дизайне и в реализации

Что почитать?

Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования

Курс Гергия Могелашвили: https://glamcoder.ru/video/design-patterns/





Еще раз коротко

- **Паттерны = Шаблоны проектирования** — это проверенные и готовые к использованию решения часто возникающих в повседневном программировании задач.
- Это не класс и не библиотека, которую можно подключить к проекту, это нечто большее.
- Шаблон проектирования, подходящий под задачу, реализуется в каждом конкретном случае.
- Шаблон не зависит от языка программирования. Хороший шаблон легко реализуется в большинстве, если не во всех языках, в зависимости от выразительных средств языка.
- Шаблон, будучи примененным неправильно или к неподходящей задаче, может принести немало проблем.
- Правильно примененный шаблон поможет решить задачу легко и просто.

Еще раз коротко

Существует три типа шаблонов:

- структурные;
- порождающие;
- поведенческие.

Структурные шаблоны определяют отношения между классами и объектами, позволяя им работать совместно.

Порождающие шаблоны предоставляют механизмы инициализации, позволяя создавать объекты удобным способом.

Поведенческие шаблоны используются для того, чтобы упростить взаимодействие между сущностями.

Спасибо за внимание!

Паттерны = Шаблоны проектирования