Основы разработки ПО

Шаблоны проектирования, спецификация, кодогенерация

Кулаков Кирилл Александрович

Что мы знаем?

- Методы проектирования ПО
  - Законы проектирования
  - Методы проектирования
    - Метод декомпозиции!!!
    - Методы конструирования!!!
- Визуальное моделирование
  - Объектные модели
  - Структурные диаграммы
  - Язык UML

Решение задач проектирования

- Аксиома:
  1) Большинство задач (структурных или вычислительных) проектирования однотипны
  2) Если задача сложная, то ее можно разбить на более мелкие и см. п. 1
- Однотипная задача — устоявшееся решение
  - Повторное использование найденных решений в будущем
- **Шаблон проектирования** или **паттерн** (англ. design pattern) — повторяемая архитектурная конструкция, представляющая собой решение проблемы проектирования в рамках некоторого часто возникающего контекста
- Решение вычислительной задачи = **Алгоритм — шаблон вычисления**

Истоки

- 1987 год, Кент Бэк, Вард Каннингем — шаблоны для разработки графических оболочек на языке Smalltalk
- 1988 год, докторская диссертация Эриха Гамма о переносимости архитектурных шаблонов на разработку программ
- 1991 год, Джеймс Коплин, «**Advanced C++ Idioms**»
- 1991 год, Эрих Гамма, Ричард Хелм, Ральф Джонсон, Джон Влиссидес «**Design Patterns — Elements of Reusable Object-Oriented Software**»
  - 23 шаблона проектирования

Достоинства/недостатки

- Снижение сложности разработки за счет готовых абстракций
- Облегчение коммуникаций между разработчиками
- Унификация деталей решений
  - Снижение числа проектных ошибок
- Повторное использование в случае удачного применения

- Слепое следование некоторому выбранному шаблону может привести к усложнению программы
- У разработчика может возникнуть желание попробовать некоторый шаблон в деле без особых оснований
- Шаблон = отказ от повторного использования в пользу дублирования

Уровни шаблонов

- **Архитектурные шаблоны** — описание структурной схемы программы в целом
  - Пример: парадигма «модель-представление-контроллер» (model-view-controller — MVC)
- **Шаблоны проектирования** - схемы детализации программных подсистем и их отношений между собой
  - Не влияют на структуру программы в целом
  - Не зависят от языка
- **Идиомы** - «Низкоуровневые» шаблоны, учитывающие специфику конкретного языка программирования
  - Пример: интеллектуальные указатели для отслеживания динамически выделенной памяти

Виды шаблонов

- **Поведенческие** (behavioral)
  - Алгоритмы и методы реализации взаимодействия объектов
  - Снижение уровня связности системы
- **Порождающие** (creational)
  - Определение правил создания объектов
  - Снижение зависимости от формирования композиции объектов
- **Структурные** (structural)
  - Определение структур высокого уровня
  - Определение отношений между субъектами структур

Поведенческие шаблоны

- **Хранитель**
  - позволяющий, не нарушая инкапсуляцию, зафиксировать и сохранить внутреннее состояние объекта так, чтобы позднее восстановить его в это состояние
  - Необходимо сохранять состояние объекта и восстанавливать его
  - Открытие доступа ко внутренностям (прямой интерфейс) нарушает принцип инкапсуляции

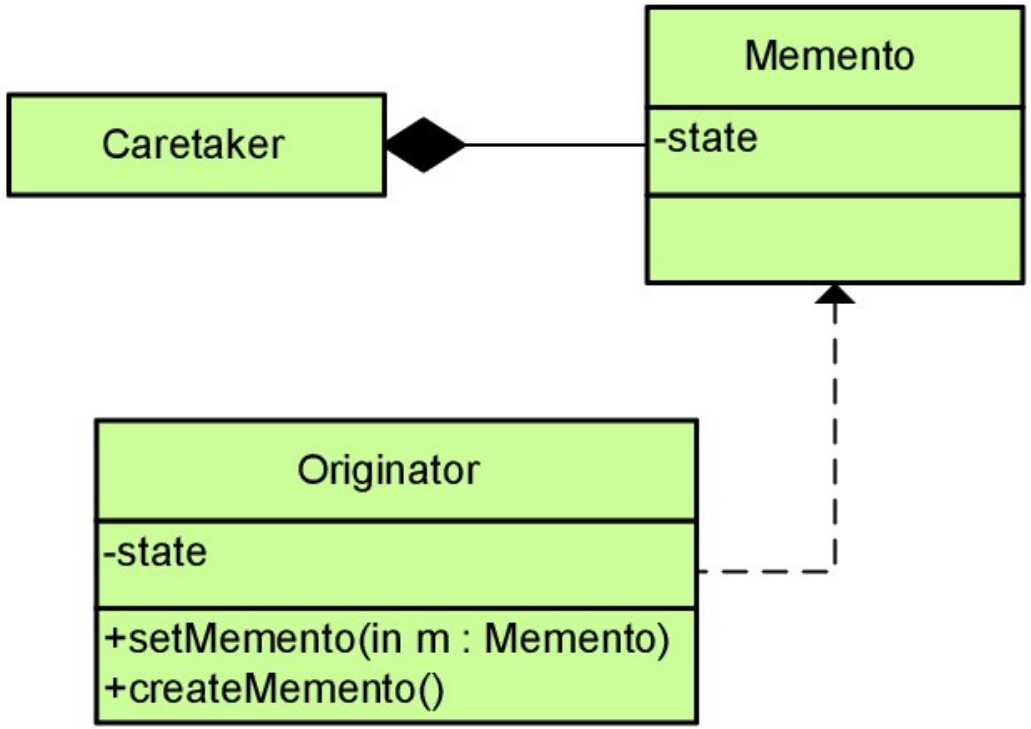
Поведенческие шаблоны: Хранитель

Хранитель
Memento

Тип: Поведенческий

Что это:

Не нарушая инкапсуляцию, определяет и сохраняет внутреннее состояние объекта и позволяет позже восстановить объект в этом состоянии.



композиция (composition) — подвид агрегации, в которой «части» не могут существовать отдельно от «целого».

агрегация (aggregation) — описывает связь «часть»–«целое», в котором «часть» может существовать отдельно от «целого». Ромб указывается со стороны «целого»

зависимость (dependency) — изменение в одной сущности (независимой) может влиять на состояние или поведение другой сущности (зависимой). Со стороны стрелки указывается независимая сущность

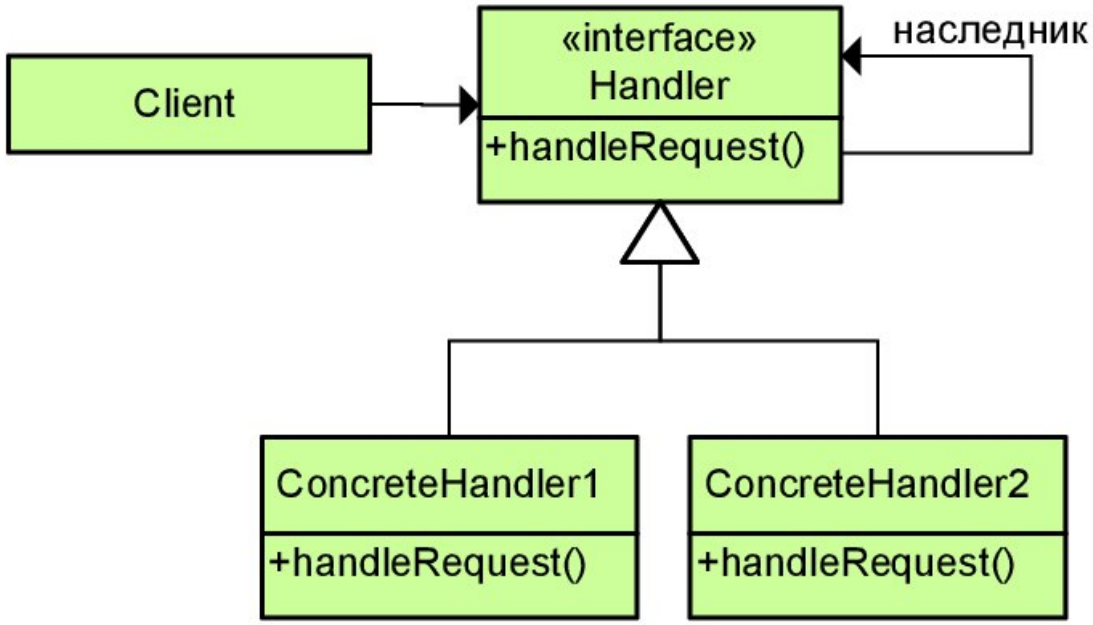
обобщение (generalization) — отношение наследования или реализации интерфейса. Со стороны стрелки находится суперкласс или интерфейс

Источник: https://habr.com/ru/post/210288/

Поведенческие шаблоны

- **Цепочка обязанностей**
  - поведенческий шаблон проектирования, предназначенный для организации в системе уровней ответственности
  - в разрабатываемой системе имеется группа объектов, которые могут обрабатывать сообщения определенного типа
  - все сообщения должны быть обработаны хотя бы одним объектом системы
  - сообщения в системе обрабатываются по схеме «обработай сам либо перешли другому», то есть одни сообщения обрабатываются на том уровне, где они получены, а другие пересылаются объектам иного уровня

Поведенческие шаблоны



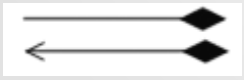
Цепочка обязанностей

Chain of responsibility

Тип: Поведенческий

Что это:

Избегает связывания отправителя запроса с его получателем, давая возможность обработать запрос более чем одному объекту. Связывает объекты-получатели и передаёт запрос по цепочке пока объект не обработает его.



композиция (composition) — подвид агрегации, в которой «части» не могут существовать отдельно от «целого».

агрегация (aggregation) — описывает связь «часть»–«целое», в котором «часть» может существовать отдельно от «целого». Ромб указывается со стороны «целого»

зависимость (dependency) — изменение в одной сущности (независимой) может влиять на состояние или поведение другой сущности (зависимой). Со стороны стрелки указывается независимая сущность

обобщение (generalization) — отношение наследования или реализации интерфейса. Со стороны стрелки находится суперкласс или интерфейс

Источник: https://habr.com/ru/post/210288/

Поведенческие шаблоны

- **Наблюдатель**
  - Реализует у класса механизм, который позволяет объекту этого класса получать оповещения об изменении состояния других объектов и тем самым наблюдать за ними
  - существует как минимум один объект, рассылающий сообщения;
  - имеется не менее одного получателя сообщений, причём их количество и состав могут изменяться во время работы приложения;
  - позволяет избежать сильного зацепления взаимодействующих классов.

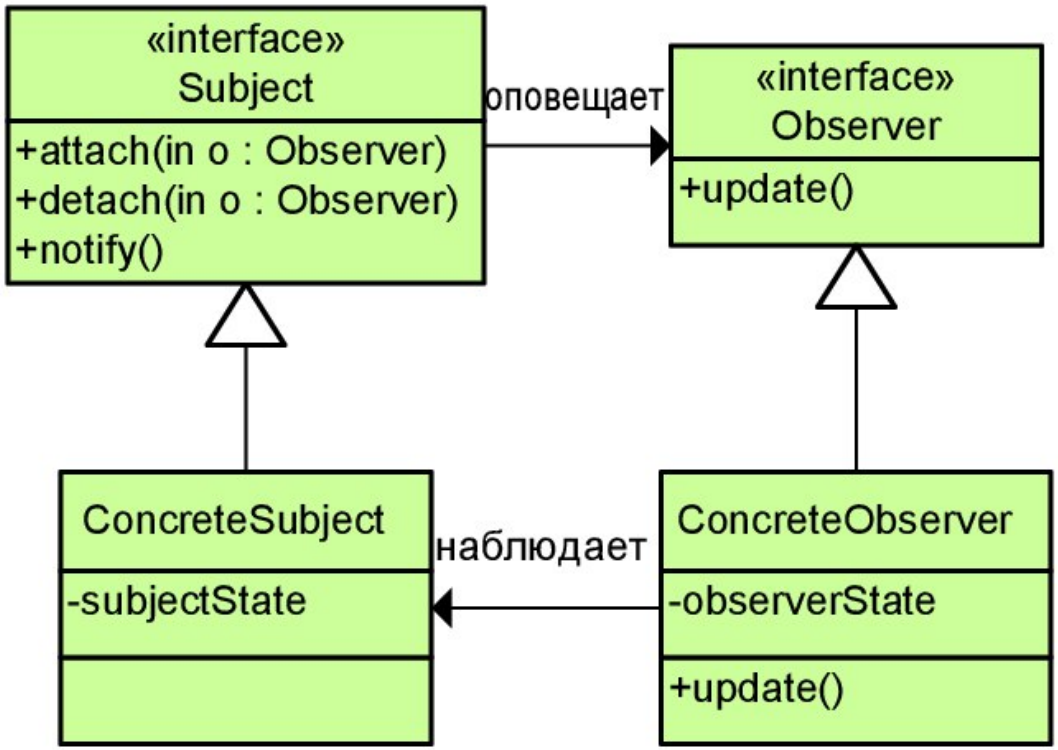
Поведенческие шаблоны

Наблюдатель
Observer

Тип: Поведенческий

Что это:

Определяет зависимость “один ко многим” между объектами так, что когда один объект меняет своё состояние, все зависимые объекты оповещаются и обновляются автоматически.



композиция (composition) — подвид агрегации, в которой «части» не могут существовать отдельно от «целого».

агрегация (aggregation) — описывает связь «часть»–«целое», в котором «часть» может существовать отдельно от «целого». Ромб указывается со стороны «целого»

зависимость (dependency) — изменение в одной сущности (независимой) может влиять на состояние или поведение другой сущности (зависимой). Со стороны стрелки указывается независимая сущность

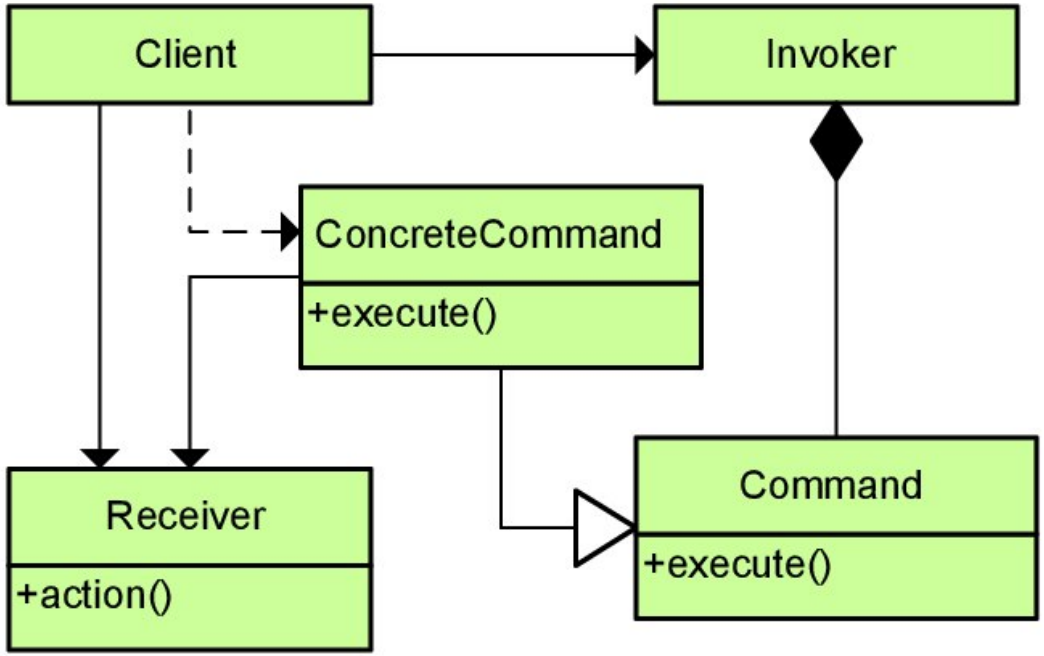
обобщение (generalization) — отношение наследования или реализации интерфейса. Со стороны стрелки находится суперкласс или интерфейс

Источник: https://habr.com/ru/post/210288/

Поведенческие шаблоны

- **Команда**
  - Создание структуры, в которой класс-отправитель и класс-получатель не зависят друг от друга напрямую. Организация обратного вызова к классу, который включает в себя класс-отправитель
  - Обработка воздействий на элементы GUI (кнопки)
  - Отмена операций (Undo)
  - Пулы потоков

Поведенческие шаблоны



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.

композиция (composition) — подвид агрегации, в которой «части» не могут существовать отдельно от «целого».

агрегация (aggregation) — описывает связь «часть»–«целое», в котором «часть» может существовать отдельно от «целого». Ромб указывается со стороны «целого»

зависимость (dependency) — изменение в одной сущности (независимой) может влиять на состояние или поведение другой сущности (зависимой). Со стороны стрелки указывается независимая сущность

обобщение (generalization) — отношение наследования или реализации интерфейса. Со стороны стрелки находится суперкласс или интерфейс

Источник: https://habr.com/ru/post/210288/

Поведенческие шаблоны

- **Состояние**
  - Используется в тех случаях, когда во время выполнения программы объект должен менять своё поведение в зависимости от своего состояния
  - Применение данного паттерна может быть затруднено, если состояния должны обмениваться данными, или одно состояние настраивает свойства другого. В этом случае понадобится глобальный объект, что не очень хорошее архитектурное решение

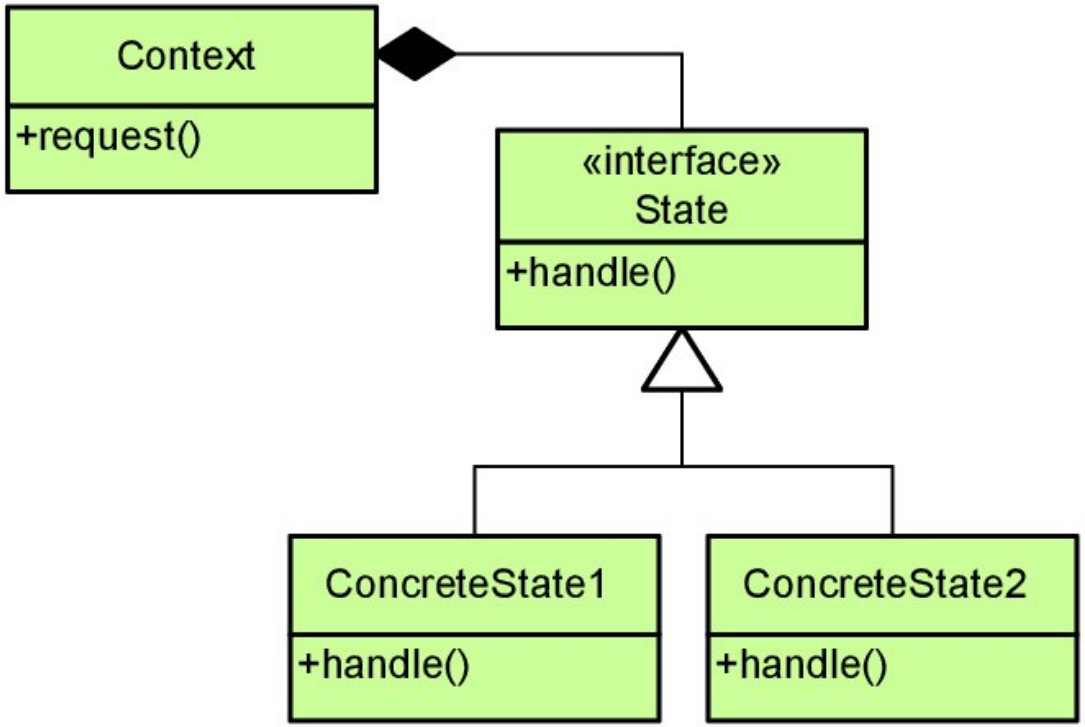
Поведенческие шаблоны

Состояние
State

Тип: Поведенческий

Что это:

Позволяет объекту изменять своё поведение в зависимости от внутреннего состояния.



композиция (composition) — подвид агрегации, в которой «части» не могут существовать отдельно от «целого».

агрегация (aggregation) — описывает связь «часть»–«целое», в котором «часть» может существовать отдельно от «целого». Ромб указывается со стороны «целого»

зависимость (dependency) — изменение в одной сущности (независимой) может влиять на состояние или поведение другой сущности (зависимой). Со стороны стрелки указывается независимая сущность

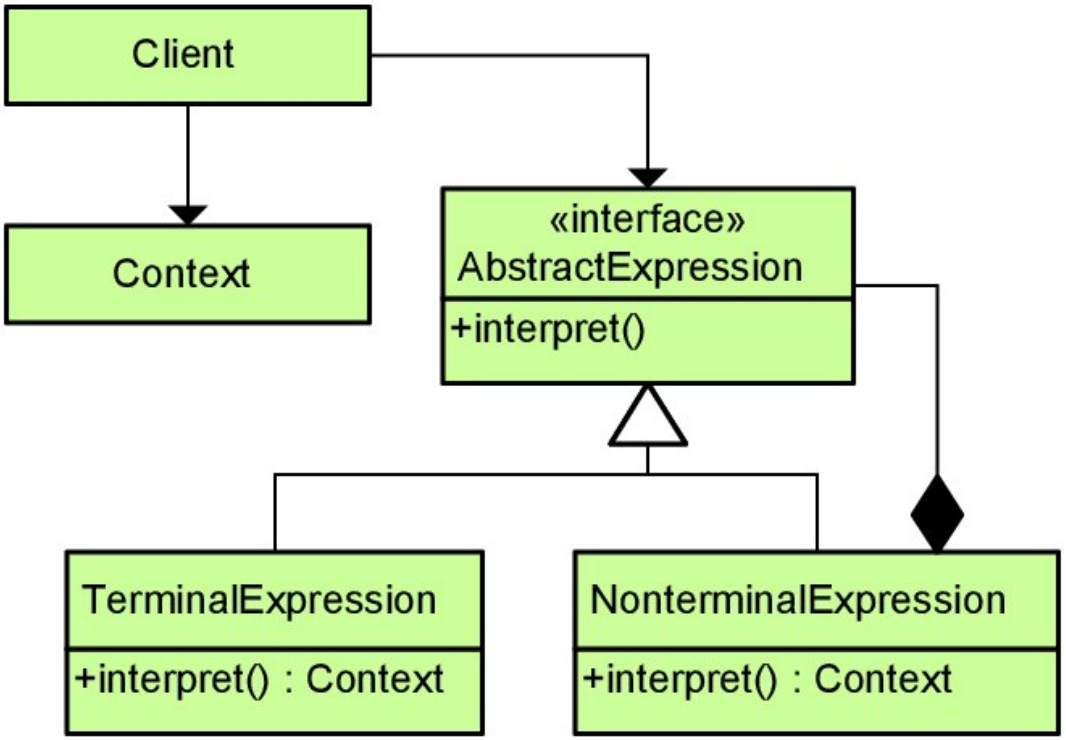
обобщение (generalization) — отношение наследования или реализации интерфейса. Со стороны стрелки находится суперкласс или интерфейс

Источник: https://habr.com/ru/post/210288/

Поведенческие шаблоны

- **Интерпретатор**
  - шаблон проектирования, решающий часто встречающуюся, но подверженную изменениям, задачу
  - Грамматику становится легко расширять и изменять, реализации классов, описывающих узлы абстрактного синтаксического дерева похожи (легко кодируются). Можно легко изменять способ вычисления выражений
  - Сопровождение грамматики с большим числом правил затруднительно

Поведенческие шаблоны



Интерпретатор
Interpreter

Тип: Поведенческий

Что это:

Получая формальный язык, определяет представление его грамматики и интерпретатор, использующий это представление для обработки выражений языка.

композиция (composition) — подвид агрегации, в которой «части» не могут существовать отдельно от «целого».

агрегация (aggregation) — описывает связь «часть»–«целое», в котором «часть» может существовать отдельно от «целого». Ромб указывается со стороны «целого»

зависимость (dependency) — изменение в одной сущности (независимой) может влиять на состояние или поведение другой сущности (зависимой). Со стороны стрелки указывается независимая сущность

обобщение (generalization) — отношение наследования или реализации интерфейса. Со стороны стрелки находится суперкласс или интерфейс

Источник: https://habr.com/ru/post/210288/

Поведенческие шаблоны

- **Стратегия**
  - определения семейства алгоритмов, инкапсуляции каждого из них и обеспечения их взаимозаменяемости. Это позволяет выбирать алгоритм путём определения соответствующего класса.
  - Наличие нескольких алгоритмов / реализаций
  - Вызов всех алгоритмов должен осуществляться стандартным образом (все они должны иметь одинаковый интерфейс)

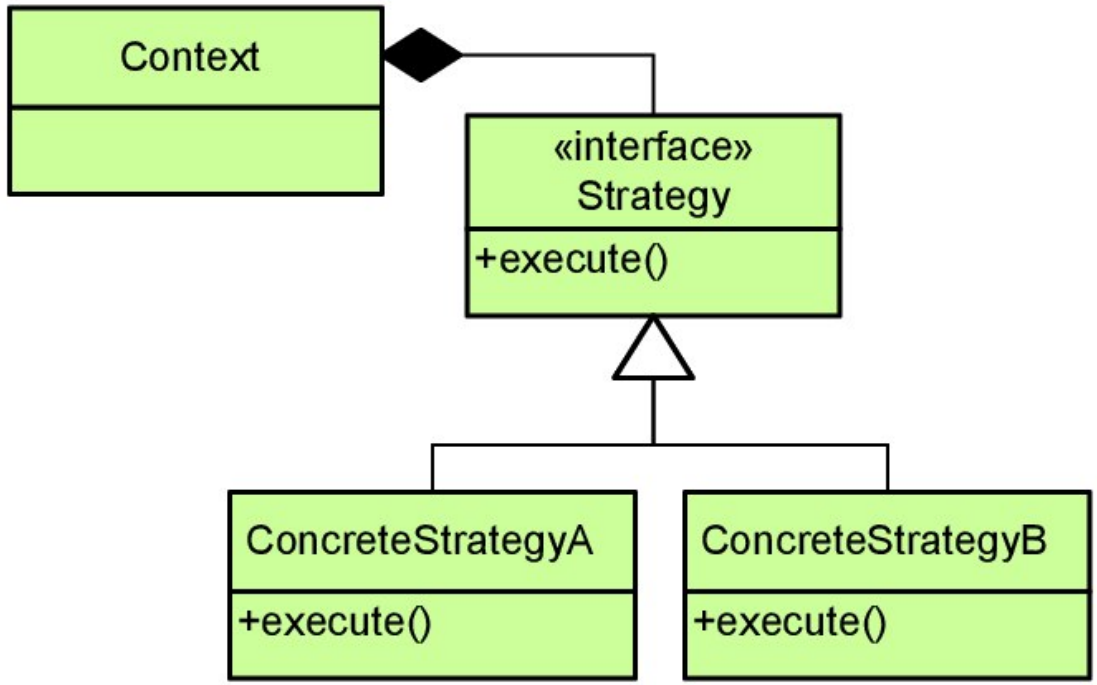
Поведенческие шаблоны

Стратегия
Strategy

Тип: Поведенческий

Что это:

Определяет группу алгоритмов, инкапсулирует их и делает взаимозаменяемыми. Позволяет изменять алгоритм независимо от клиентов, его использующих.



композиция (composition) — подвид агрегации, в которой «части» не могут существовать отдельно от «целого».

агрегация (aggregation) — описывает связь «часть»–«целое», в котором «часть» может существовать отдельно от «целого». Ромб указывается со стороны «целого»

зависимость (dependency) — изменение в одной сущности (независимой) может влиять на состояние или поведение другой сущности (зависимой). Со стороны стрелки указывается независимая сущность

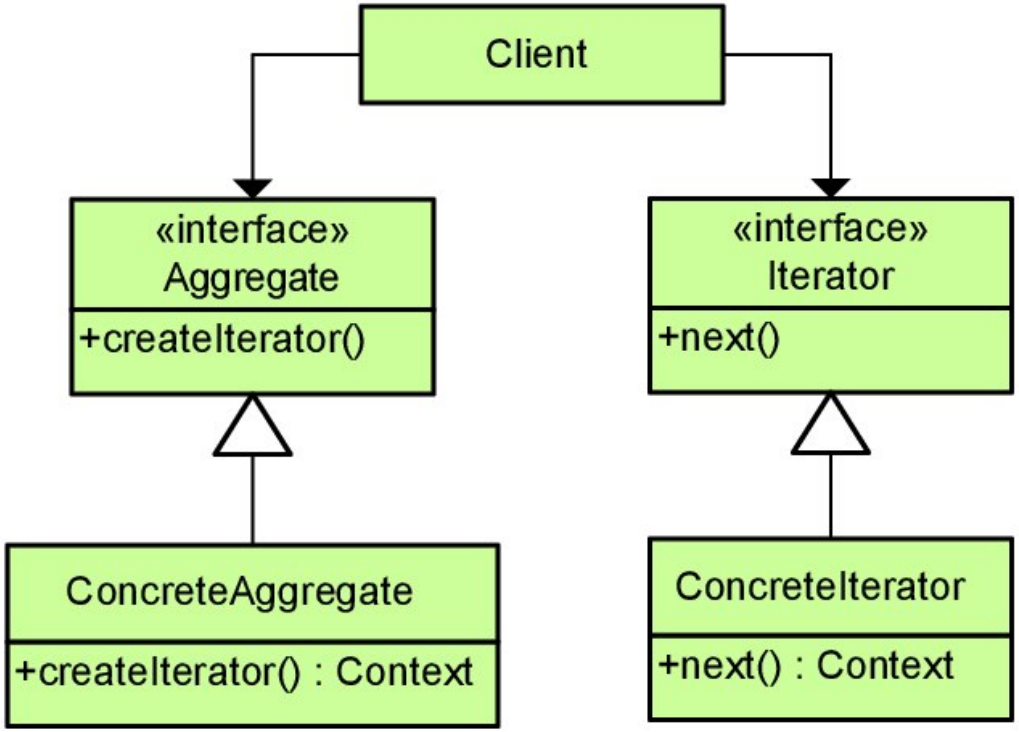
обобщение (generalization) — отношение наследования или реализации интерфейса. Со стороны стрелки находится суперкласс или интерфейс

Источник: https://habr.com/ru/post/210288/

Поведенческие шаблоны

- **Итератор**
  - Представляет собой объект, позволяющий получить последовательный доступ к элементам объекта-агрегата без использования описаний каждого из агрегированных объектов
  - Перебор элементов выполняется объектом итератора, а не самой коллекцией
  - Особенностью полноценно реализованного итератора является то, что код, использующий итератор, может ничего не знать о типе итерируемого агрегата

Поведенческие шаблоны



Итератор
Iterator

Тип: Поведенческий

Что это:

Предоставляет способ последовательного доступа к элементам множества, независимо от его внутреннего устройства.

композиция (composition) — подвид агрегации, в которой «части» не могут существовать отдельно от «целого».

агрегация (aggregation) — описывает связь «часть»–«целое», в котором «часть» может существовать отдельно от «целого». Ромб указывается со стороны «целого»

зависимость (dependency) — изменение в одной сущности (независимой) может влиять на состояние или поведение другой сущности (зависимой). Со стороны стрелки указывается независимая сущность

обобщение (generalization) — отношение наследования или реализации интерфейса. Со стороны стрелки находится суперкласс или интерфейс

Источник: https://habr.com/ru/post/210288/

Поведенческие шаблоны

- **Шаблонный метод**
  - Определяет основу алгоритма и позволяет наследникам переопределять некоторые шаги алгоритма, не изменяя его структуру в целом
  - Однократное использование инвариантной части алгоритма, с оставлением изменяющейся части на усмотрение наследникам.
  - Локализация и вычленение общего для нескольких классов кода для избегания дублирования.
  - Разрешение расширения кода наследниками только в определенных местах

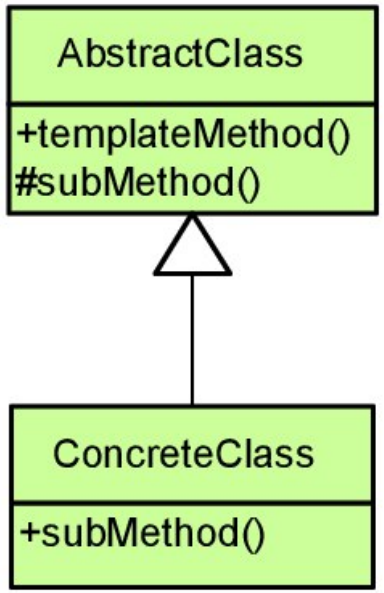
Поведенческие шаблоны

Шаблонный метод
Template method

Тип: Поведенческий

Что это:

Определяет алгоритм, некоторые этапы которого делегируются подклассам. Позволяет подклассам переопределить эти этапы, не меняя структуру алгоритма.



композиция (composition) — подвид агрегации, в которой «части» не могут существовать отдельно от «целого».

агрегация (aggregation) — описывает связь «часть»–«целое», в котором «часть» может существовать отдельно от «целого». Ромб указывается со стороны «целого»

зависимость (dependency) — изменение в одной сущности (независимой) может влиять на состояние или поведение другой сущности (зависимой). Со стороны стрелки указывается независимая сущность

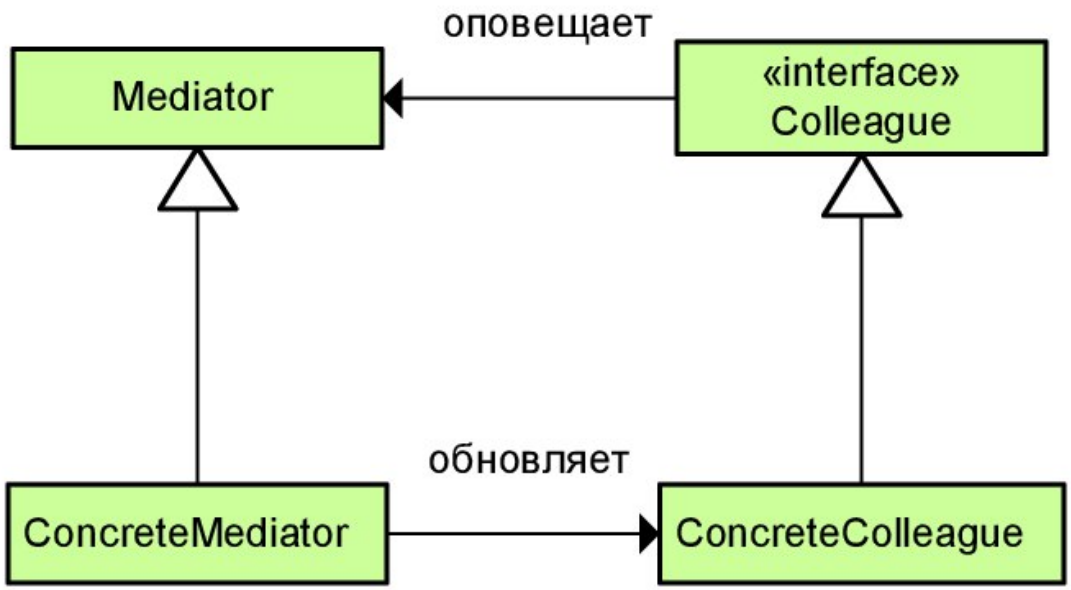
обобщение (generalization) — отношение наследования или реализации интерфейса. Со стороны стрелки находится суперкласс или интерфейс

Источник: https://habr.com/ru/post/210288/

Поведенческие шаблоны

- **Посредник**
  - Обеспечивает взаимодействие множества объектов, формируя при этом слабое зацепление и избавляя объекты от необходимости явно ссылаться друг на друга
  - Создание объекта, инкапсулирующего способ взаимодействия множества объектов
  - Устраняется связанность между "Коллегами", централизуется управление

Поведенческие шаблоны

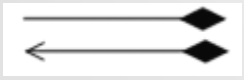


Посредник
Mediator

Тип: Поведенческий

Что это:

Определяет объект, инкапсулирующий способ взаимодействия объектов. Обеспечивает слабую связь, избавляя объекты от необходимости прямо ссылать друг на друга и даёт возможность независимо изменять их взаимодействие.



композиция (composition) — подвид агрегации, в которой «части» не могут существовать отдельно от «целого».

агрегация (aggregation) — описывает связь «часть»–«целое», в котором «часть» может существовать отдельно от «целого». Ромб указывается со стороны «целого»

зависимость (dependency) — изменение в одной сущности (независимой) может влиять на состояние или поведение другой сущности (зависимой). Со стороны стрелки указывается независимая сущность

обобщение (generalization) — отношение наследования или реализации интерфейса. Со стороны стрелки находится суперкласс или интерфейс

Источник: https://habr.com/ru/post/210288/

Поведенческие шаблоны

- **Посетитель**
  - Описывает операцию, которая выполняется над объектами других классов. При изменении visitor нет необходимости изменять обслуживаемые классы
  - имеются различные объекты разных классов с разными интерфейсами, но над ними нужно совершать операции, зависящие от конкретных классов;
  - необходимо над структурой выполнить различные, усложняющие структуру операции;
  - часто добавляются новые операции над структурой

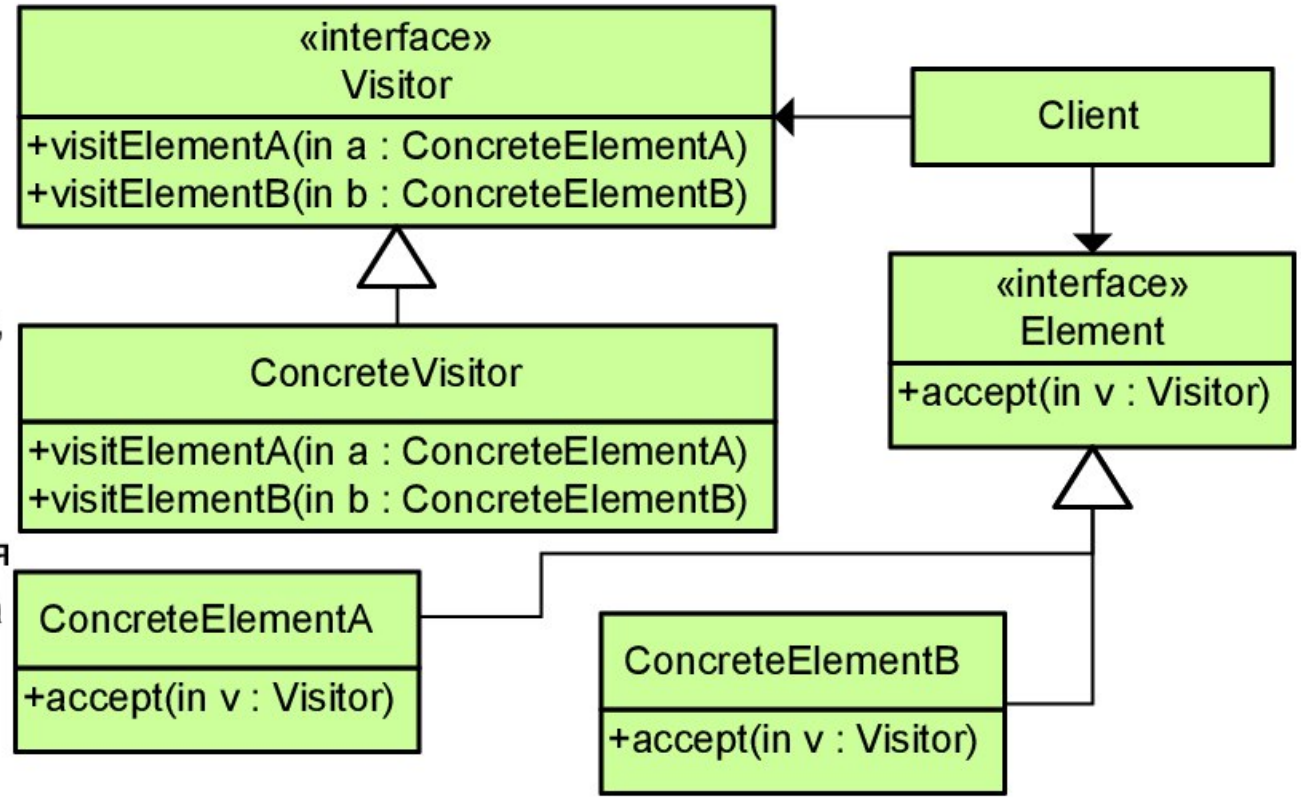
Поведенческие шаблоны

Посетитель
Visitor

Тип: Поведенческий

Что это:

Представляет собой операцию, которая будет выполнена над объектами группы классов. Даёт возможность определить новую операцию без изменения кода классов, над которыми эта операция проводится.



композиция (composition) — подвид агрегации, в которой «части» не могут существовать отдельно от «целого».

агрегация (aggregation) — описывает связь «часть»–«целое», в котором «часть» может существовать отдельно от «целого». Ромб указывается со стороны «целого»

зависимость (dependency) — изменение в одной сущности (независимой) может влиять на состояние или поведение другой сущности (зависимой). Со стороны стрелки указывается независимая сущность

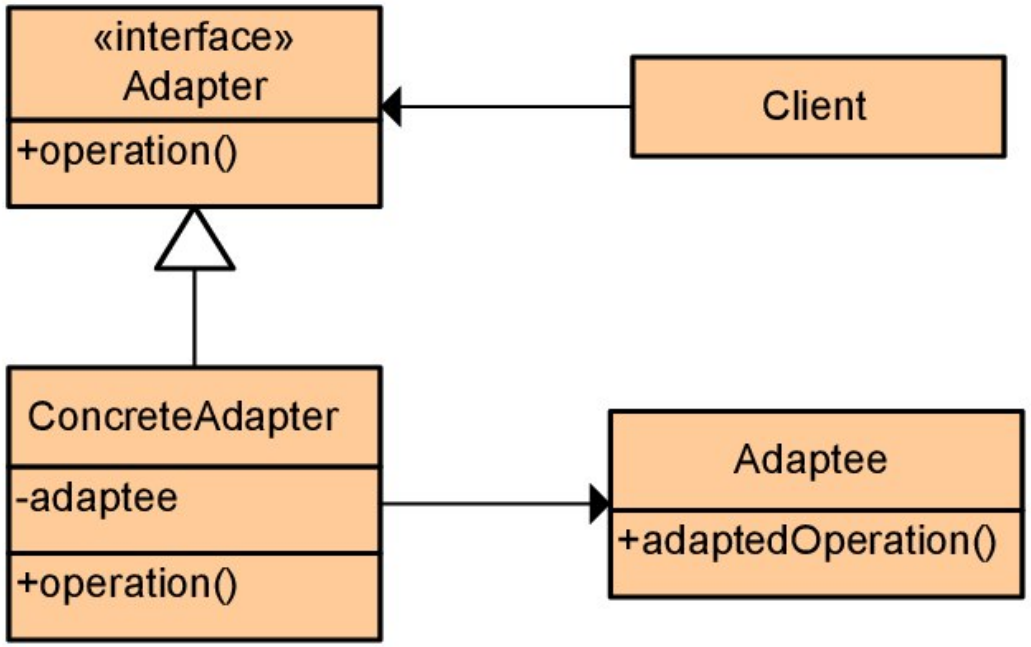
обобщение (generalization) — отношение наследования или реализации интерфейса. Со стороны стрелки находится суперкласс или интерфейс

Источник: https://habr.com/ru/post/210288/

Поведенческие шаблоны

- **Адаптер**
  - Предназначен для организации использования функций объекта, недоступного для модификации, через специально созданный интерфейс
  - Шаблон Адаптер позволяет в процессе проектирования не принимать во внимание возможные различия в интерфейсах уже существующих классов
  - Если есть класс, обладающий требуемыми методами и свойствами (по крайней мере, концептуально), то при необходимости всегда можно воспользоваться шаблоном Адаптер для приведения его интерфейса к нужному виду

Структурные шаблоны

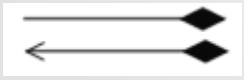


Адаптер
Adapter

Тип: Структурный

Что это:

Конвертирует интерфейс класса в другой интерфейс, ожидаемый клиентом. Позволяет классам с разными интерфейсами работать вместе.



композиция (composition) — подвид агрегации, в которой «части» не могут существовать отдельно от «целого».

агрегация (aggregation) — описывает связь «часть»–«целое», в котором «часть» может существовать отдельно от «целого». Ромб указывается со стороны «целого»

зависимость (dependency) — изменение в одной сущности (независимой) может влиять на состояние или поведение другой сущности (зависимой). Со стороны стрелки указывается независимая сущность

обобщение (generalization) — отношение наследования или реализации интерфейса. Со стороны стрелки находится суперкласс или интерфейс

Источник: https://habr.com/ru/post/210288/

Поведенческие шаблоны

- **Прокси (Заместитель)**
  - Предоставляет объект, который контролирует доступ к другому объекту, перехватывая все вызовы (выполняет функцию контейнера).
  - удалённый заместитель
  - виртуальный заместитель может выполнять оптимизацию
  - защищающий заместитель
  - «умная» ссылка (указатель)
  - Шаблон Proxy может применяться в случаях работы с сетевым соединением, с огромным объектом в памяти (или на диске) или с любым другим ресурсом, который сложно или тяжело копировать

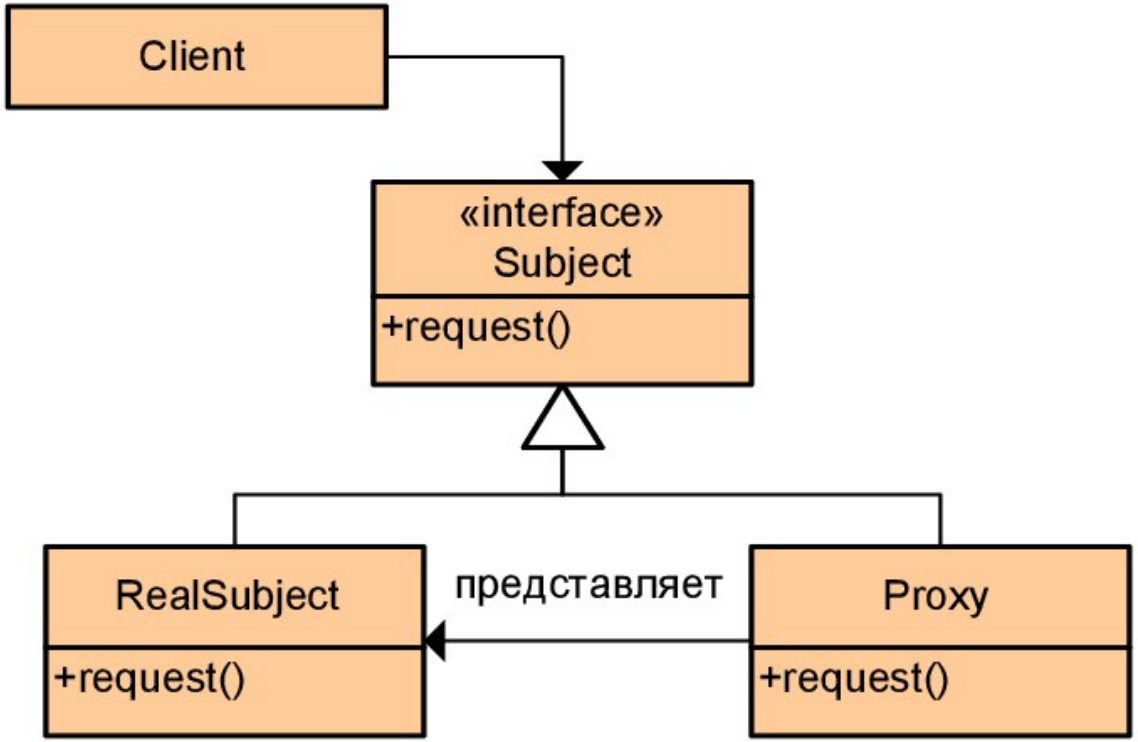
Структурные шаблоны

Прокси
Proxy

Тип: Структурный

Что это:

Предоставляет замену другого объекта для контроля доступа к нему.



композиция (composition) — подвид агрегации, в которой «части» не могут существовать отдельно от «целого».

агрегация (aggregation) — описывает связь «часть»–«целое», в котором «часть» может существовать отдельно от «целого». Ромб указывается со стороны «целого»

зависимость (dependency) — изменение в одной сущности (независимой) может влиять на состояние или поведение другой сущности (зависимой). Со стороны стрелки указывается независимая сущность

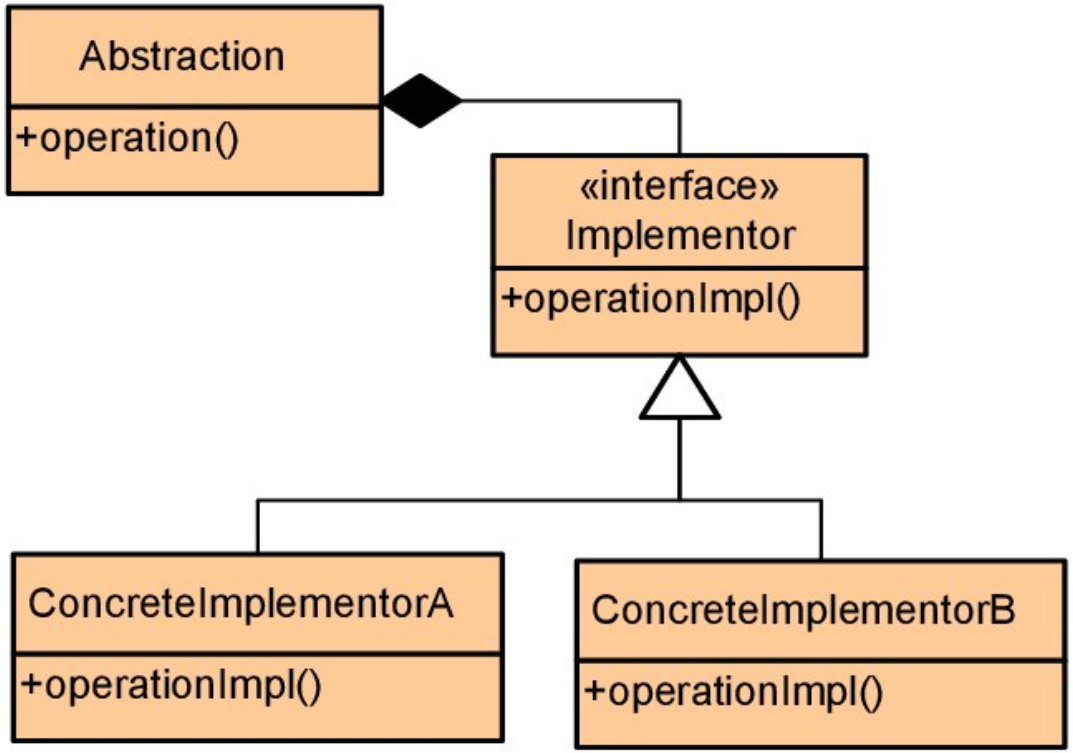
обобщение (generalization) — отношение наследования или реализации интерфейса. Со стороны стрелки находится суперкласс или интерфейс

Источник: https://habr.com/ru/post/210288/

Поведенческие шаблоны

- **Мост**
  - используется в проектировании программного обеспечения чтобы «разделять абстракцию и реализацию так, чтобы они могли изменяться независимо».
  - Проще расширять систему новыми типами за счет сокращения общего числа родственных подклассов.
  - Возможность динамического изменения реализации в процессе выполнения программы.
  - Шаблон полностью скрывает реализацию от клиента. В случае модификации реализации пользовательский код не требует перекомпиляции.
  -

Структурные шаблоны



Мост
Bridge

Тип: Структурный

Что это:

Разделяет абстракцию и реализацию так, чтобы они могли изменяться независимо.

композиция (composition) — подвид агрегации, в которой «части» не могут существовать отдельно от «целого».

агрегация (aggregation) — описывает связь «часть»–«целое», в котором «часть» может существовать отдельно от «целого». Ромб указывается со стороны «целого»

зависимость (dependency) — изменение в одной сущности (независимой) может влиять на состояние или поведение другой сущности (зависимой). Со стороны стрелки указывается независимая сущность

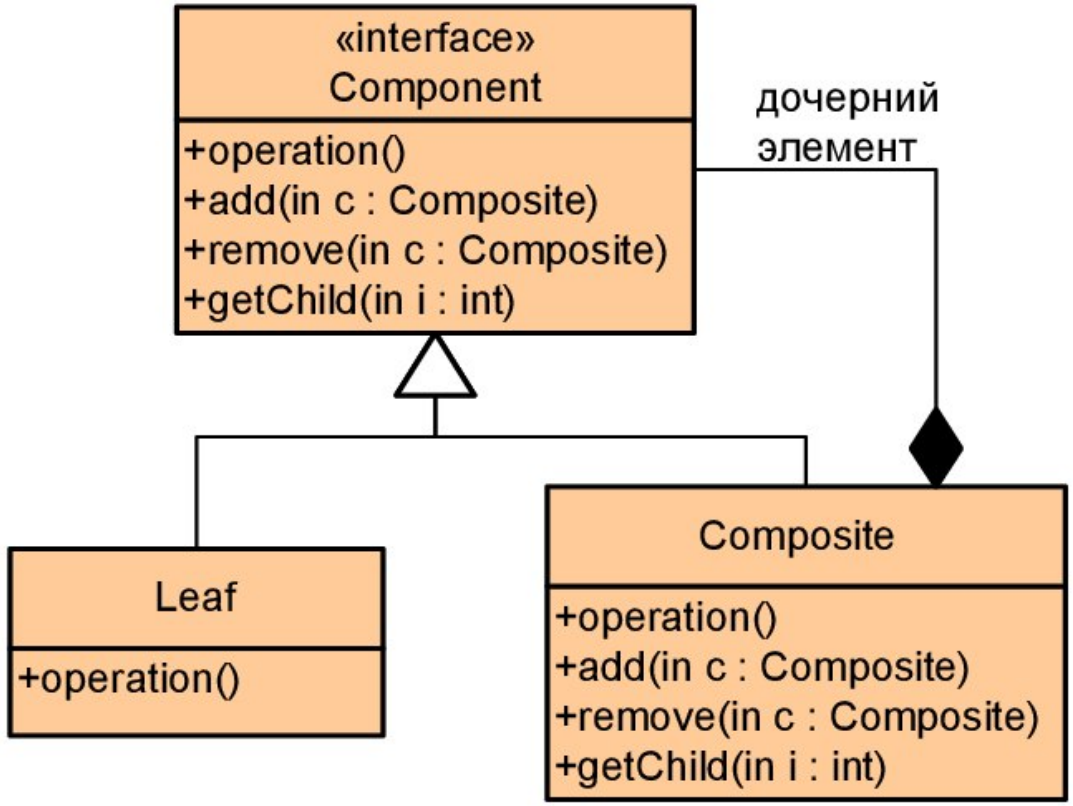
обобщение (generalization) — отношение наследования или реализации интерфейса. Со стороны стрелки находится суперкласс или интерфейс

Источник: https://habr.com/ru/post/210288/

Поведенческие шаблоны

- **Компоновщик**
  - объединяет объекты в древовидную структуру для представления иерархии от частного к целому
  - Необходимо объединять группы схожих объектов и управлять ими.
  - Объекты могут быть как примитивными (элементарными), так и составными (сложными). Составной объект может включать в себя коллекции других объектов, образуя сложные древовидные структуры. Пример: директория файловой системы состоит из элементов, каждый их которых также может быть директорией.
  - Код клиента работает с примитивными и составными объектами единообразно.

Структурные шаблоны



Компоновщик
Composite

Тип: Структурный

Что это:

Компонует объекты в древовидную структуру, представляя их в виде иерархии. Позволяет клиенту одинаково обращаться как к отдельному объекту, так и к целому поддереву.

композиция (composition) — подвид агрегации, в которой «части» не могут существовать отдельно от «целого».

агрегация (aggregation) — описывает связь «часть»–«целое», в котором «часть» может существовать отдельно от «целого». Ромб указывается со стороны «целого»

зависимость (dependency) — изменение в одной сущности (независимой) может влиять на состояние или поведение другой сущности (зависимой). Со стороны стрелки указывается независимая сущность

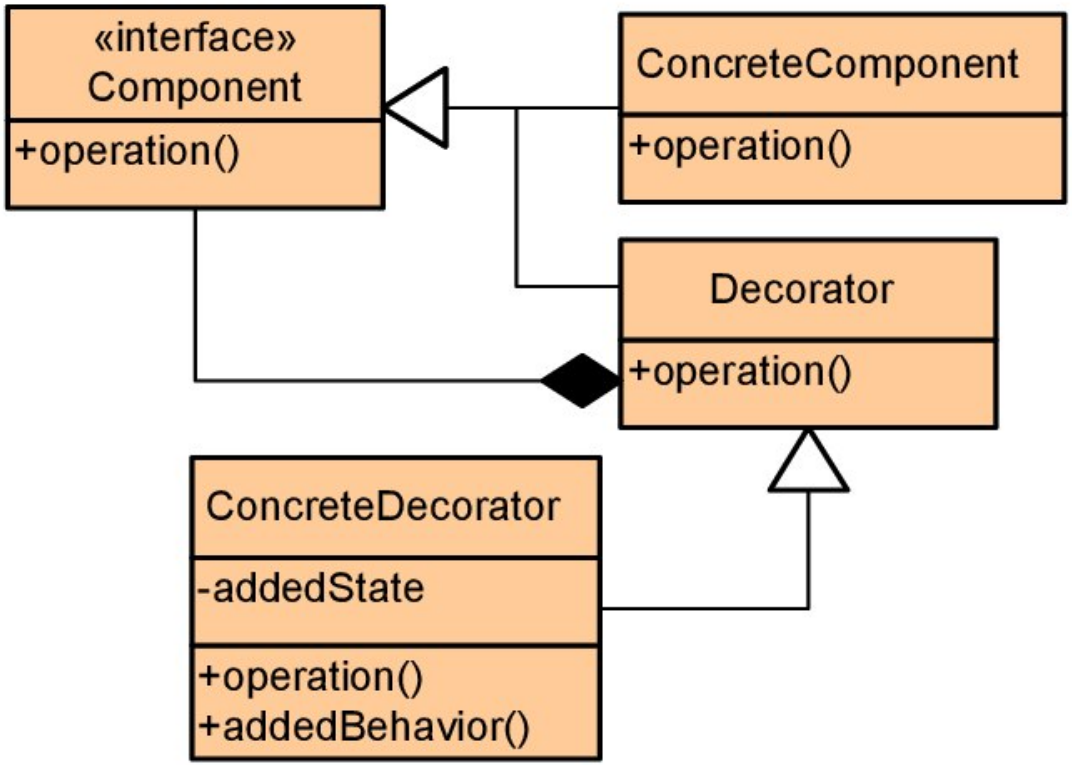
обобщение (generalization) — отношение наследования или реализации интерфейса. Со стороны стрелки находится суперкласс или интерфейс

Источник: https://habr.com/ru/post/210288/

Поведенческие шаблоны

- **Декоратор**
  - Класс, расширяющий функциональность другого класса без использования наследования
  - Шаблон динамически добавляет новые обязанности объекту. Декораторы являются гибкой альтернативой порождению подклассов для расширения функциональности
  - Рекурсивно декорирует основной объект
  - Шаблон использует схему "обертываем подарок, кладем его в коробку, обертываем коробку"

Структурные шаблоны

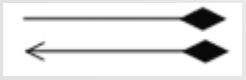


Декоратор
Decorator

Тип: Структурный

Что это:

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



композиция (composition) — подвид агрегации, в которой «части» не могут существовать отдельно от «целого».

агрегация (aggregation) — описывает связь «часть»–«целое», в котором «часть» может существовать отдельно от «целого». Ромб указывается со стороны «целого»

зависимость (dependency) — изменение в одной сущности (независимой) может влиять на состояние или поведение другой сущности (зависимой). Со стороны стрелки указывается независимая сущность

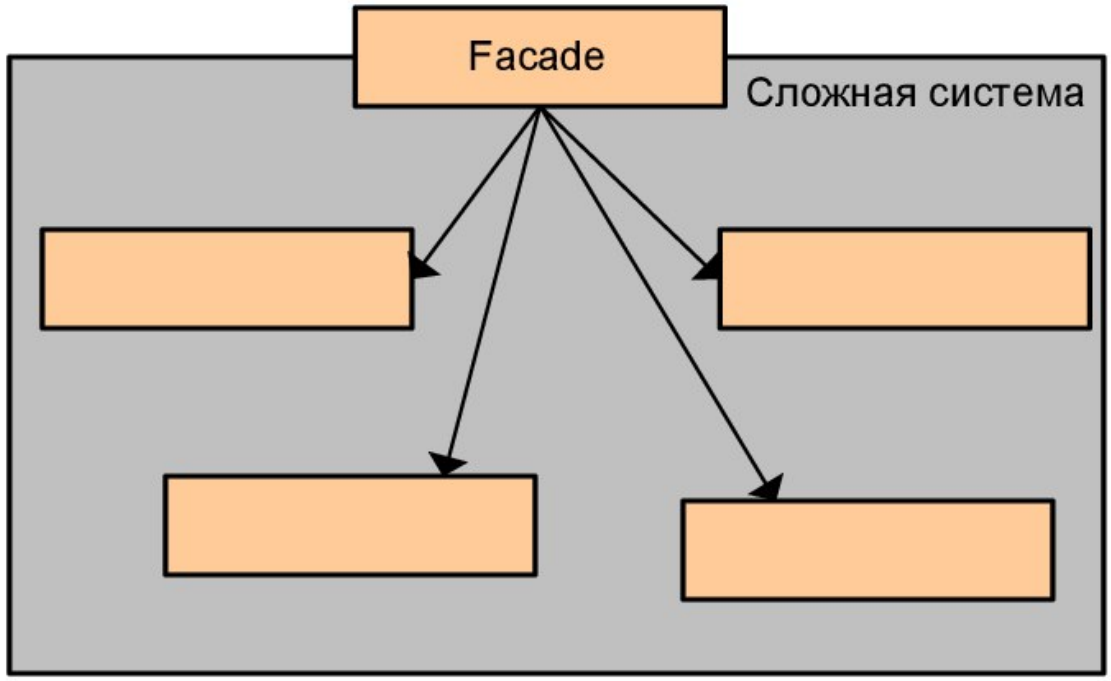
обобщение (generalization) — отношение наследования или реализации интерфейса. Со стороны стрелки находится суперкласс или интерфейс

Источник: https://habr.com/ru/post/210288/

Поведенческие шаблоны

- **Фасад**
  - Объект, который абстрагирует работу с несколькими классами, объединяя их в единое целое
  - Шаблон предоставляет унифицированный интерфейс вместо набора интерфейсов некоторой подсистемы. Фасад определяет интерфейс более высокого уровня, упрощающий использование подсистемы
  - Шаблон "обертывает" сложную подсистему более простым интерфейсом

Структурные шаблоны

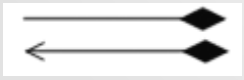


Фасад
Facade

Тип: Структурный

Что это:

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



композиция (composition) — подвид агрегации, в которой «части» не могут существовать отдельно от «целого».

агрегация (aggregation) — описывает связь «часть»–«целое», в котором «часть» может существовать отдельно от «целого». Ромб указывается со стороны «целого»

зависимость (dependency) — изменение в одной сущности (независимой) может влиять на состояние или поведение другой сущности (зависимой). Со стороны стрелки указывается независимая сущность

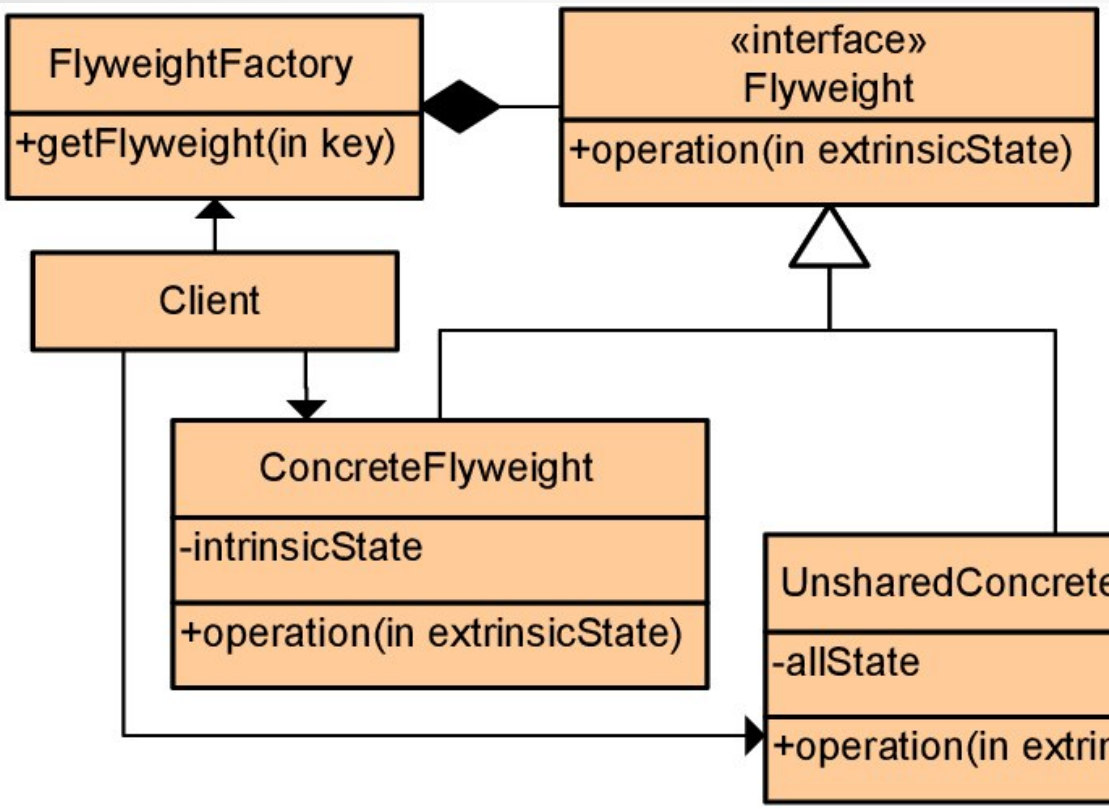
обобщение (generalization) — отношение наследования или реализации интерфейса. Со стороны стрелки находится суперкласс или интерфейс

Источник: https://habr.com/ru/post/210288/

Поведенческие шаблоны

- **Приспособленец**
  - Это объект, представляющий себя как уникальный экземпляр в разных местах программы, но фактически не являющийся таковым
  - Шаблон использует разделение для эффективной поддержки большого числа мелких объектов
  - Оптимизация работы с памятью путём предотвращения создания экземпляров элементов, имеющих общую сущность

Структурные шаблоны



Приспособленец

Flyweight

Тип: Структурный

Что это:

Благодаря совместному использованию, поддерживает эффективную работу с большим количеством объектов.

композиция (composition) — подвид агрегации, в которой «части» не могут существовать отдельно от «целого».

агрегация (aggregation) — описывает связь «часть»–«целое», в котором «часть» может существовать отдельно от «целого». Ромб указывается со стороны «целого»

зависимость (dependency) — изменение в одной сущности (независимой) может влиять на состояние или поведение другой сущности (зависимой). Со стороны стрелки указывается независимая сущность

обобщение (generalization) — отношение наследования или реализации интерфейса. Со стороны стрелки находится суперкласс или интерфейс

Источник: https://habr.com/ru/post/210288/

Поведенческие шаблоны

- **Абстрактная фабрика**
  - Класс, который представляет собой интерфейс для создания компонентов системы
  - Система должна оставаться независимой как от процесса создания новых объектов, так и от типов порождаемых объектов. Непосредственное использование выражения new в коде приложения нежелательно
  - Необходимо создавать группы или семейства взаимосвязанных объектов, исключая возможность одновременного использования объектов из разных семейств в одном контексте

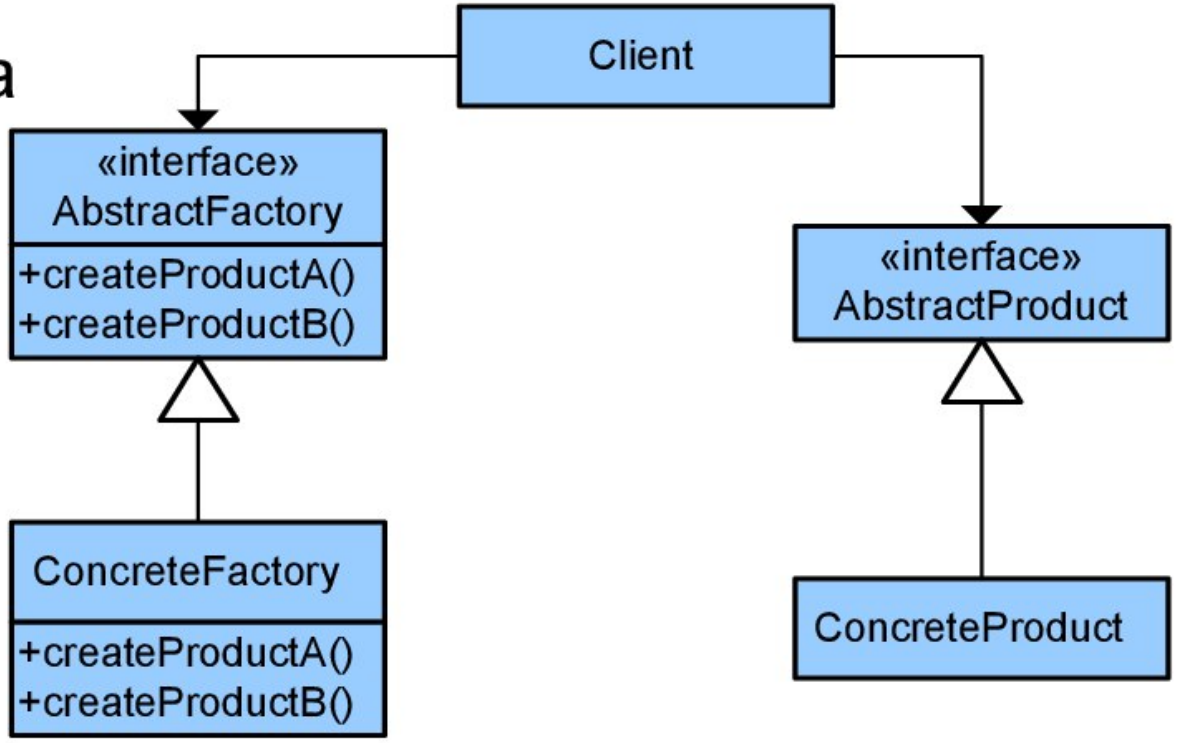
Порождающие шаблоны

Абстрактная фабрика
Abstract factory

Тип: Порождающий

Что это:

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



композиция (composition) — подвид агрегации, в которой «части» не могут существовать отдельно от «целого».

агрегация (aggregation) — описывает связь «часть»–«целое», в котором «часть» может существовать отдельно от «целого». Ромб указывается со стороны «целого»

зависимость (dependency) — изменение в одной сущности (независимой) может влиять на состояние или поведение другой сущности (зависимой). Со стороны стрелки указывается независимая сущность

обобщение (generalization) — отношение наследования или реализации интерфейса. Со стороны стрелки находится суперкласс или интерфейс

Источник: https://habr.com/ru/post/210288/

Поведенческие шаблоны

- **Строитель**
  - Класс, который представляет собой интерфейс для создания сложного объекта
  - В системе могут существовать сложные объекты, создание которых за одну операцию затруднительно или невозможно. Требуется поэтапное построение объектов с контролем результатов выполнения каждого этапа.
  - Данные должны иметь несколько представлений. Например, документ в формате RTF (Rich Text Format) можно преобразовать в другие форматы (например, Microsoft Word или простой ASCII-текст), то полученные документы и будут представлениями исходных данных

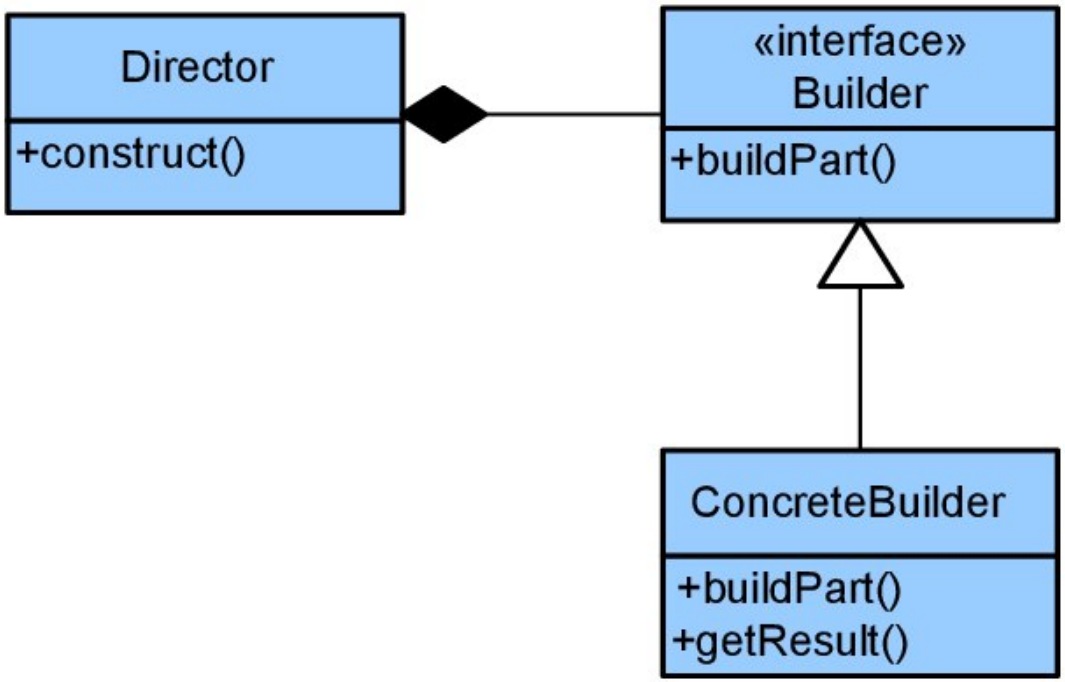
Порождающие шаблоны

Строитель
Builder

Тип: Порождающий

Что это:

Разделяет создание сложного объекта и инициализацию его состояния так, что одинаковый процесс построения может создать объекты с разным состоянием.



композиция (composition) — подвид агрегации, в которой «части» не могут существовать отдельно от «целого».

агрегация (aggregation) — описывает связь «часть»–«целое», в котором «часть» может существовать отдельно от «целого». Ромб указывается со стороны «целого»

зависимость (dependency) — изменение в одной сущности (независимой) может влиять на состояние или поведение другой сущности (зависимой). Со стороны стрелки указывается независимая сущность

обобщение (generalization) — отношение наследования или реализации интерфейса. Со стороны стрелки находится суперкласс или интерфейс

Источник: https://habr.com/ru/post/210288/

Поведенческие шаблоны

- **Фабричный метод**
  - Определяет интерфейс для создания объекта, но оставляет подклассам решение о том, какой класс инстанцировать
  - Система должна оставаться расширяемой путем добавления объектов новых типов
  - Позволяет системе оставаться независимой как от самого процесса порождения объектов, так и от их типов
  - Заранее известно, когда нужно создавать объект, но неизвестен его тип

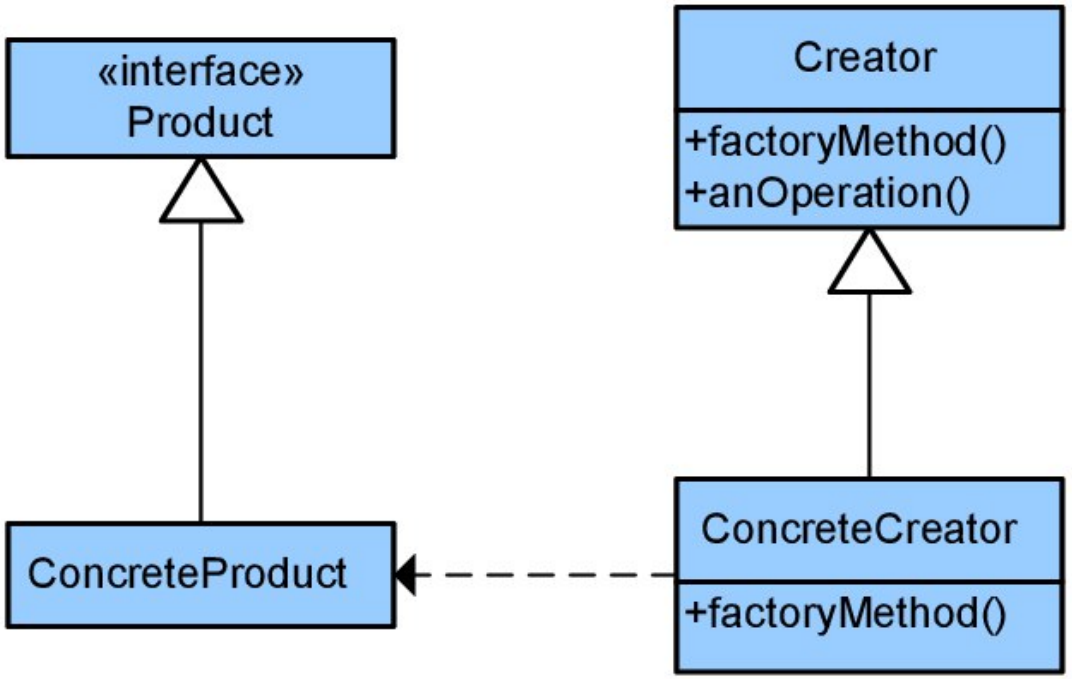
Порождающие шаблоны

Фабричный метод
Factory method

Тип: Порождающий

Что это:

Определяет интерфейс для создания объекта, но позволяет подклассам решать, какой класс инстанцировать. Позволяет делегировать создание объекта подклассам.



композиция (composition) — подвид агрегации, в которой «части» не могут существовать отдельно от «целого».

агрегация (aggregation) — описывает связь «часть»–«целое», в котором «часть» может существовать отдельно от «целого». Ромб указывается со стороны «целого»

зависимость (dependency) — изменение в одной сущности (независимой) может влиять на состояние или поведение другой сущности (зависимой). Со стороны стрелки указывается независимая сущность

обобщение (generalization) — отношение наследования или реализации интерфейса. Со стороны стрелки находится суперкласс или интерфейс

Источник: https://habr.com/ru/post/210288/

Поведенческие шаблоны

- **Прототип**
  - Определяет интерфейс создания объекта через клонирование другого объекта вместо создания через конструктор
  - Система должна оставаться независимой как от процесса создания новых объектов, так и от типов порождаемых объектов
  - Необходимо создавать объекты, точные классы которых становятся известными уже на стадии выполнения программы

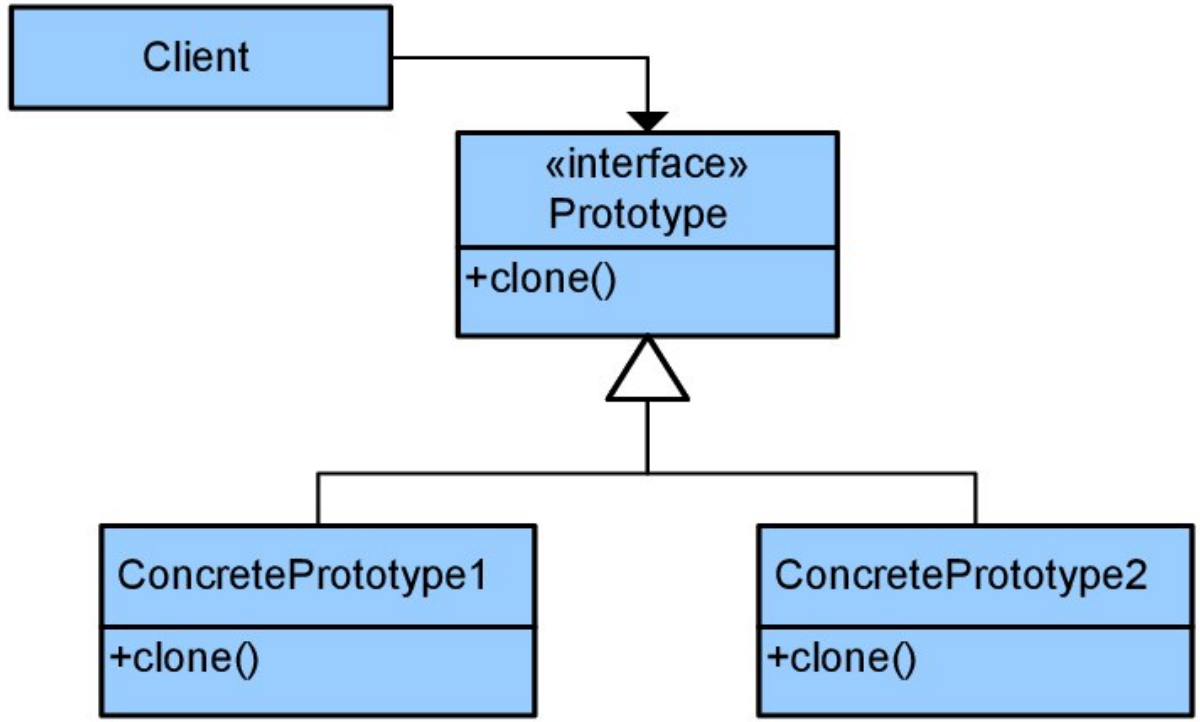
Порождающие шаблоны

Прототип
Prototype

Тип: Порождающий

Что это:

Определяет несколько видов объектов, чтобы при создании использовать объект-прототип и создаёт новые объекты, копируя прототип.



композиция (composition) — подвид агрегации, в которой «части» не могут существовать отдельно от «целого».

агрегация (aggregation) — описывает связь «часть»–«целое», в котором «часть» может существовать отдельно от «целого». Ромб указывается со стороны «целого»

зависимость (dependency) — изменение в одной сущности (независимой) может влиять на состояние или поведение другой сущности (зависимой). Со стороны стрелки указывается независимая сущность

обобщение (generalization) — отношение наследования или реализации интерфейса. Со стороны стрелки находится суперкласс или интерфейс

Источник: https://habr.com/ru/post/210288/

Поведенческие шаблоны

- **Одиночка**
  - Класс, который может иметь только один экземпляр
  - Необходимость наличия сущности только в единственном экземпляре
  - Необходимо уметь создавать единственный экземпляр некоторого типа, предоставлять к нему доступ извне и запрещать создание нескольких экземпляров того же типа

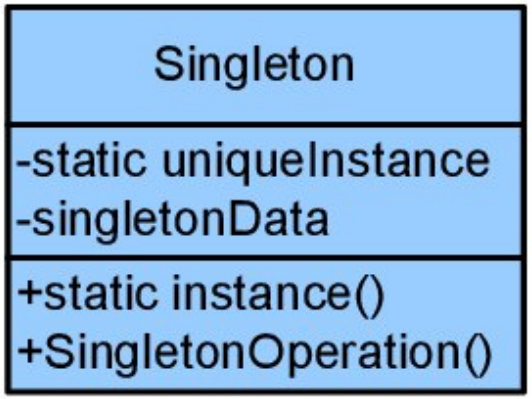
Порождающие шаблоны

Одиночка
Singleton

Тип: Порождающий

Что это:

Гарантирует, что класс имеет только один экземпляр и предоставляет глобальную точку доступа к нему.



композиция (composition) — подвид агрегации, в которой «части» не могут существовать отдельно от «целого».

агрегация (aggregation) — описывает связь «часть»–«целое», в котором «часть» может существовать отдельно от «целого». Ромб указывается со стороны «целого»

зависимость (dependency) — изменение в одной сущности (независимой) может влиять на состояние или поведение другой сущности (зависимой). Со стороны стрелки указывается независимая сущность

обобщение (generalization) — отношение наследования или реализации интерфейса. Со стороны стрелки находится суперкласс или интерфейс

Источник: https://habr.com/ru/post/210288/

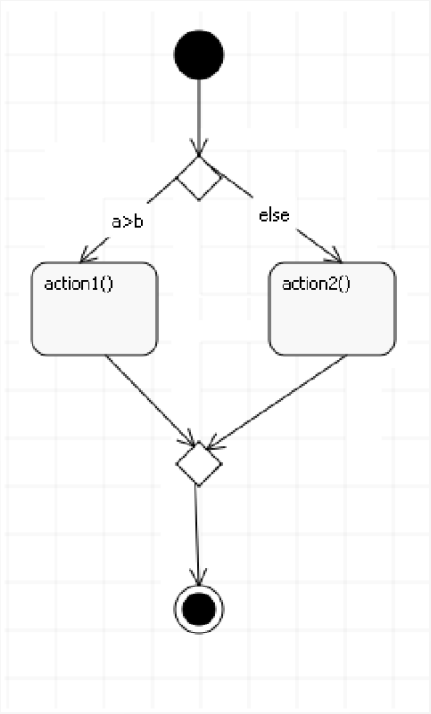
Анти-паттерны

- Шаблоны ошибок, которые совершаются при решении различных задач

- Программирование копи-пастом (Copy and Paste Programming) — создание дубликата функции
- «Брось, можно писать не только одну функцию!» или Спагетти-код (Spaghetti code)
- Золотой молоток (Golden hammer) — любое решение универсально!
- «Что за 42?» или Магические числа (Magic numbers)
- «Что значит d:\proj\tests.dat?» или Жёсткое кодирование (Hard code)
- Мягкое кодирование (Soft code) — противоположность жесткому кодированию
- «Как это вы передали строку вместо числа?!» или Слепая вера (Blind faith)
- Божественный объект (God Object)
- ...

Вишенка на торте

- **Спецификация** — формальное описание объекта проектирования
  - Использование формальных языков (UML)
  - Использование шаблонов
- **Кодогенерация** — автоматизированное создание кода на основе спецификации
  - Специальные инструменты
  - Общий случай: создание заготовок под код
  - Пример:
    - Создание БД по описаниям структур
    - Генерация заголовочных файлов и заготовок файлов с кодом на основе диаграммы классов




```
if (a>b){
  action1();
}
else{
  action2();
}
```