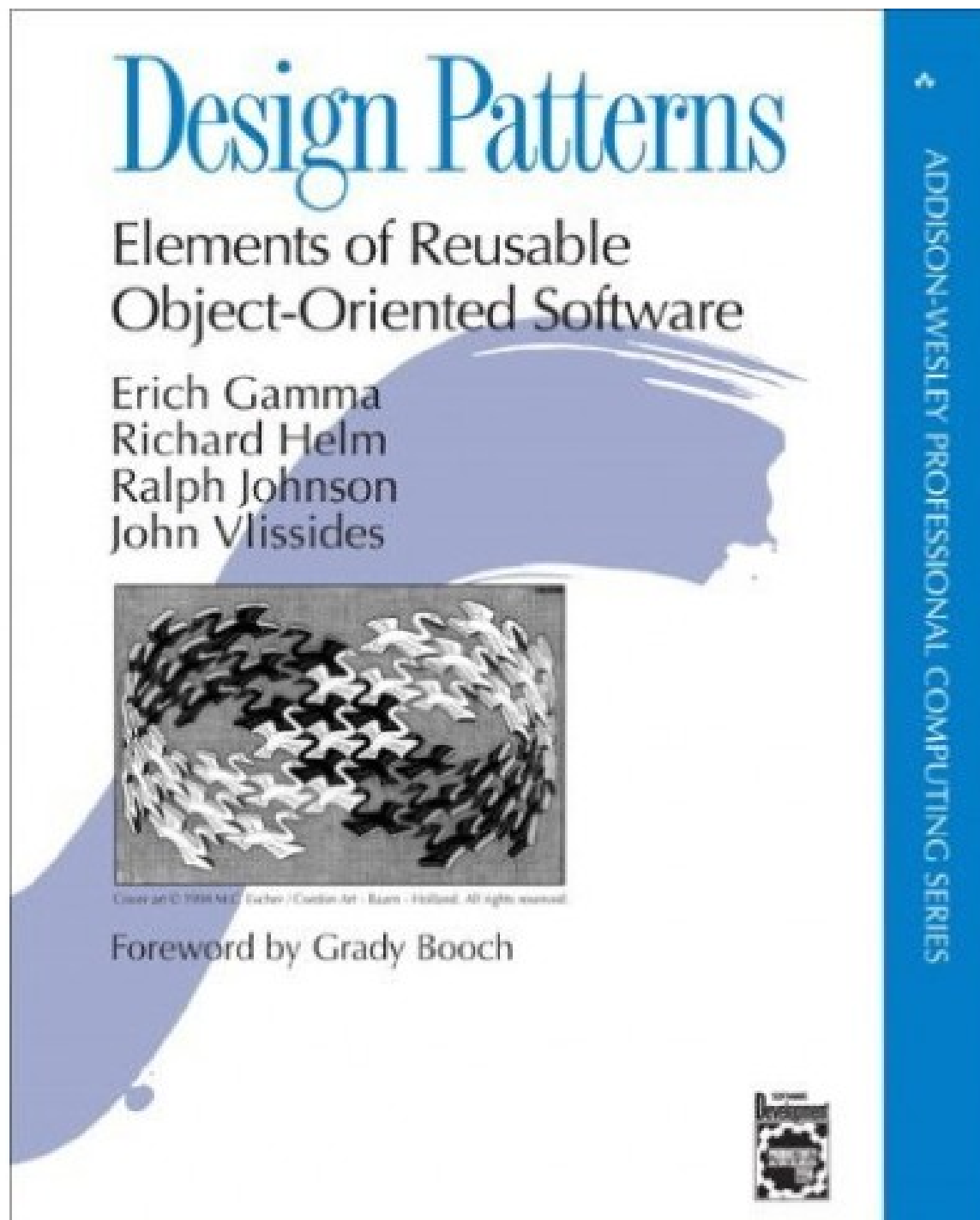


РАЗРАБОТКА ВЕБ-САЙТОВ\*, ПРОЕКТИРОВАНИЕ И РЕФАКТОРИНГ\*, АНАЛИЗ И  
ПРОЕКТИРОВАНИЕ СИСТЕМ\*

## Шпаргалка по шаблонам проектирования

WarAngel\_alk 25 января 2014 в 21:07 👁 663k







Перевод pdf файла с сайта <http://www.mcdonaldland.info/> с

описанием 23-х шаблонов проектирования GOF. Каждый пункт содержит [очень] короткое описание паттерна и UML-диаграмму. Сама шпаргалка доступна в pdf, в виде двух png файлов (как в оригинале), и в виде 23-х отдельных частей изображений. Для самых нетерпеливых — все файлы в [конце статьи](#).

**Под катом — много картинок.**

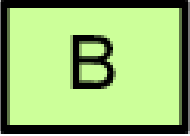
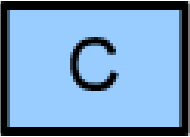

## Условные обозначения

### Отношения между классами

-  — агрегация (aggregation) — описывает связь «часть»–«целое», в котором «часть» может существовать отдельно от «целого». Ромб указывается со стороны «целого».
-  — композиция (composition) — подвид агрегации, в которой «части» не могут существовать отдельно от «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.

## Виды паттернов

-  — поведенческие (behavioral);
-  — порождающие (creational);
-  — структурные (structural).

## Список шаблонов

 Абстрактная фабрика	 Фасад	 Прокси
 Адаптер	 Фабричный метод	 Наблюдатель
 Мост	 Приспособленец	 Одиночка
 Строитель	 Интерпретатор	 Состояние
 Цепочка обязанностей	 Итератор	 Стратегия
 Команда	 Посредник	 Шаблонный метод
 Компоновщик	 Хранитель	 Посетитель
 Декоратор	 Прототип	

## Хранитель (memento)

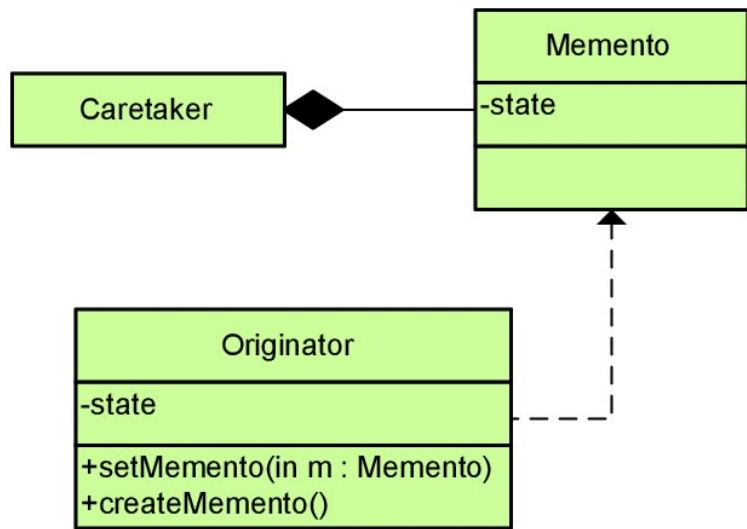
## Хранитель

*Memento*

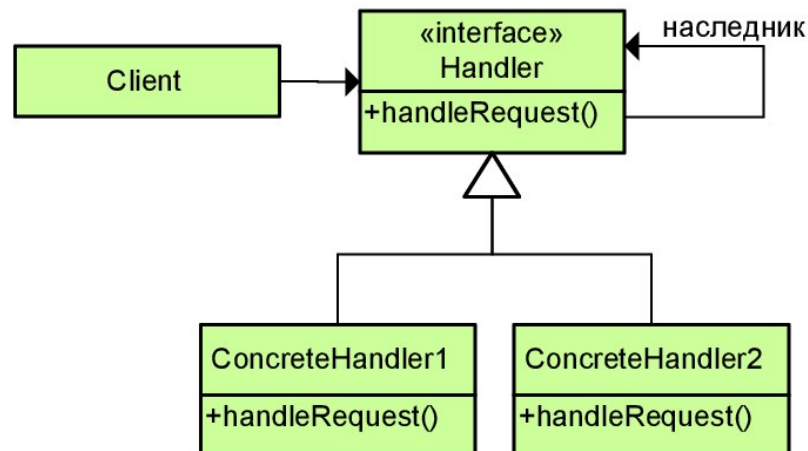
Тип: Поведенческий

Что это:

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



## Цепочка обязанностей (chain of responsibility)



## 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.

## Наблюдатель (observer)

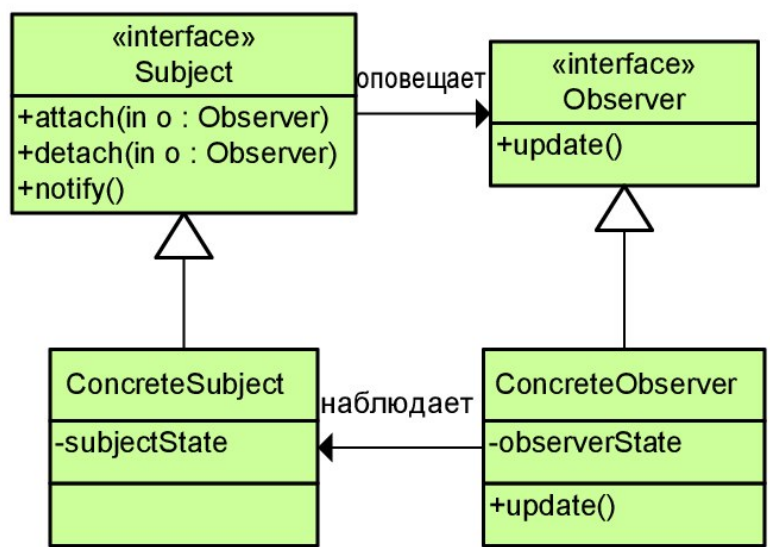
# Наблюдатель

*Observer*

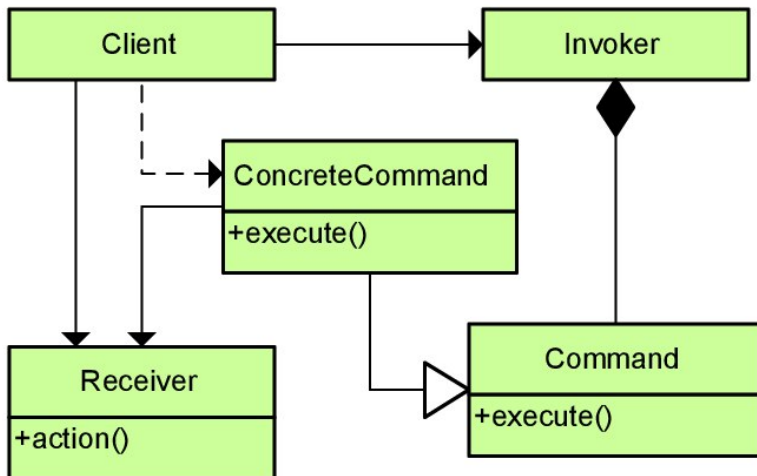
**Тип:** Поведенческий

**Что это:**

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



## Команда (command)



## 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.

## Состояние (state)

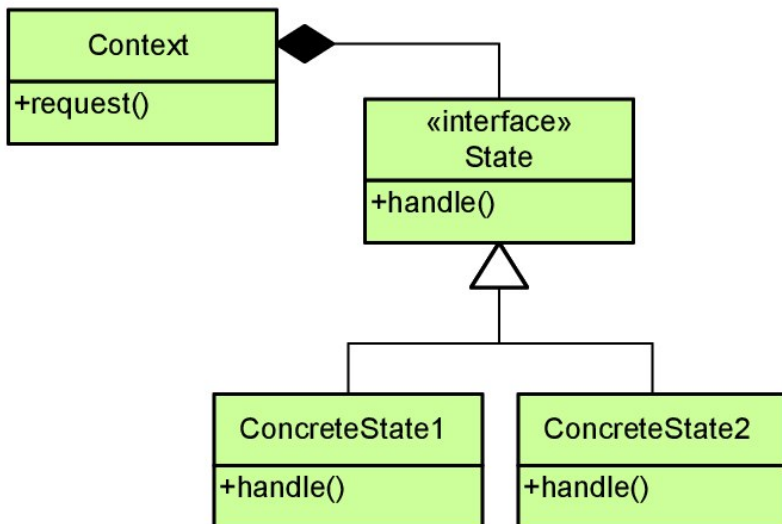
## Состояние

*State*

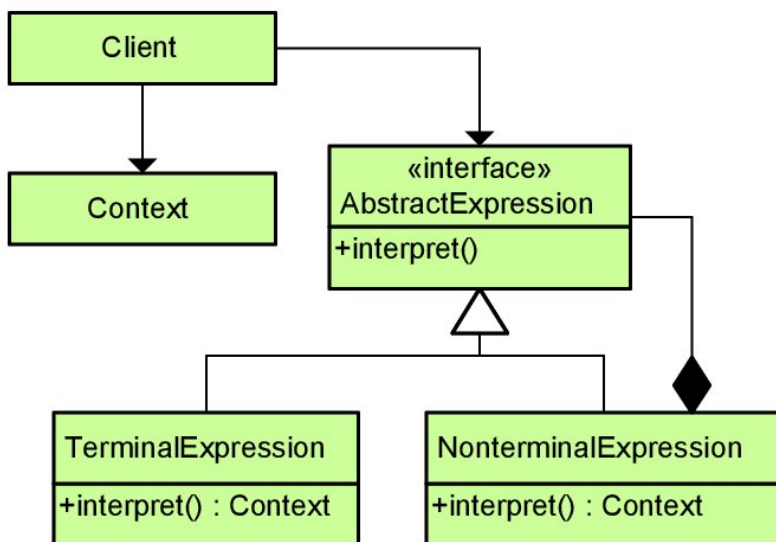
**Тип:** Поведенческий

**Что это:**

Позволяет объекту изменять своё поведение в зависимости от внутреннего состояния.



## Интерпретатор (interpreter)



## Интерпретатор

*Interpreter*

**Тип:** Поведенческий

**Что это:**

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

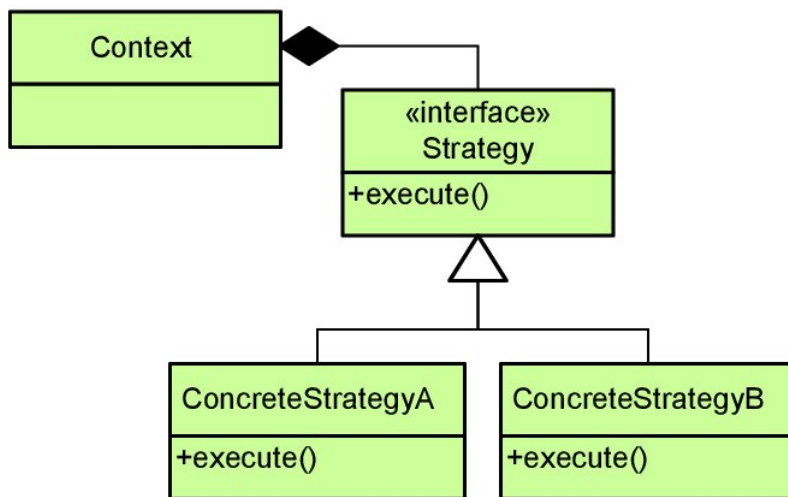
## Стратегия (strategy)

# Стратегия *Strategy*

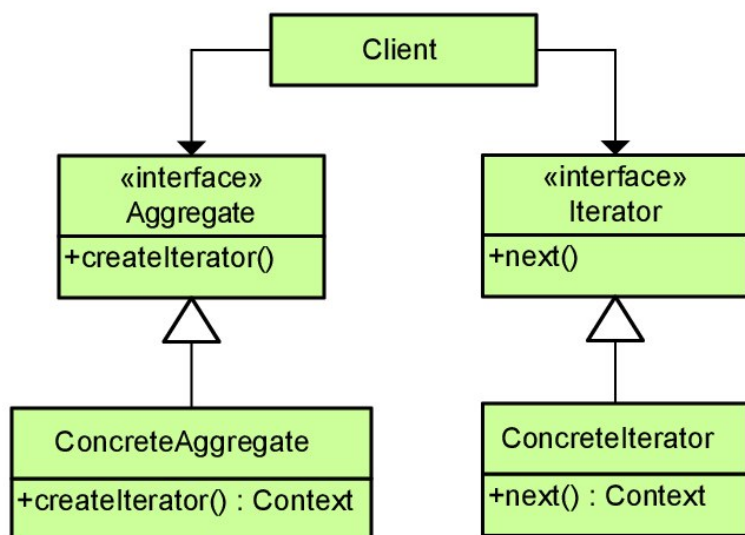
**Тип:** Поведенческий

**Что это:**

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



## Итератор (iterator)



## Итератор *Iterator*

**Тип:** Поведенческий

**Что это:**

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

## Шаблонный метод (template method)

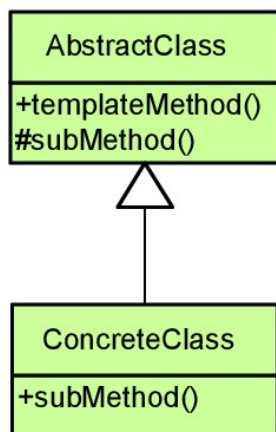
# Шаблонный метод

*Template method*

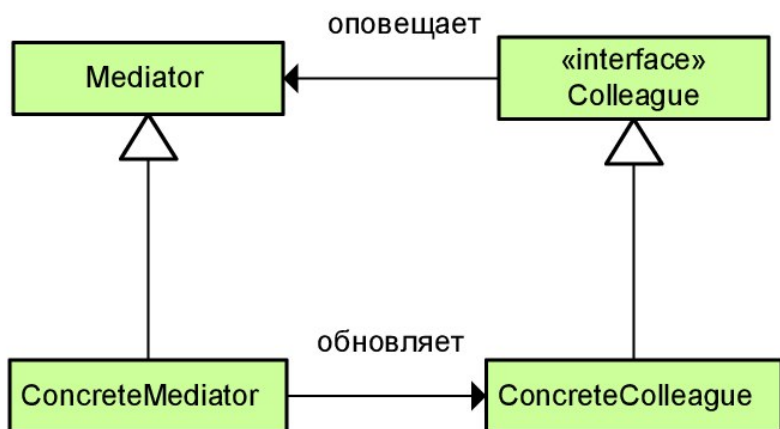
**Тип:** Поведенческий

**Что это:**

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



## Посредник (mediator)



## Посредник *Mediator*

**Тип:** Поведенческий

**Что это:**

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

## Посетитель (visitor)

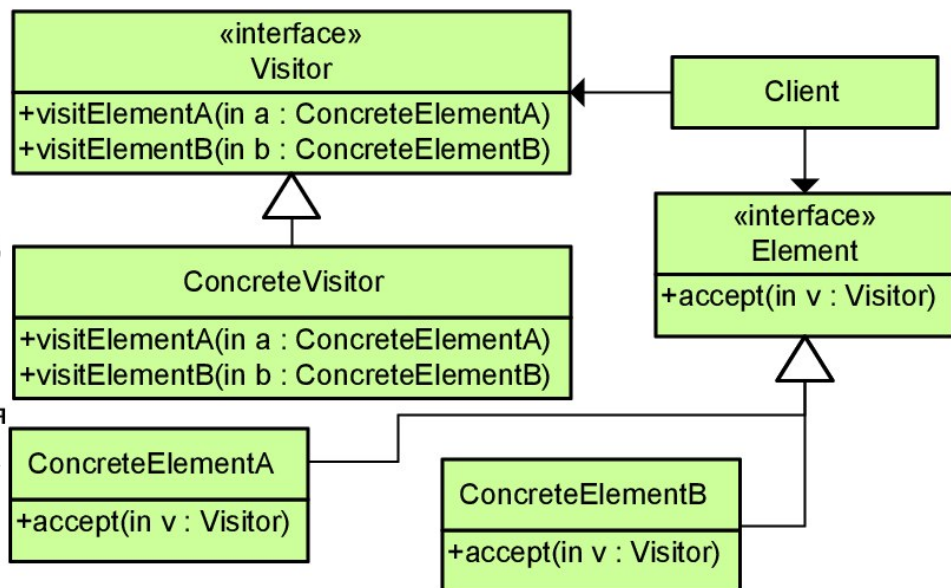


# Посетитель *Visitor*

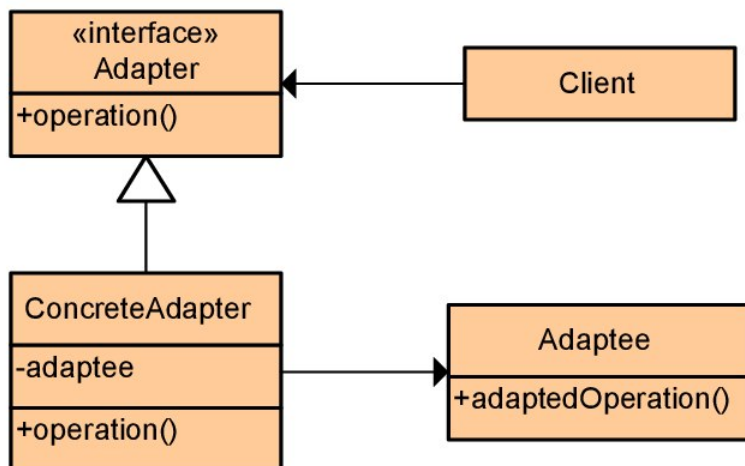
**Тип:** Поведенческий

**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.



## Адаптер (adapter)



## Адаптер *Adapter*

**Тип:** Структурный

**Что это:**

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

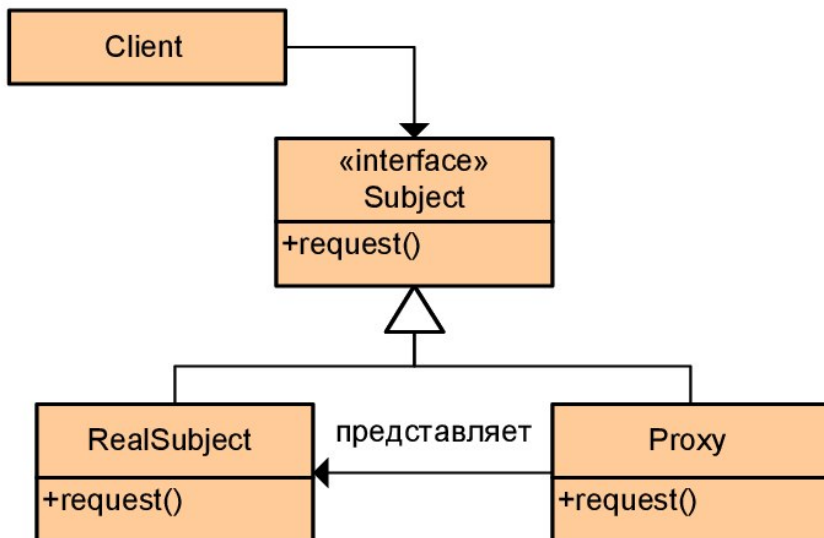
## Прокси (proxy)

# Прокси *Proxy*

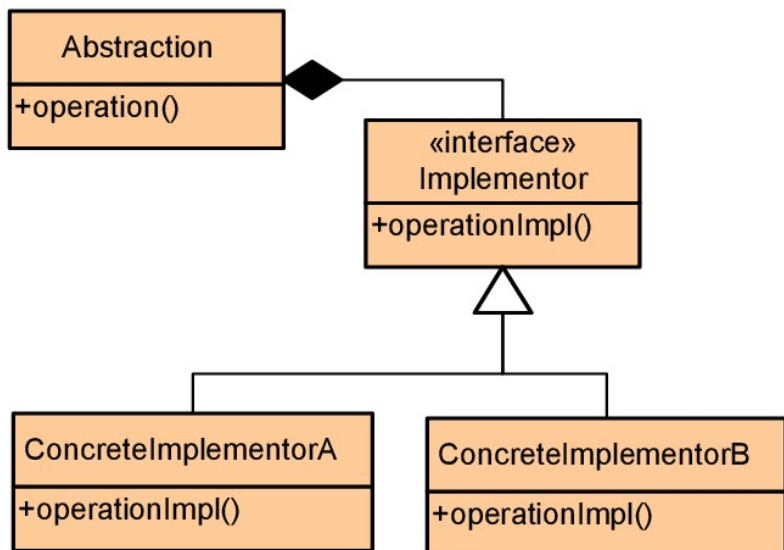
Тип: Структурный

Что это:

Предоставляет замену другого объекта для контроля доступа к нему.



## Мост (bridge)



## Мост *Bridge*

Тип: Структурный

Что это:

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

## Абстрактная фабрика (abstract factory)

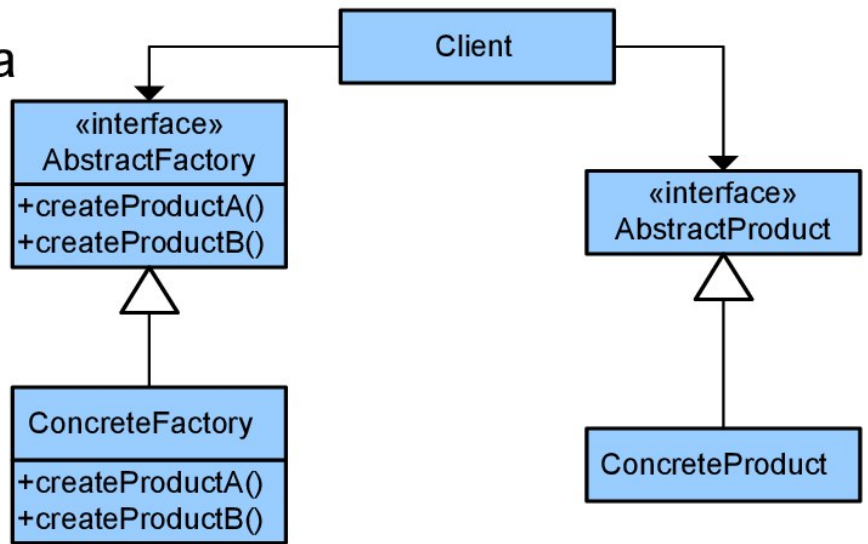
# Абстрактная фабрика

*Abstract factory*

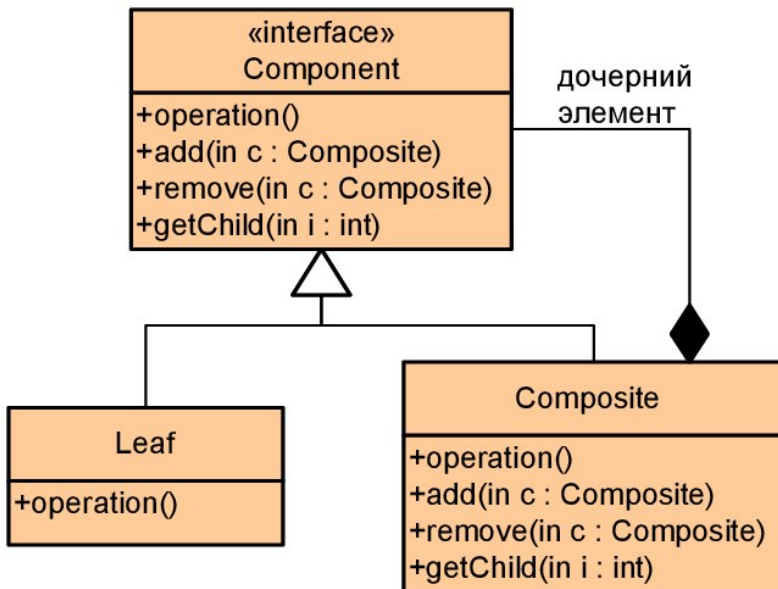
**Тип:** Порождающий

**Что это:**

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



## Компоновщик (composite)



## Компоновщик

*Composite*

**Тип:** Структурный

**Что это:**

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

## Строитель (builder)

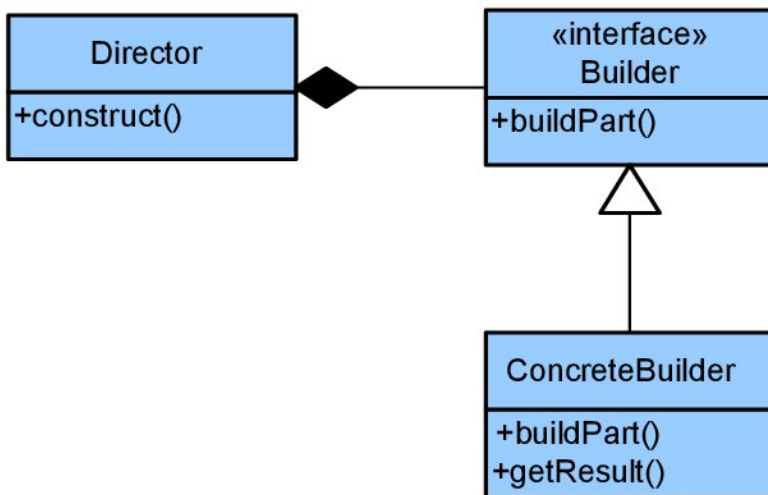
# Строитель

*Builder*

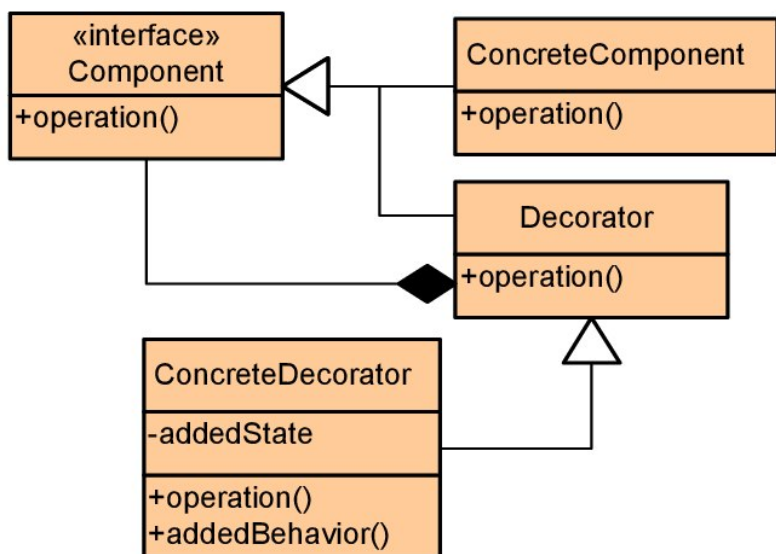
**Тип:** Порождающий

**Что это:**

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



## Декоратор (decorator)



## Декоратор

*Decorator*

**Тип:** Структурный

**Что это:**

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

## Фабричный метод (factory method)

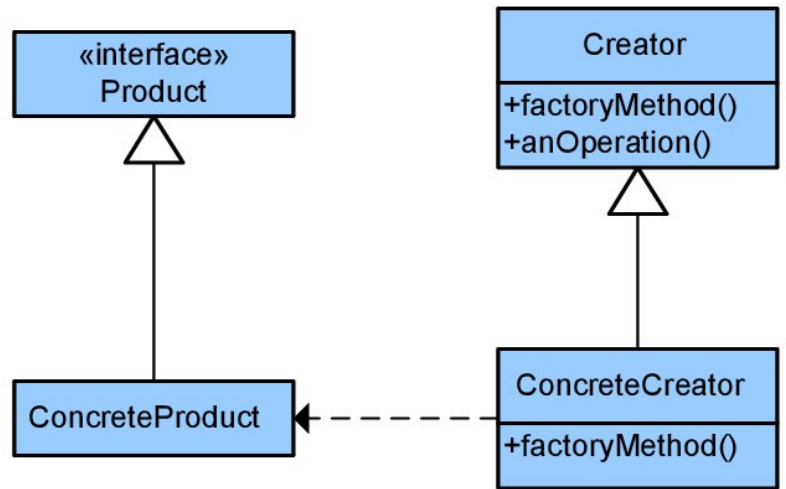
# Фабричный метод

*Factory method*

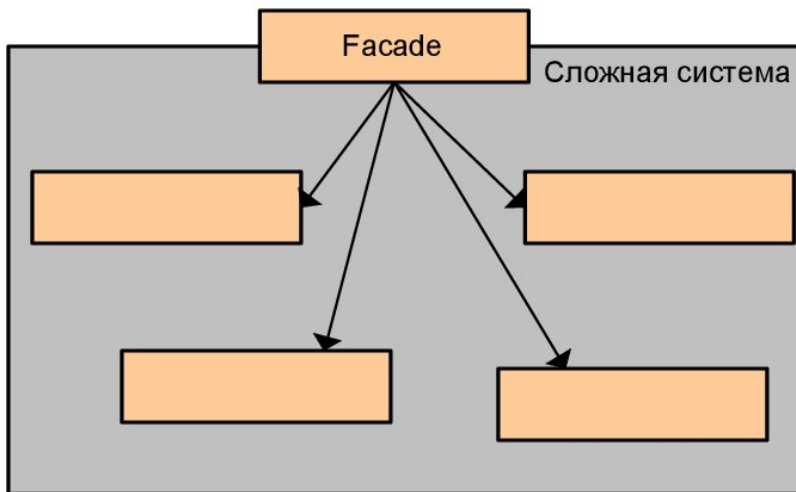
**Тип:** Порождающий

**Что это:**

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



# Фасад (facade)



## Фасад

*Facade*

**Тип:** Структурный

**Что это:**

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

# Прототип (prototype)

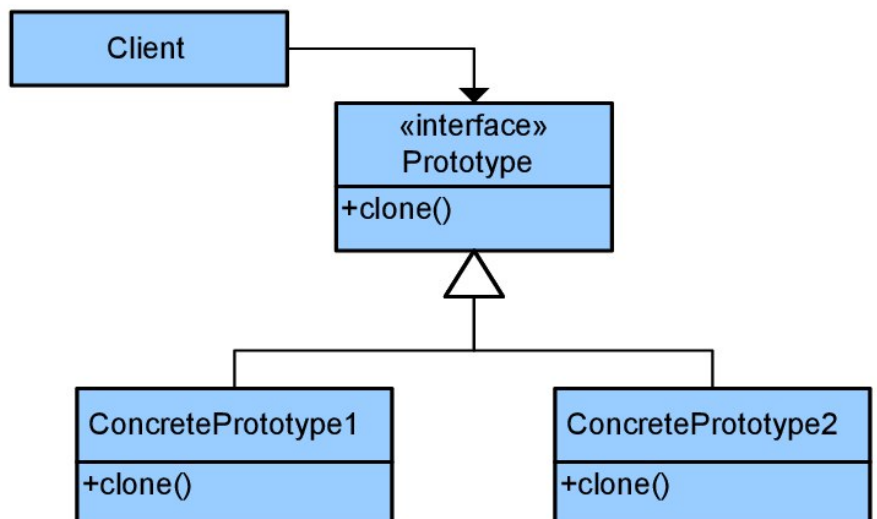
## Прототип

*Prototype*

**Тип:** Порождающий

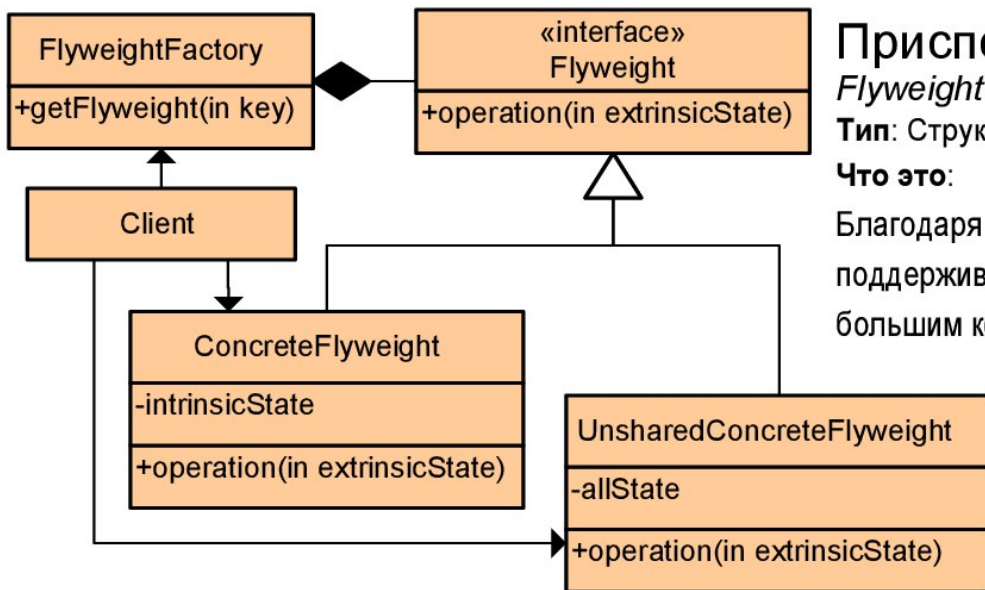
**Что это:**

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





## Приспособленец (flyweight)



## Приспособленец

*Flyweight*

Тип: Структурный

Что это:

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

## Одиночка (singleton)

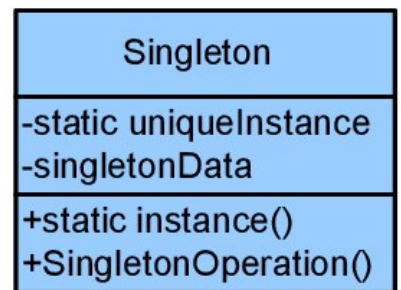
### Одиночка

*Singleton*

Тип: Порождающий

Что это:

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



## Файлы

- все паттерны в [pdf-файле](#).
- то же самое, но в png — [1](#) и [2](#) части.
- [архив](#) с нарезанными изображениями.

Upd. оригинальный [pdf](#) и изображения ([1](#), [2](#)).

P.S. По запросу «шаблоны проектирования» 636 топиков, а хаба

нет; а по «bitcoin» — 278 топиков и хаб есть. Прошу восстановить справедливость!

## Опросы

Проголосовать:



+166



Поделиться:



Сохранить:



Комментарии (64)

## Похожие публикации

### Шаблоны проектирования в адвенчурах: часть первая

ПЕРЕВОД

Silf • 1 сентября 2013 в 14:40

32

### Использование паттернов проектирования в JavaScript: Порождающие паттерны

ИЗ ПЕСОЧНИЦЫ

yarkeev • 4 апреля 2013 в 16:15

30

Nagg • 4 марта 2013 в 17:11

## Популярное за сутки

**Наташа — библиотека для извлечения структурированной информации из текстов на русском языке**

14

alexkuku • вчера в 16:12

**Unit-тестирование скриншотами: преодолеваем звуковой барьер. Расшифровка доклада**

4

lahmatiy • вчера в 13:05

**Люди не хотят чего-то действительно нового — они хотят привычное, но сделанное иначе**

25

ПЕРЕВОД

Smileek • вчера в 10:32

**Руководство по SEO JavaScript-сайтов. Часть 2. Проблемы, эксперименты и рекомендации**

2

ПЕРЕВОД

ru\_vds • вчера в 12:04

**Как адаптировать игру на Unity под iPhone X к апрелю**

0

P1CACHU • вчера в 16:13



## Стивен Хокинг, автор «Краткой истории времени», умер на 77 году жизни

HostingManager • вчера в 13:49

33

## Обзор рынка моноколес 2018

lozga • вчера в 06:58

70

## «Битва за Telegram»: 35 пользователей подали в суд на ФСБ

alizar • вчера в 15:14

40

## Стивен Хокинг и его работа — что дал ученый человечеству?

marks • вчера в 14:46

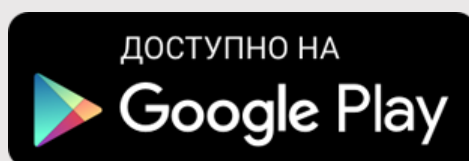
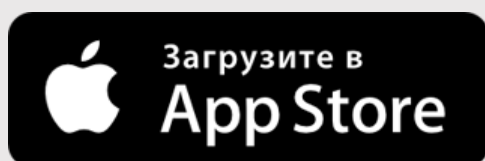
8

## Sunlike — светодиодный свет нового поколения

AlexeyNadezhin • вчера в 20:32

17

Мобильное приложение



Полная версия

