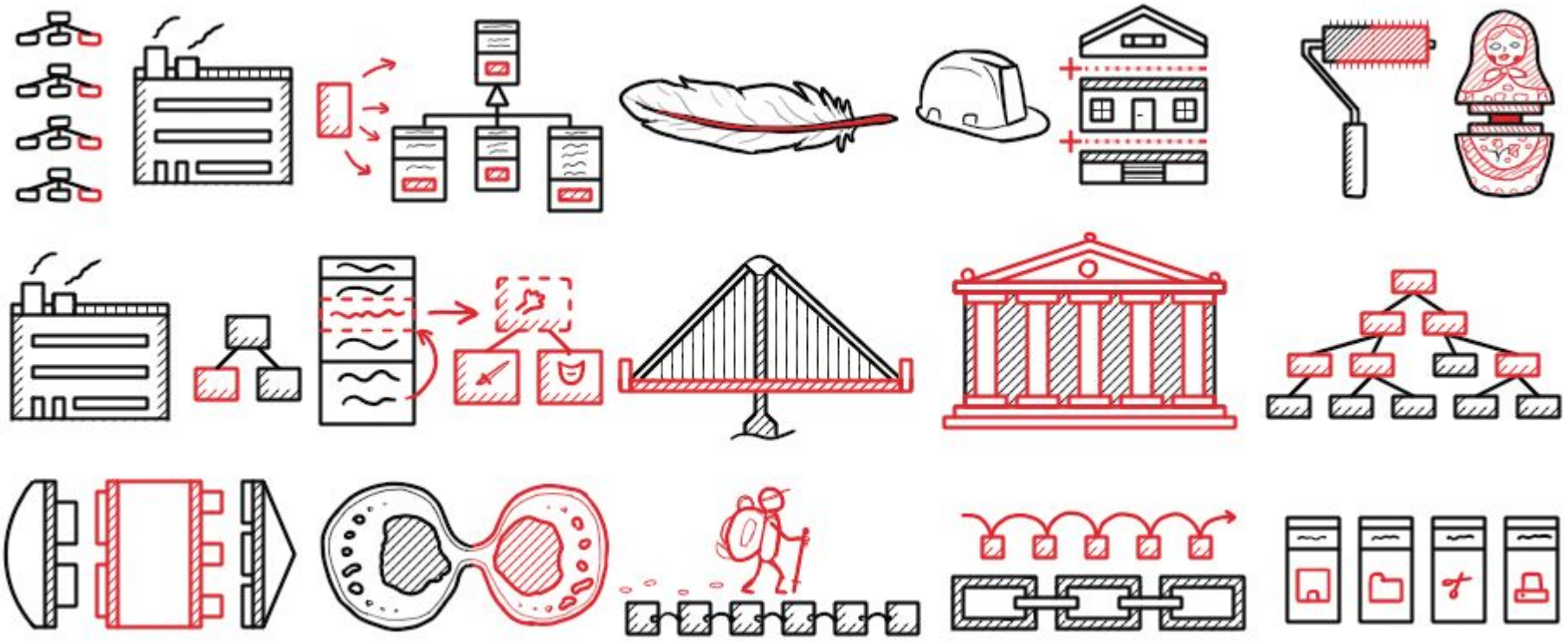


Как я работал с кодом

Сначала было так:

```
<?php
$files = scandir('lib/');
for ($i = 2; $i< sizeof ($files); $i++){
    include_once ('lib/' . $files[$i]);
}
?>
```

Потом стало так:

```
class ProductController extends Controller {

}
```

Потом я пришел сюда

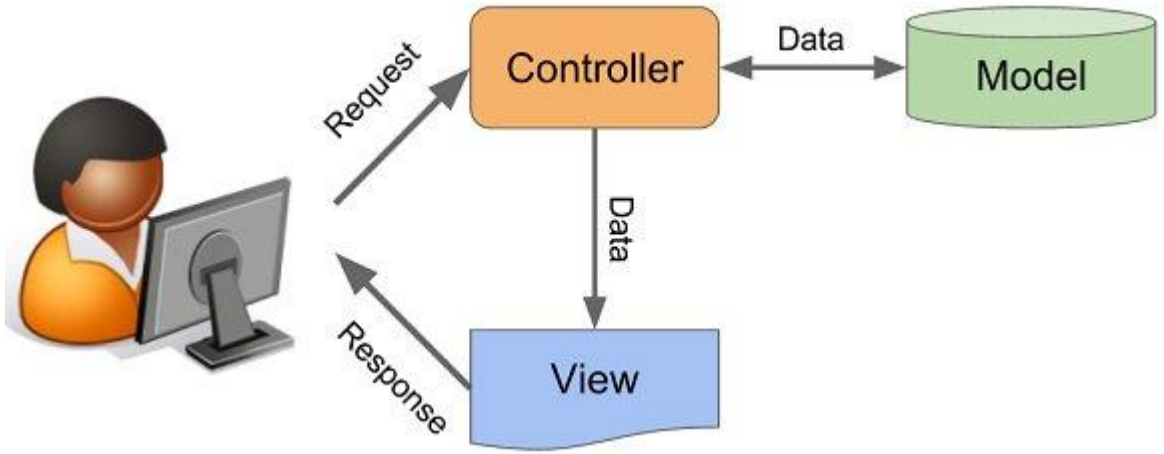
```
<?xml version="1.0"?>

<config

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="urn:magento:framework:
Module/etc/module.xsd">

<module name="Ninydev_Module"
setup_version="0.0.1"></module>

</config>
```

Пока мне не захотелось разобраться

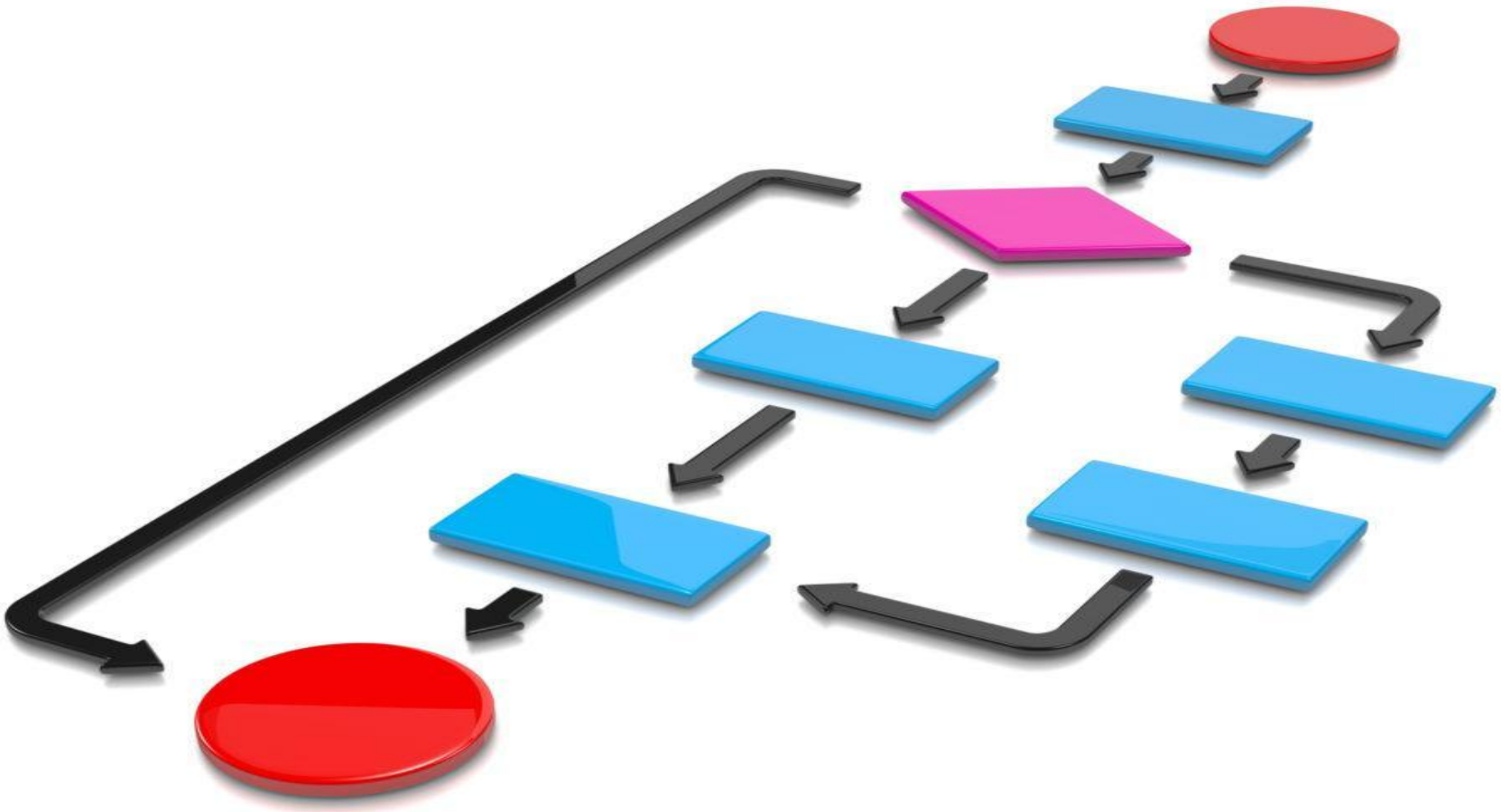
```
<?php


abstract class Controller {

	protected $registry;

	public function __construct($registry) {
		$this->registry = $registry;
	}

	public function __get($key) {
		return $this->registry->get($key);
	}

	public function __set($key, $value) {
		$this->registry->set($key, $value);
	}

}
```

Что было на самом деле.

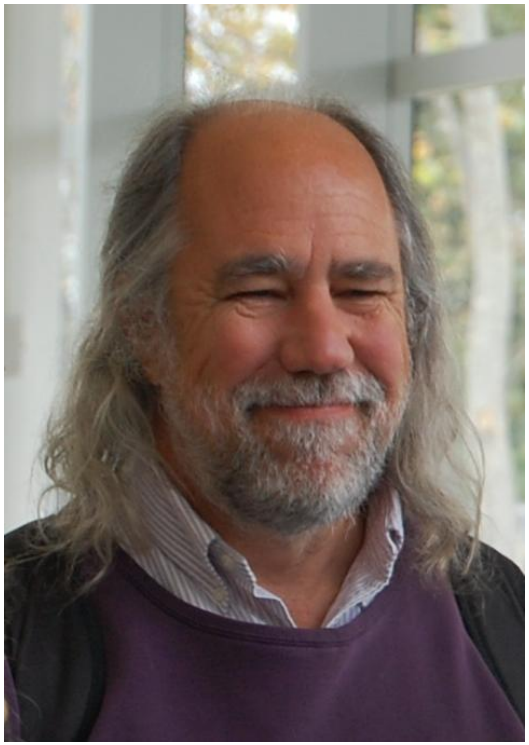
Почему я остановился в росте?

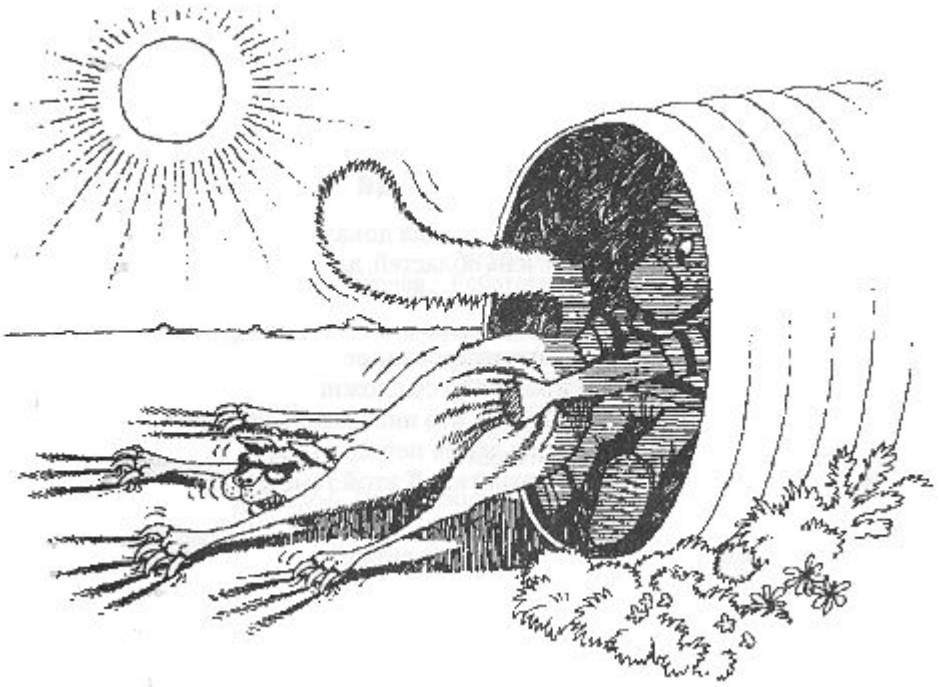


Что я знал хорошо



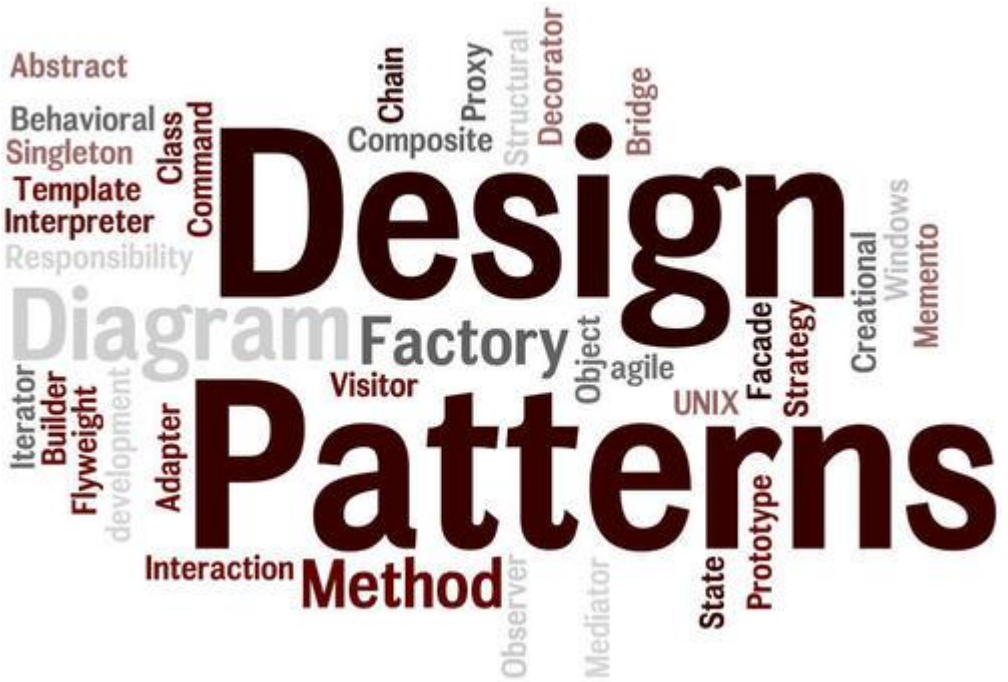
Кого я забыл?





Банда четырех (GoF)

- **Ralph Johnson, Richard Helm, Erich Gamma, and John Vlissides (left to right)**


Abstract
Behavioral
Singleton
Template
Interpreter
Responsibility
Class
Command
Chain
Composite
Proxy
Structural
Decorator
Bridge
Design
Diagram
Factory
Visitor
Object
agile
UNIX
Facade
Strategy
Creational
Windows
Memento
Iterator
Builder
Flyweight
development
Adapter
Patterns
Interaction
Method
Observer
Mediator
State
Prototype


V.Gusik

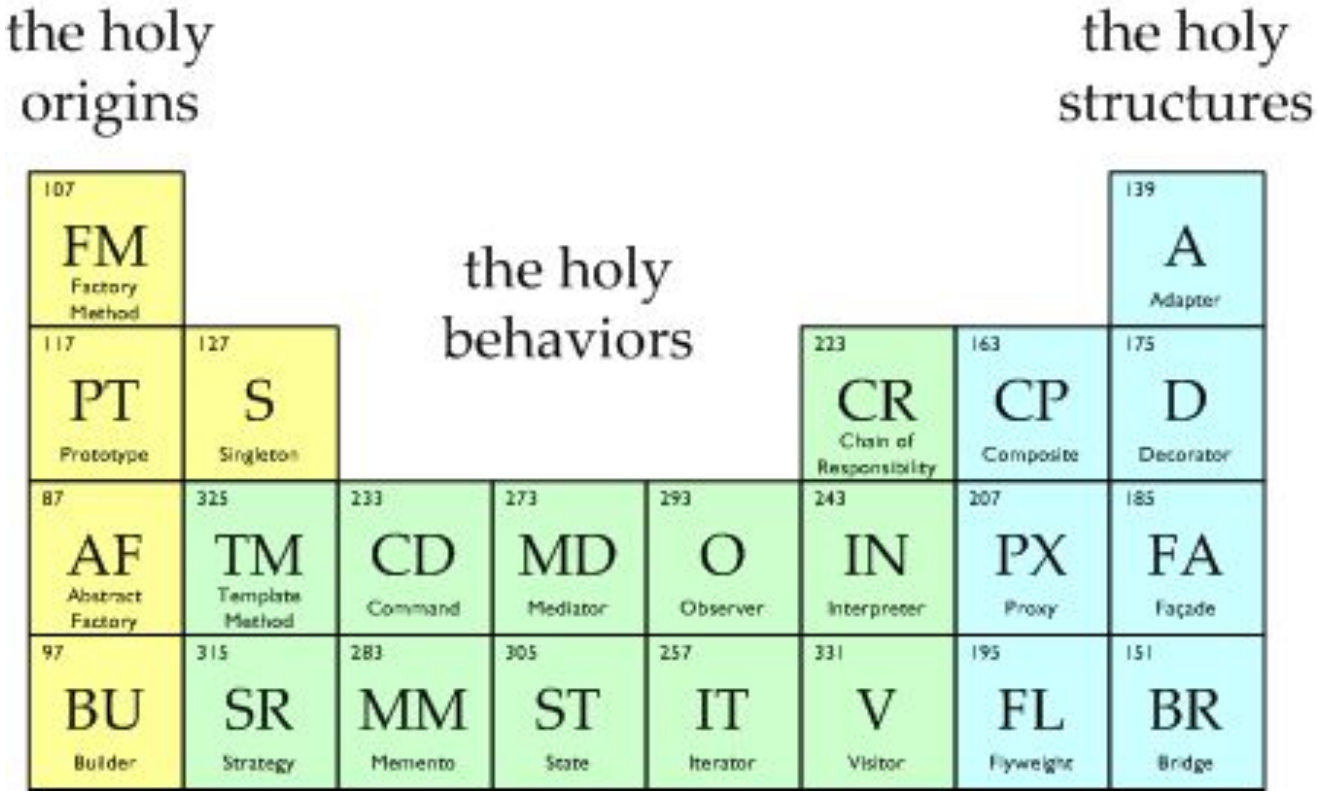
Основные паттерны

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

Виды паттернов

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

The Sacred Elements of the Faith



Перечень порождающих шаблонов

абстрактная фабрика (abstract factory);
строитель (builder);
фабричный метод (factory method);
ленивая инициализация (lazy initialization);
объектный пул (object pool);
прототип (prototype);
одиночка (singleton).
пул одиночек (Multiton)

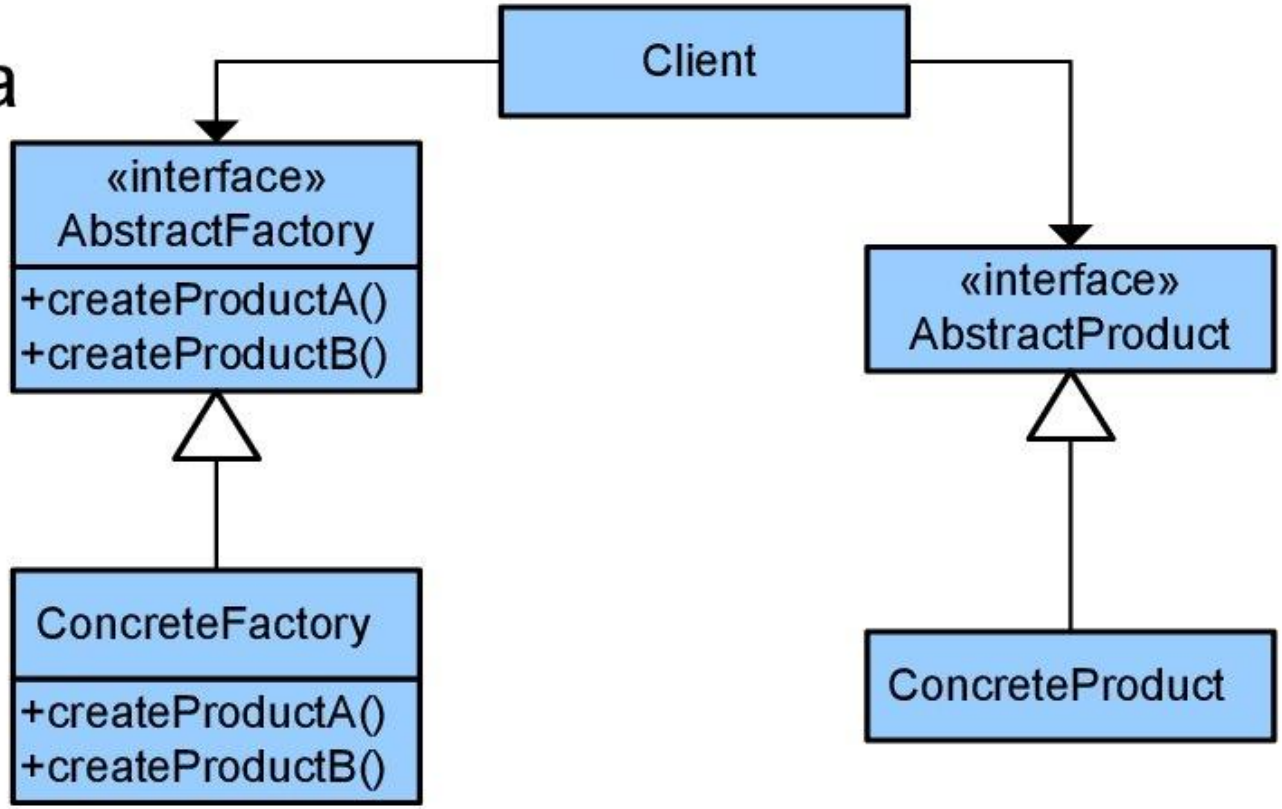
Абстрактная фабрика

Abstract factory

Тип: Порождающий

Что это:

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

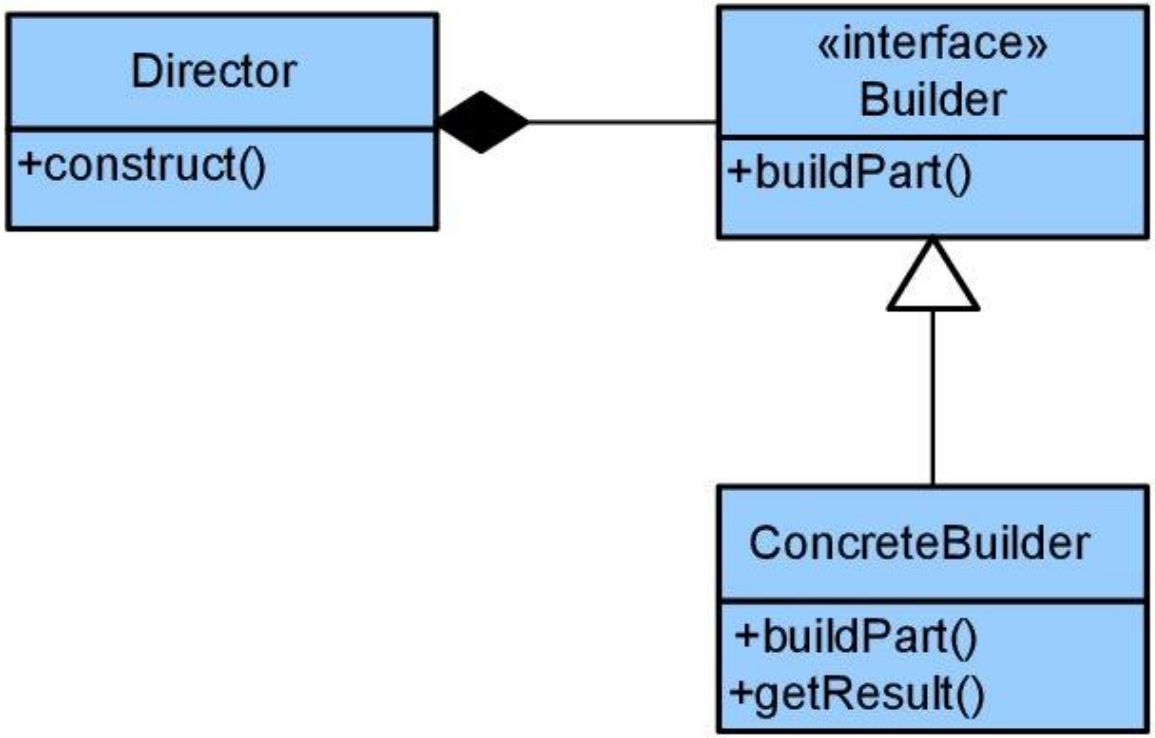


Строитель
Builder

Тип: Порождающий

Что это:

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



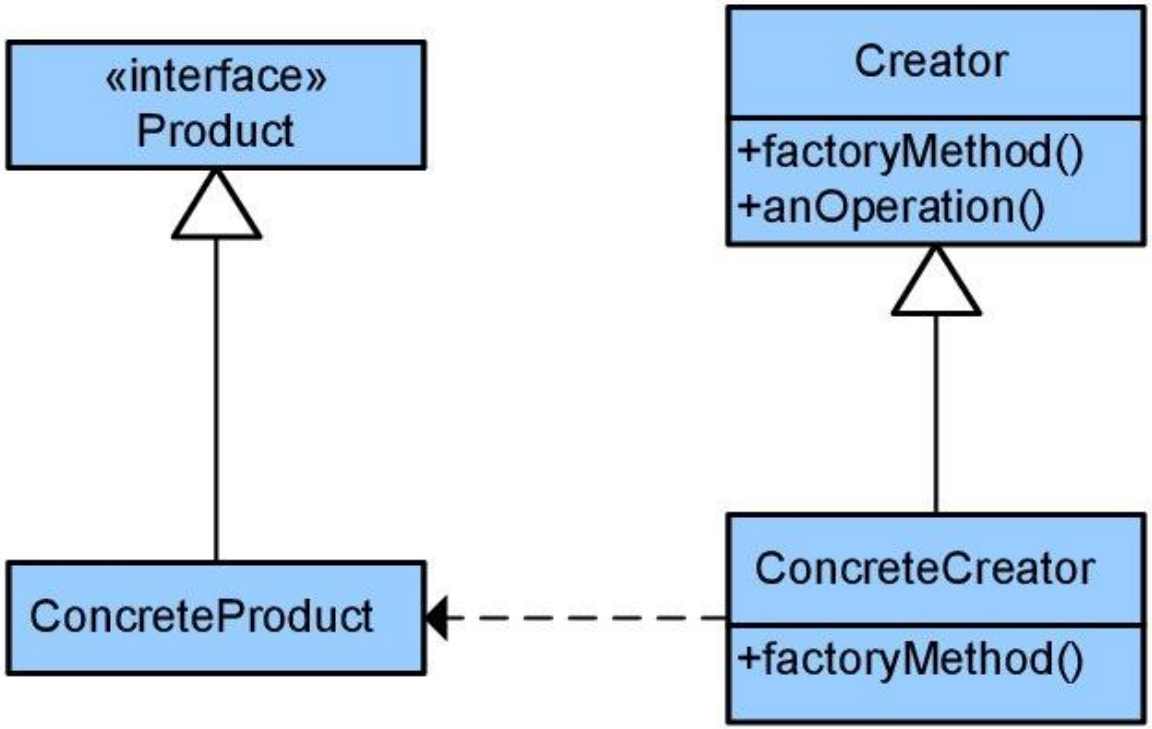
Фабричный метод

Factory method

Тип: Порождающий

Что это:

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

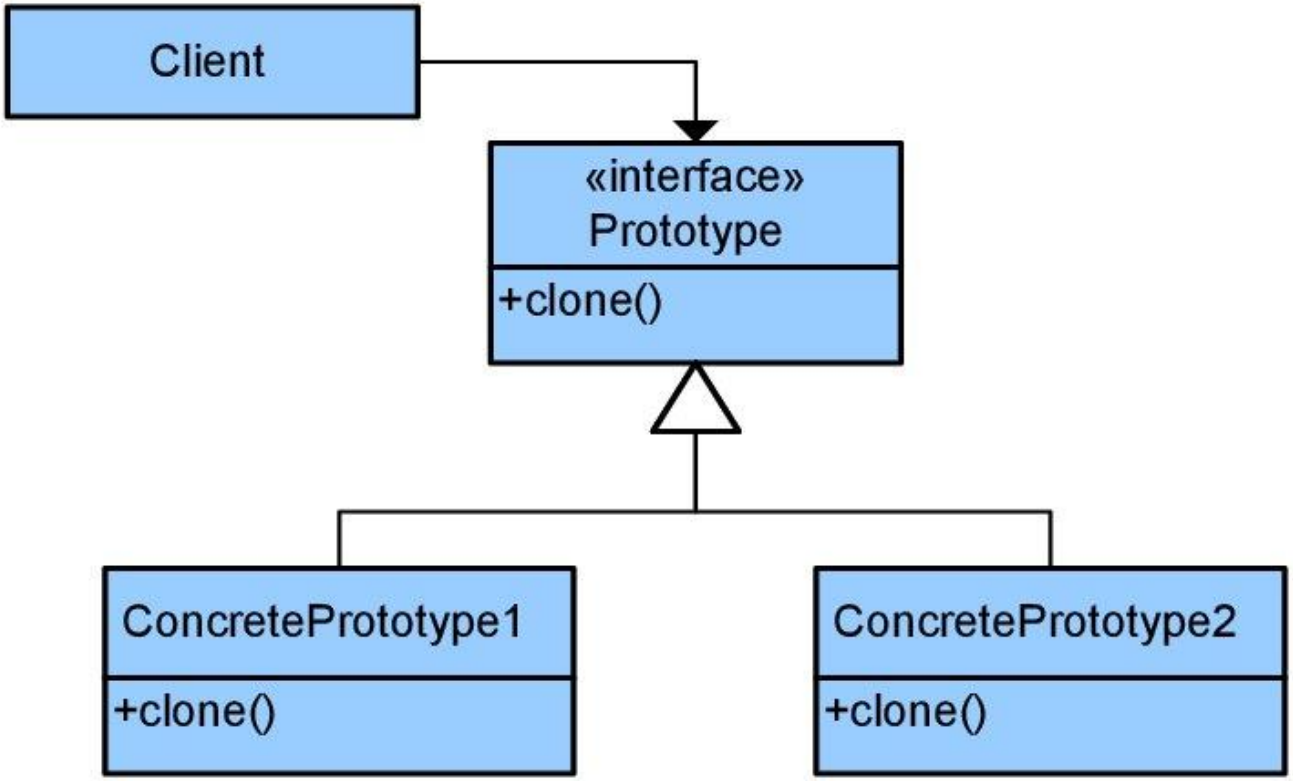


Прототип
Prototype

Тип: Порождающий

Что это:

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



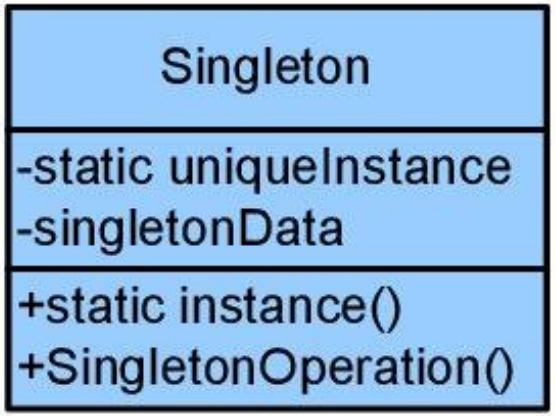
Одиночка

Singleton

Тип: Порождающий

Что это:

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



Поведенческие паттерны проектирования

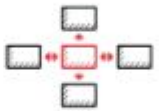
Эти паттерны решают задачи эффективного и безопасного взаимодействия между объектами программы.

Стратегия
Strategy

Состояние
State

Команда
Command

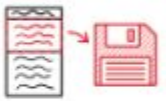
Итератор
Iterator



Посредник
Mediator

Наблюдатель
Observer

Посетитель
Visitor



Снимок
Memento

Цепочка обязанностей
Chain of Responsibility

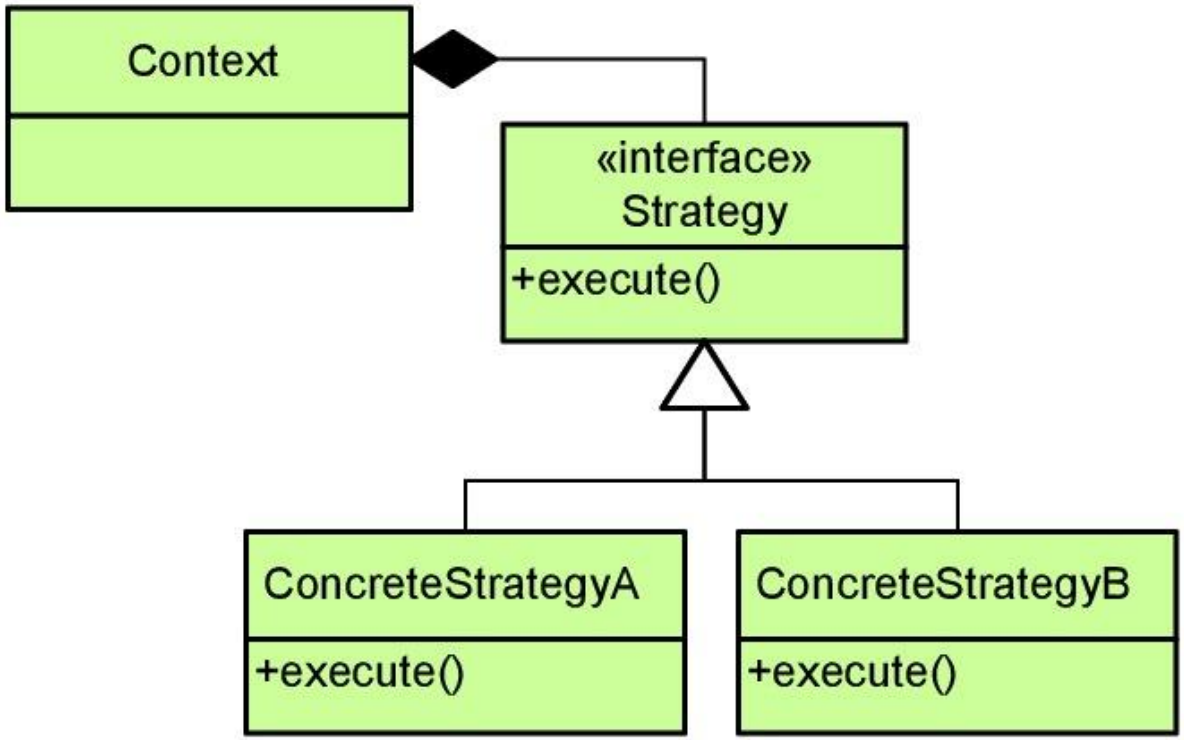
Шаблонный метод
Template method

Стратегия
Strategy

Тип: Поведенческий

Что это:

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

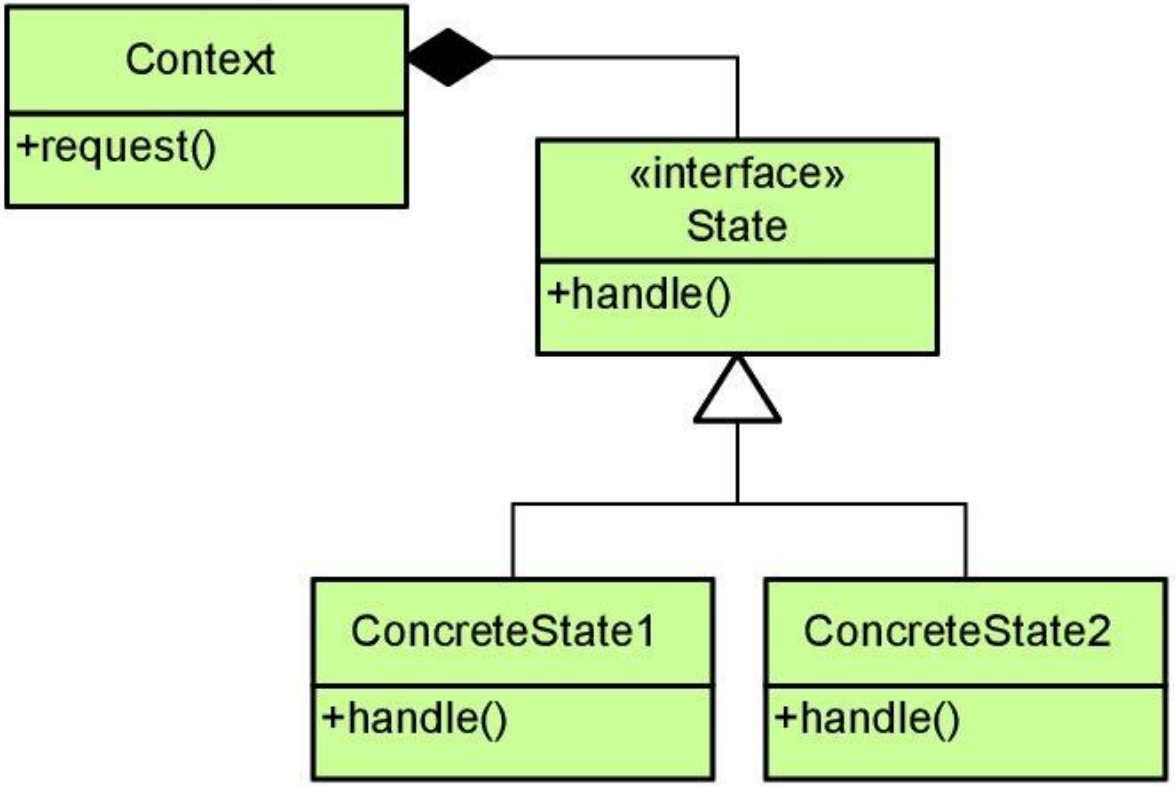


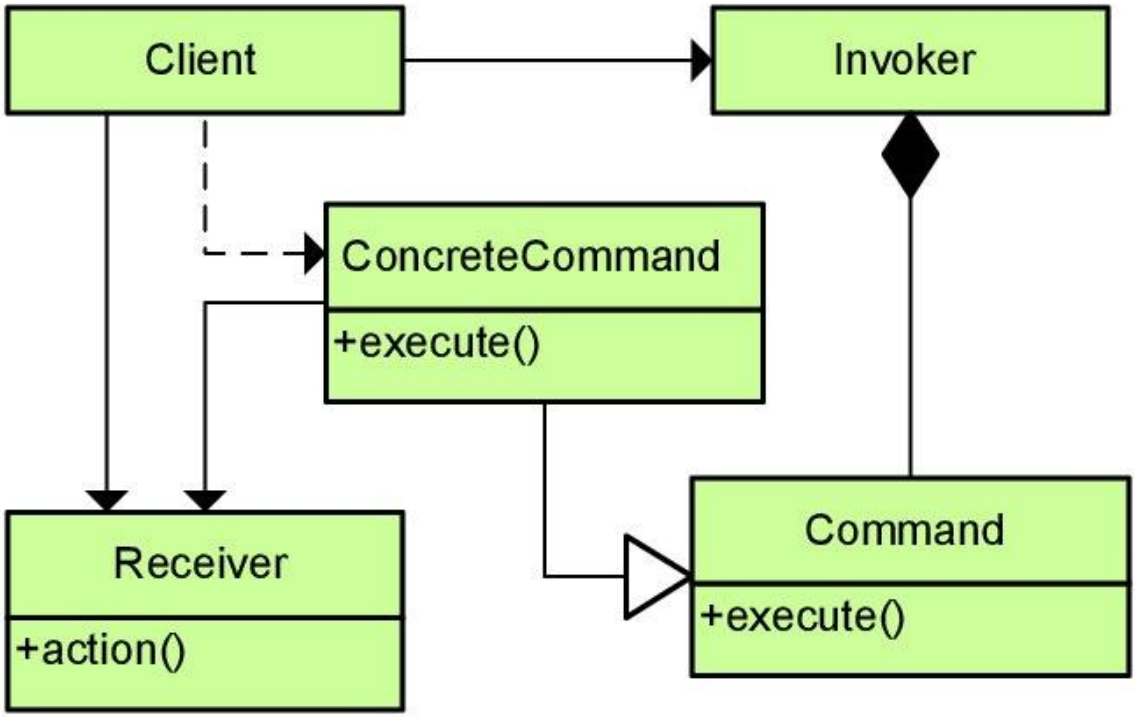
Состояние
State

Тип: Поведенческий

Что это:

Позволяет объекту изменять своё поведение в зависимости от внутреннего состояния.



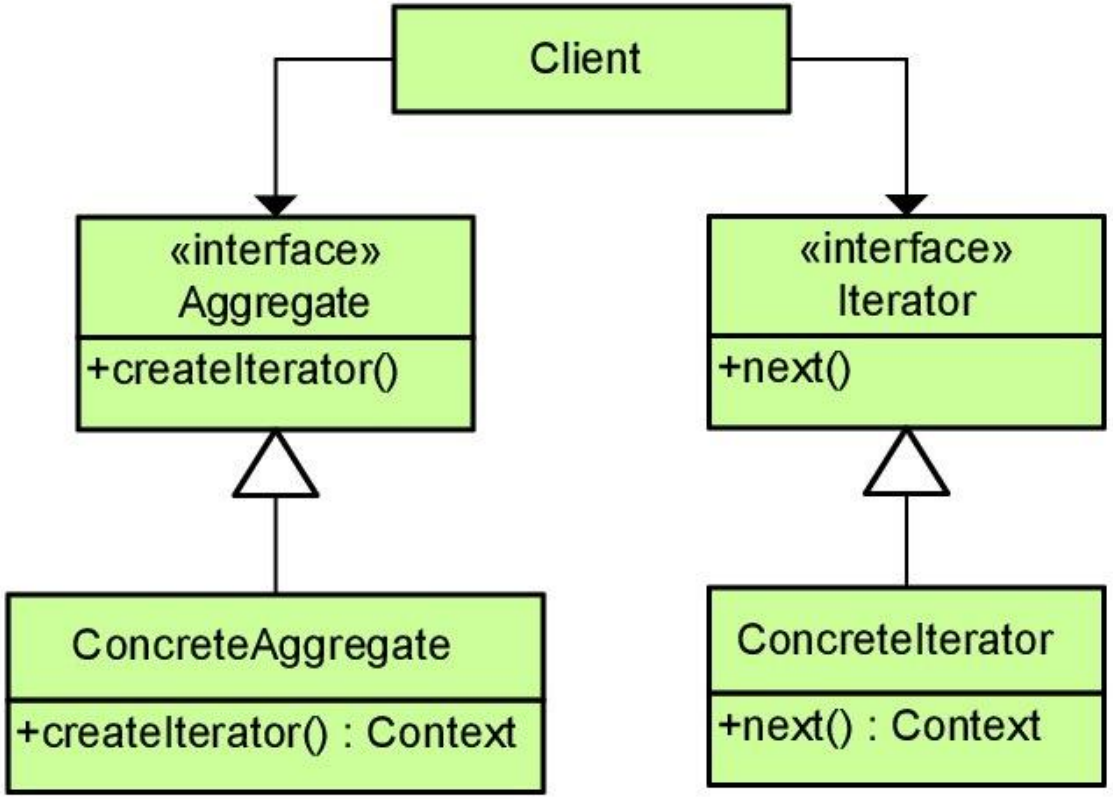


Команда
Command

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.



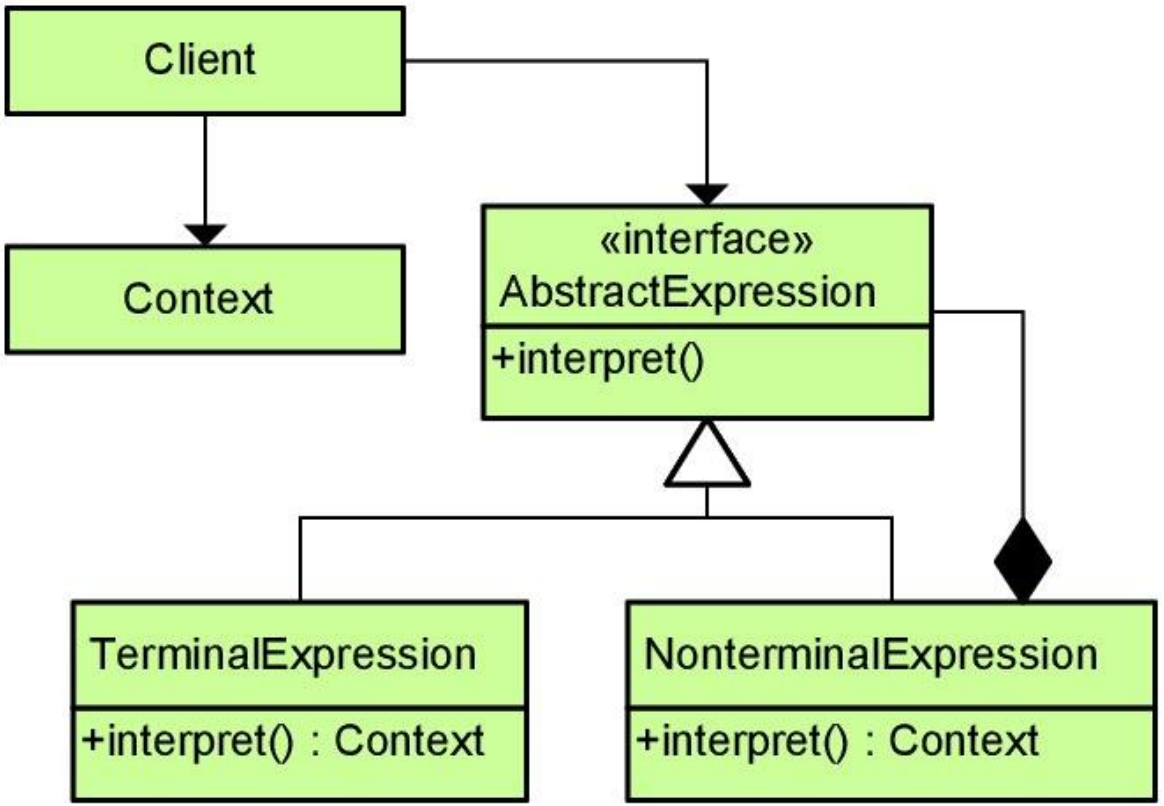
Итератор

Iterator

Тип: Поведенческий

Что это:

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

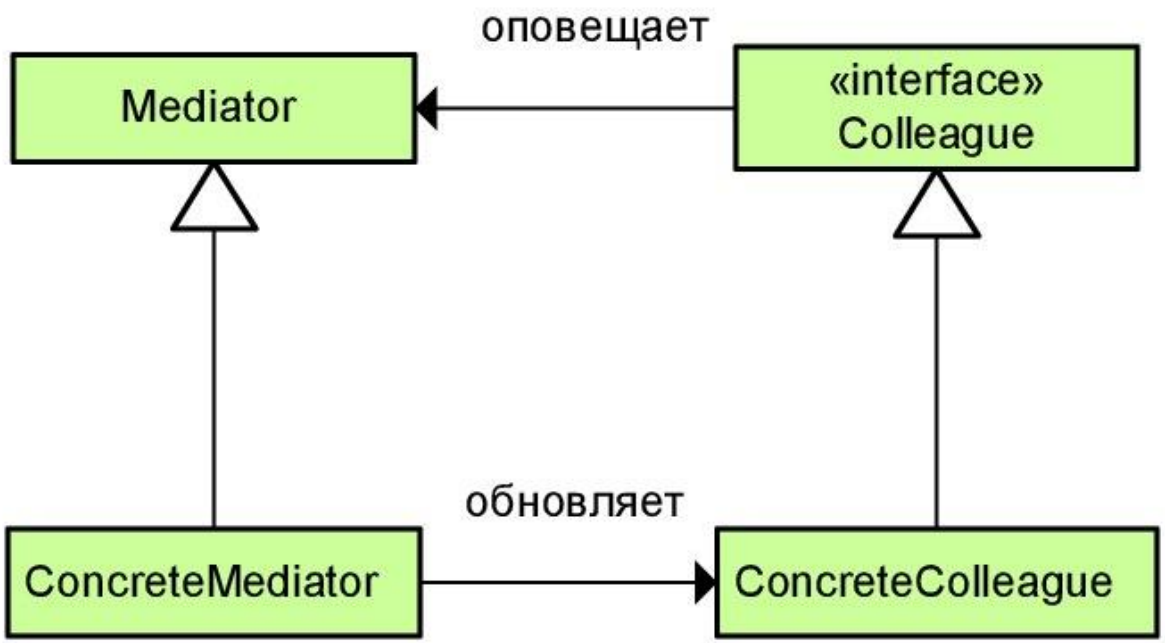


Интерпретатор
Interpreter

Тип: Поведенческий

Что это:

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



Посредник
Mediator

Тип: Поведенческий

Что это:

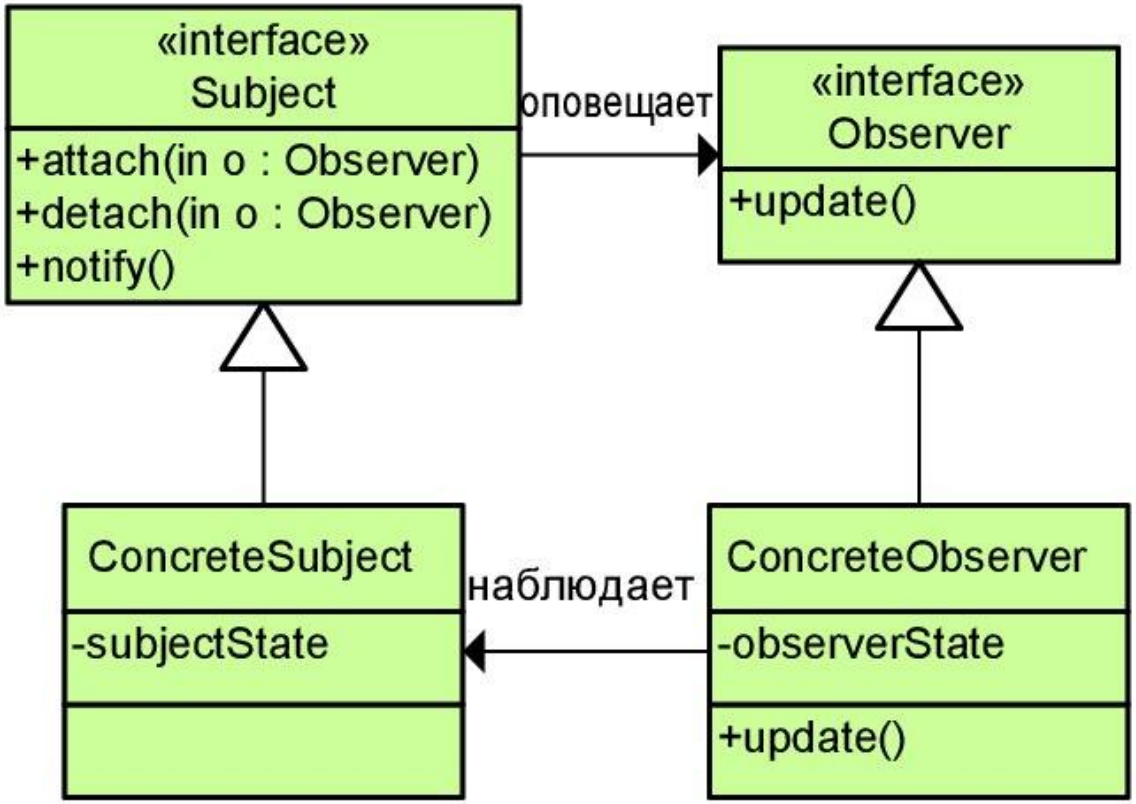
Определяет объект, инкапсулирующий способ взаимодействия объектов. Обеспечивает слабую связь, избавляя объекты от необходимости прямо ссылать друг на друга и даёт возможность независимо изменять их взаимодействие.

Наблюдатель
Observer

Тип: Поведенческий

Что это:

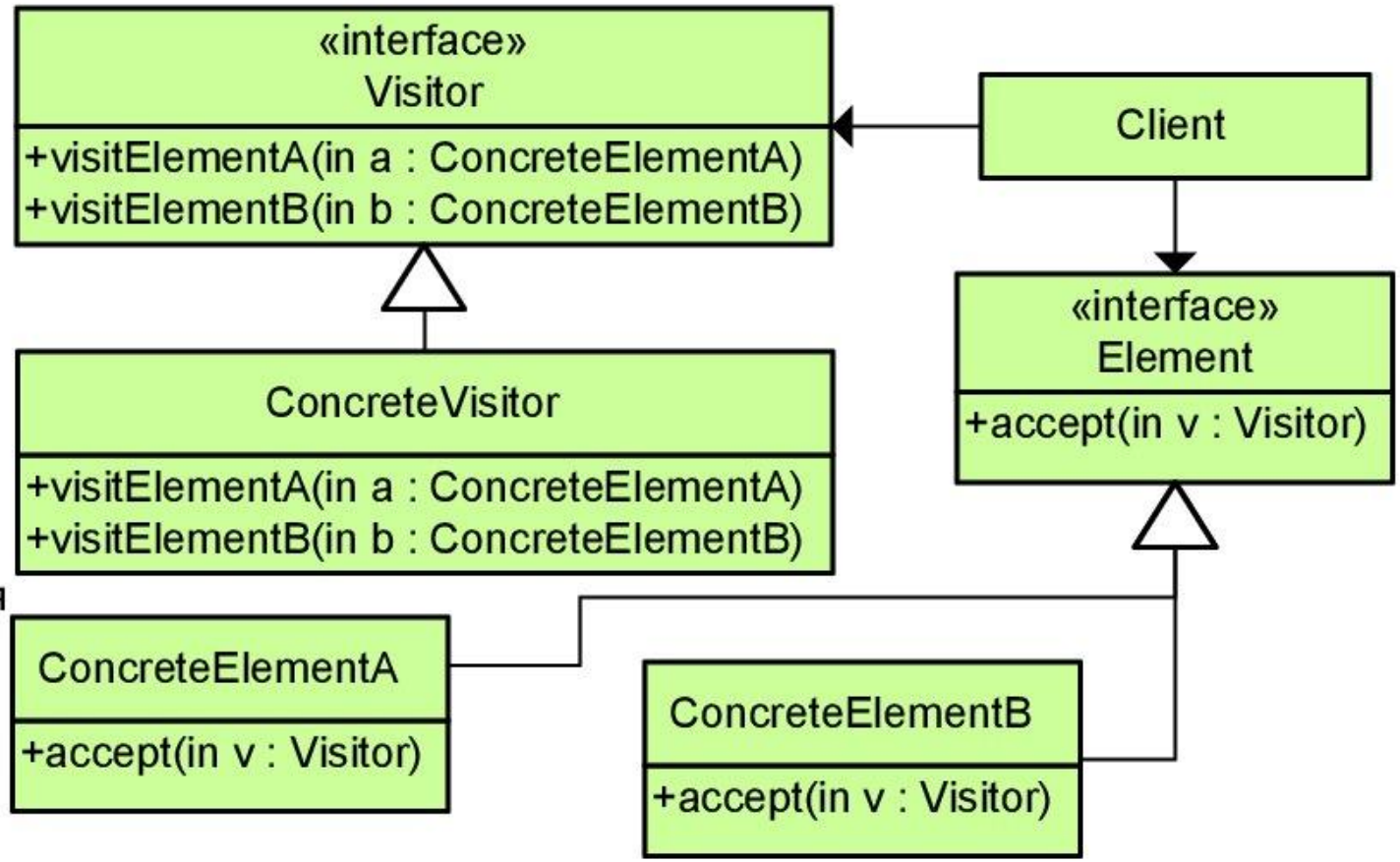
Определяет зависимость “один ко многим” между объектами так, что когда один объект меняет своё состояние, все зависимые объекты оповещаются и обновляются автоматически.



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.

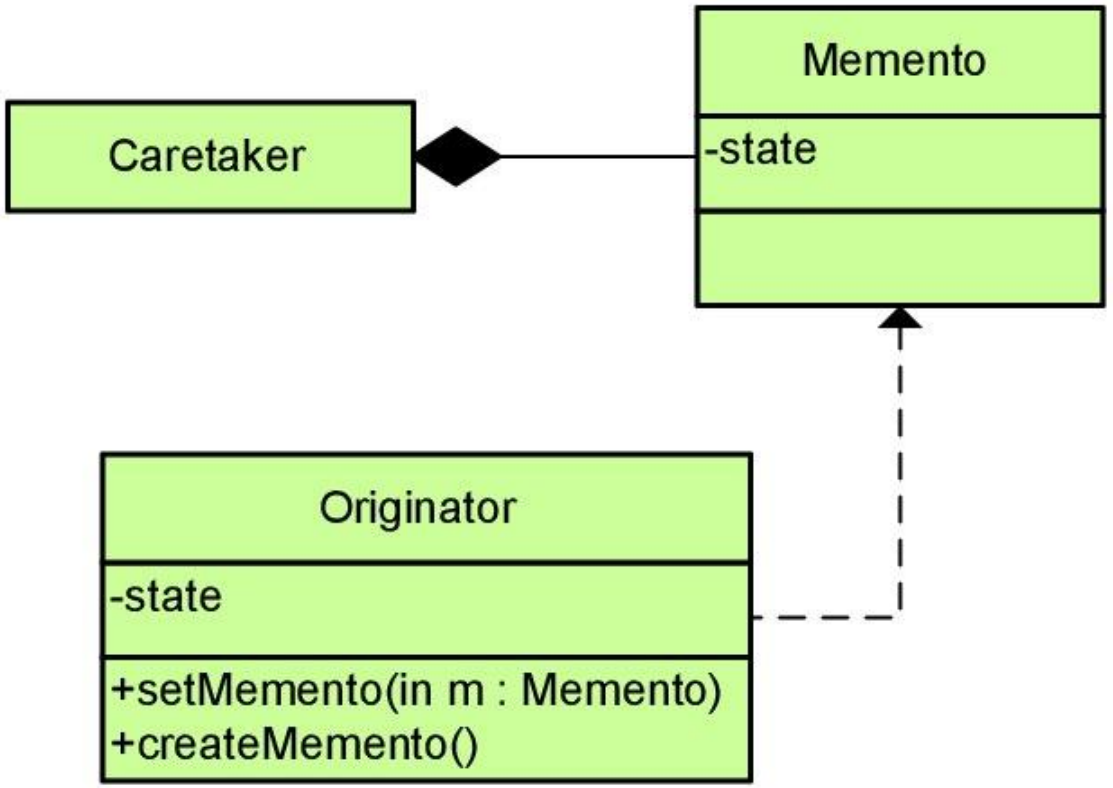


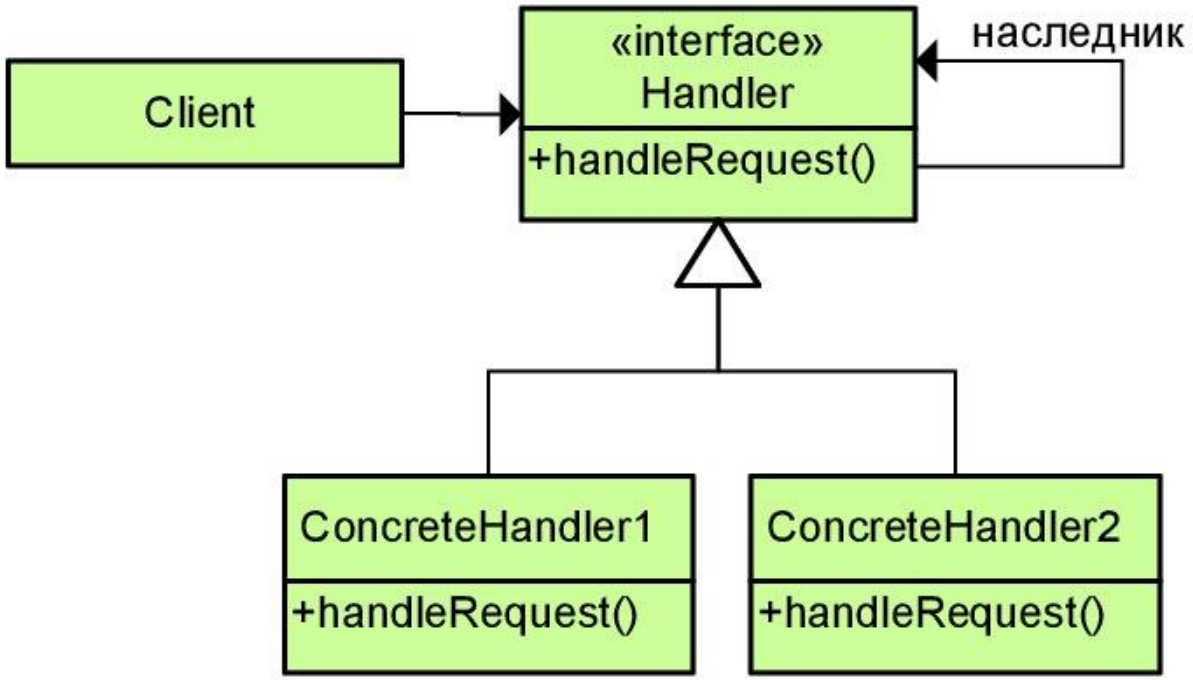
Хранитель
Memento

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

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.

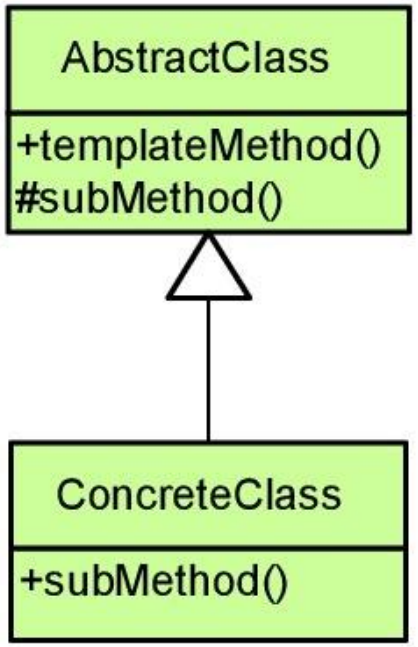
Шаблонный метод

Template method

Тип: Поведенческий

Что это:

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

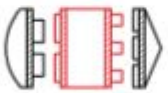


Структурные паттерны проектирования

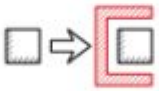
Эти паттерны отвечают за построение удобных в поддержке иерархий классов.

Компоновщик
Composite

Декоратор
Decorator



Адаптер
Adapter

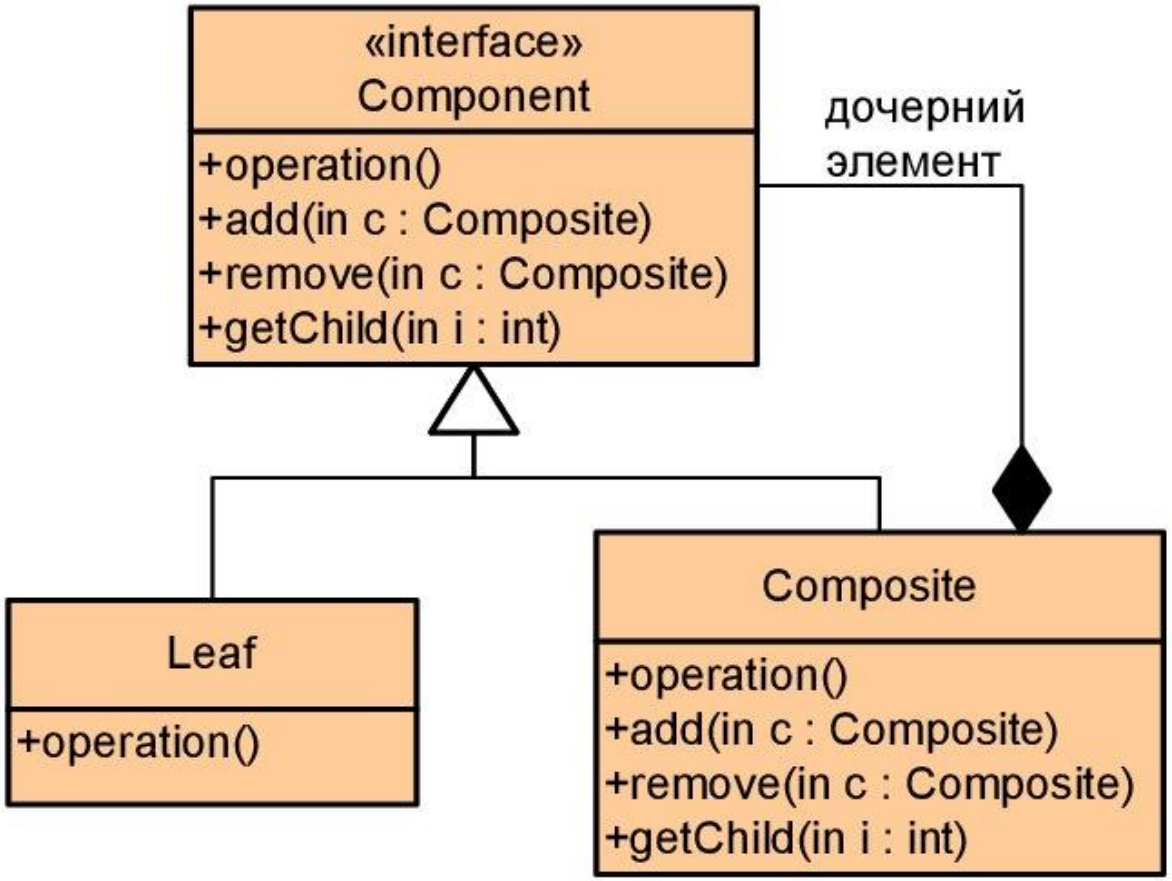


Заместитель
Proxy

Фасад
Façade

Мост
Bridge

Легковес
Flyweight



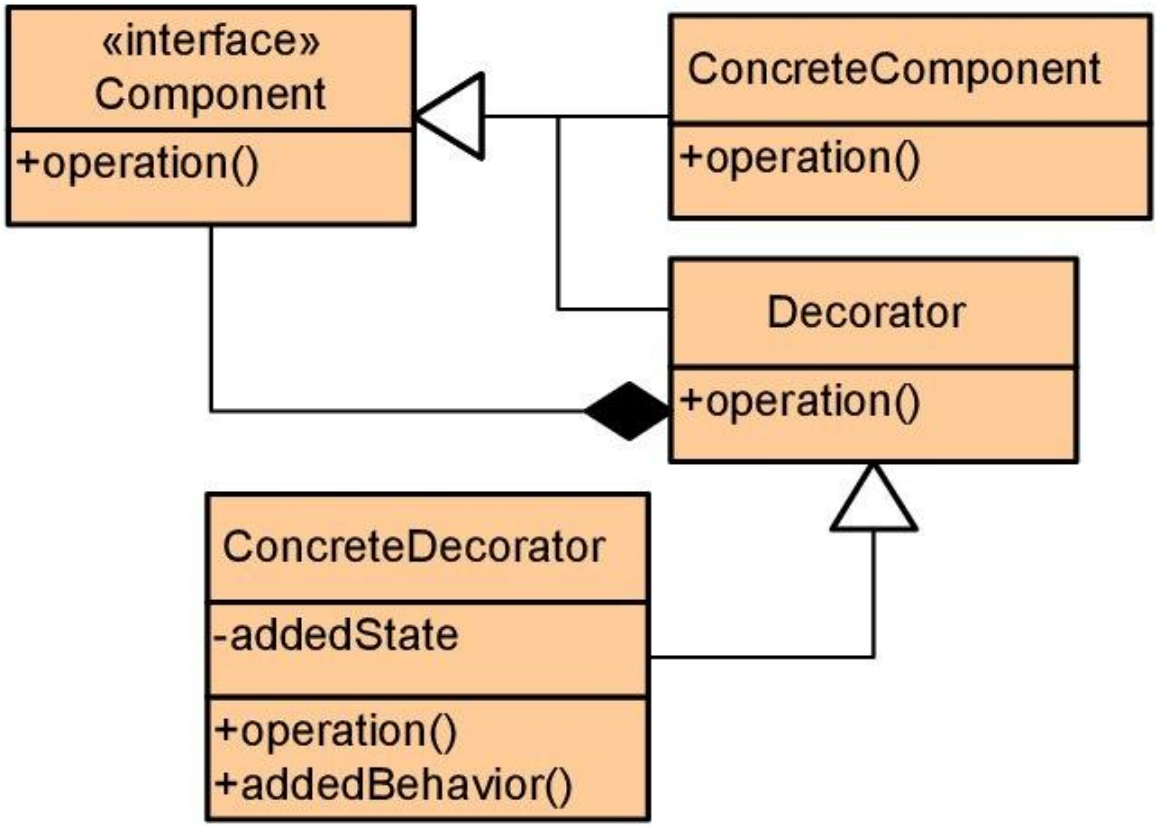
Компоновщик

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.

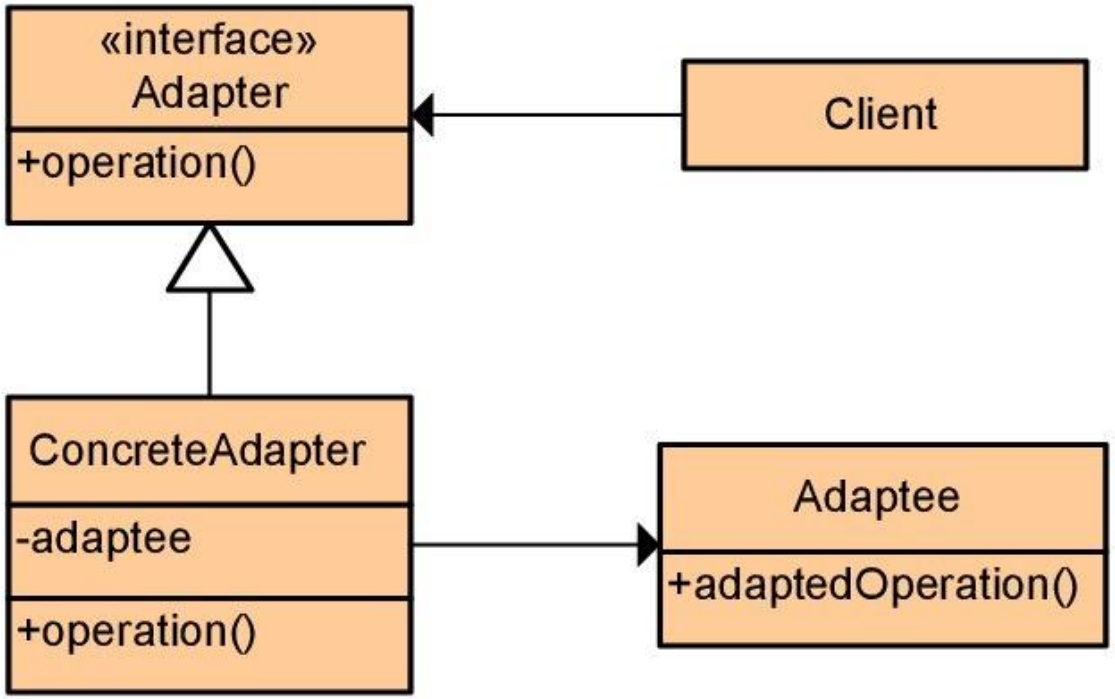


Декоратор
Decorator

Тип: Структурный

Что это:

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



Адаптер

Adapter

Тип: Структурный

Что это:

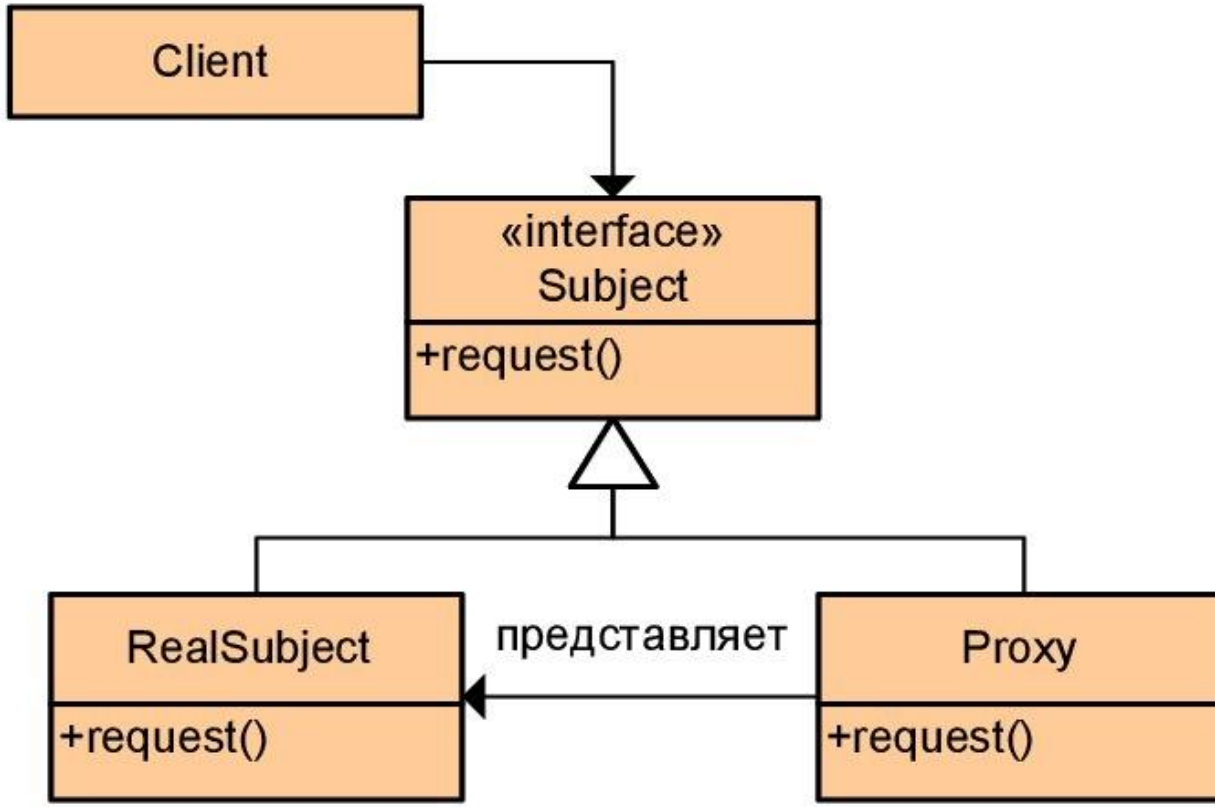
Конвертирует интерфейс класса в другой интерфейс, ожидаемый клиентом. Позволяет классам с разными интерфейсами работать вместе.

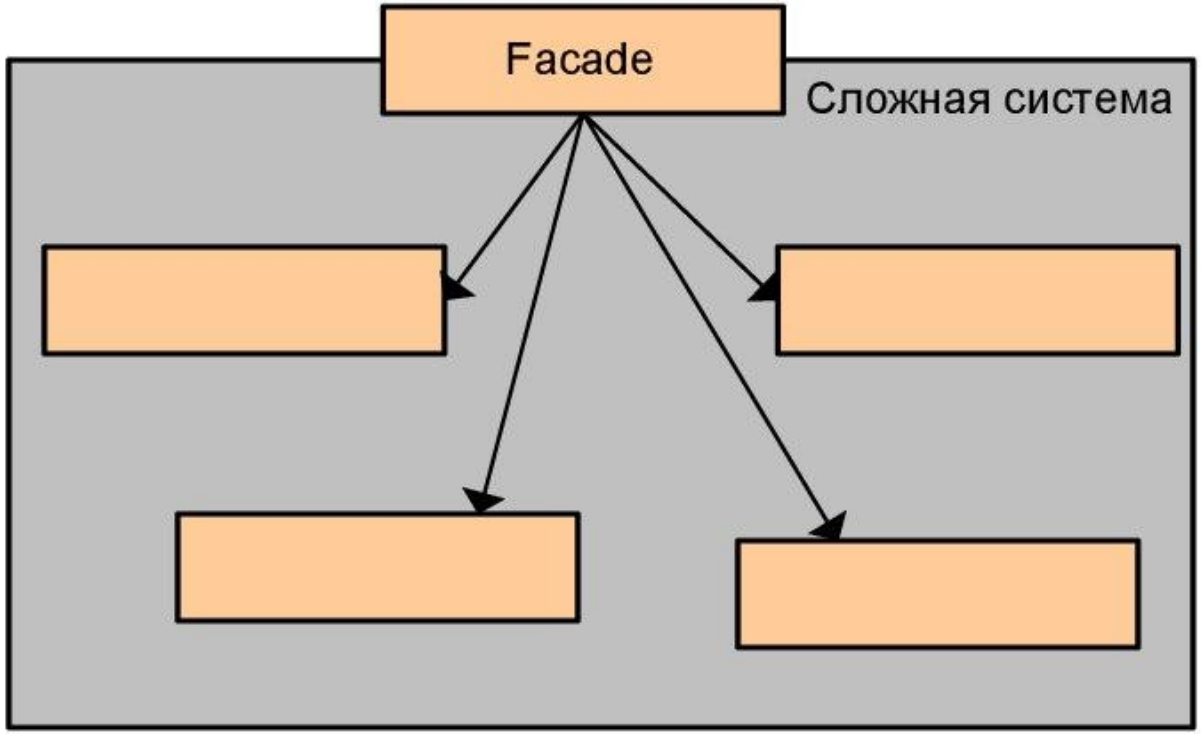
Прокси
Proxy

Тип: Структурный

Что это:

Предоставляет замену другого объекта для контроля доступа к нему.





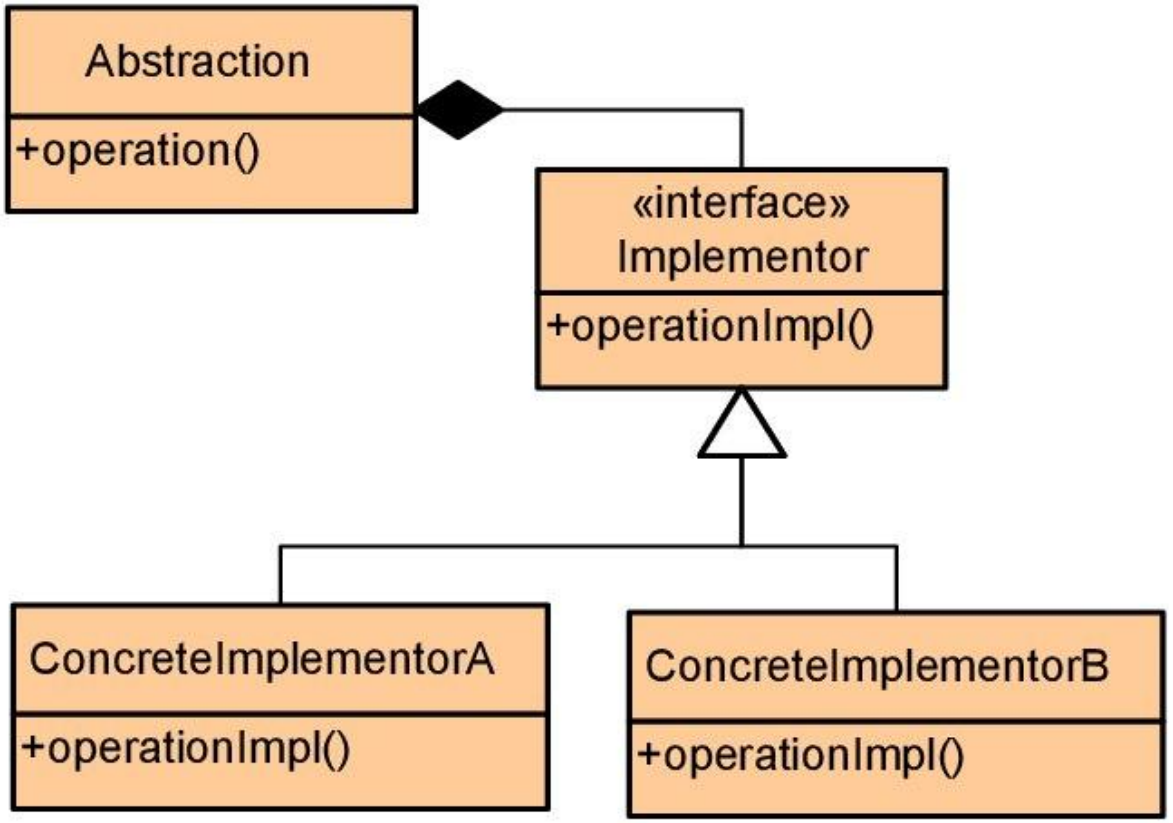
Фасад

Facade

Тип: Структурный

Что это:

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



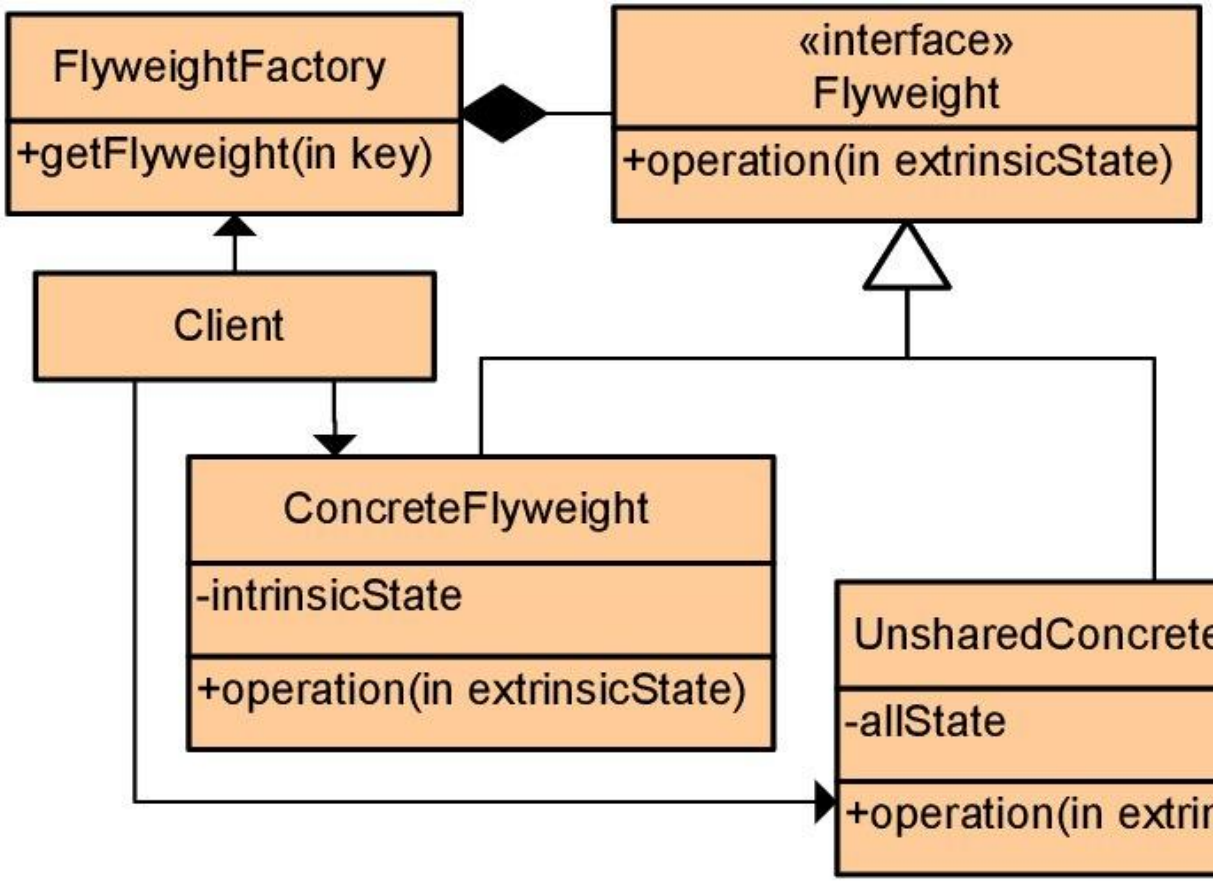
Мост

Bridge

Тип: Структурный

Что это:

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



Приспособленец

Flyweight

Тип: Структурный

Что это:

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