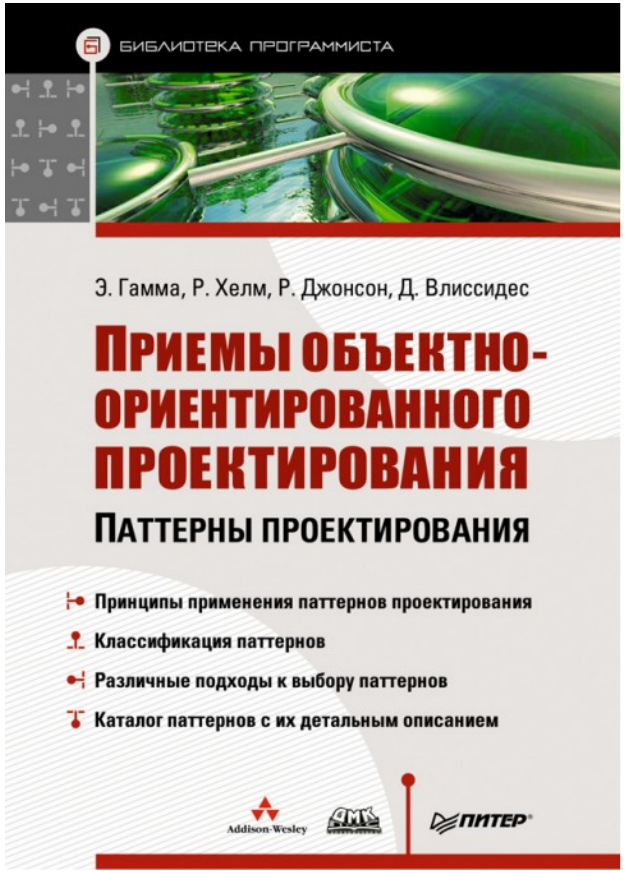
6. Паттерны (шаблоны) проектирования

Christofer Alexander.
«A Pattern Language. Towns. Buildings. Constructions»
1977



Erich Gamma, Richard Helm, Ralph Johnson,John Vlissides
«Design Patterns. Elements of Reusable Object-Oriented Softwart»
1995

Зачем знать паттерны?

Шаблоны упрощают проектирование и поддержку программ.

- **Проверенные решения.**

Ваш код более предсказуем когда вы используете готовые решения, вместо повторного изобретения велосипеда.

- **Стандартизация кода.**

Использование типовых унифицированных решений — должно быть меньше ошибок.

- **Общий язык.**

По названию шаблона понятно, какой подход вы придумали и какие классы для этого нужны.

Классификация шаблонов

- **Порождающие шаблоны** — для гибкого создания объектов без внесения в программу лишних зависимостей.
- **Структурные шаблоны** — показывают различные способы построения связей между объектами.
- **Поведенческие шаблоны** — для эффективной коммуникации между объектами.

Порождающие шаблоны

- Factory method
- Abstract factory
- Singleton
- …

Структурные шаблоны

- Adapter
- Decorator
- …

Поведенческие шаблоны

- Observer
- Strategy

На третьем ходу выяснилось, что гроссмейстер играет восемнадцать **испанских партий**. В остальных двенадцати черные применили хотя и устаревшую, но довольно верную **защиту Филидора**. Если б Остап узнал, что он играет такие мудреные партии и сталкивается с такой испытанной защитой, он крайне бы удивился. Дело в том, что великий комбинатор играл в шахматы второй раз в жизни.

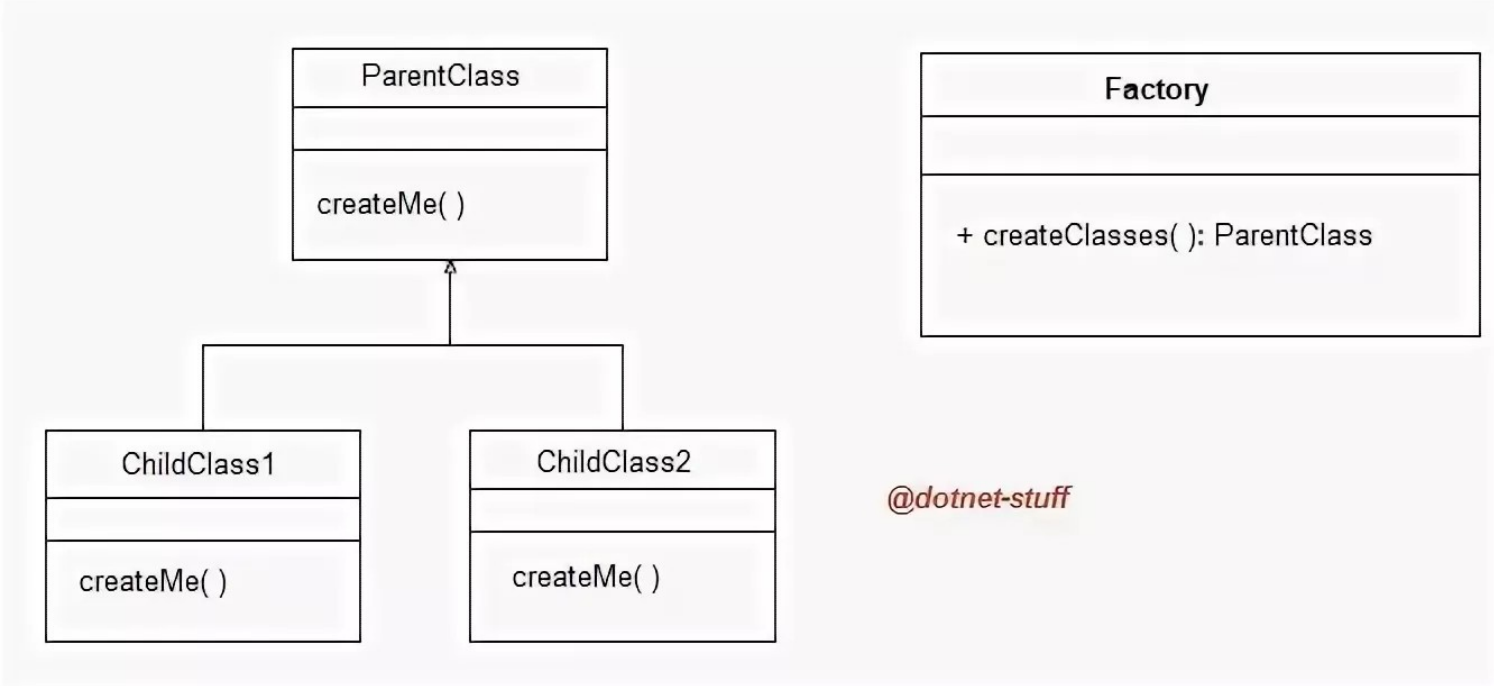
И. Ильф, Е. Петров «Двенадцать стульев»

Simple Factory

Фабрика — объект, создающий другие объекты.
Простая фабрика генерирует экземпляр класса для клиента.

Когда использовать?

Когда создание объекта подразумевает какую-то логику, а не просто несколько присваиваний, то имеет смысл делегировать задачу

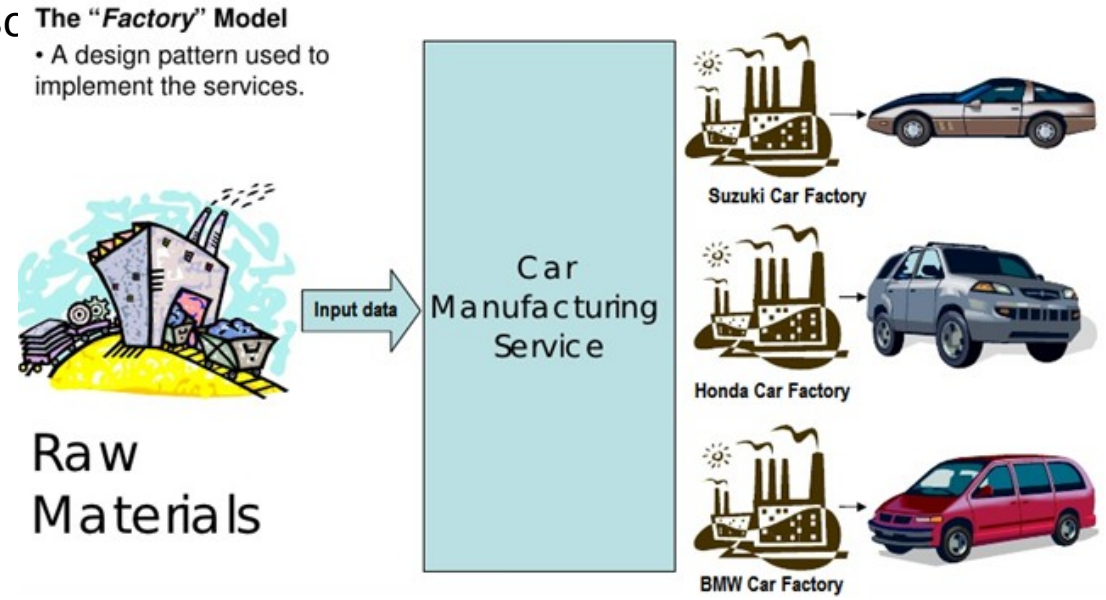


Factory method

Способ делегирования логики создания объектов дочерним классам.

Когда использовать?

- Когда заранее неизвестно, объекты каких типов необходимо создавать;
- Когда система должна быть независимой от процесса создания новых объектов и расширяемой: в нее можно легко вводить новые классы, объекты которых система должна создавать;
- Когда создание новых объектов необходимо делегировать из базовс



Builder

Позволяет создавать разные свойства объекта, избегая загрязнения конструктора.
Это полезно, когда у объекта может быть несколько свойств. Или когда создание объекта состоит из большого количества этапов.

Когда использовать?

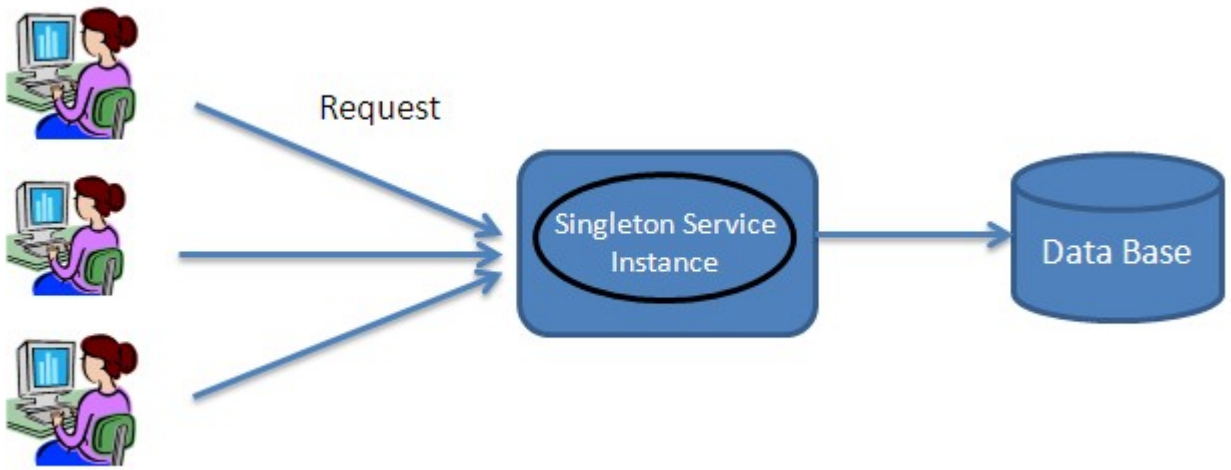
- Когда у объекта много свойств;
- Когда создание объекта состоит из большого количества этапов.

Шаблон Builder решает проблему антишаблона Telescoping constructor:

```
public function __construct($size, $cheese = true, $pepperoni = true, $tomato = false, $lettuce = true)
{
}
```

Singleton

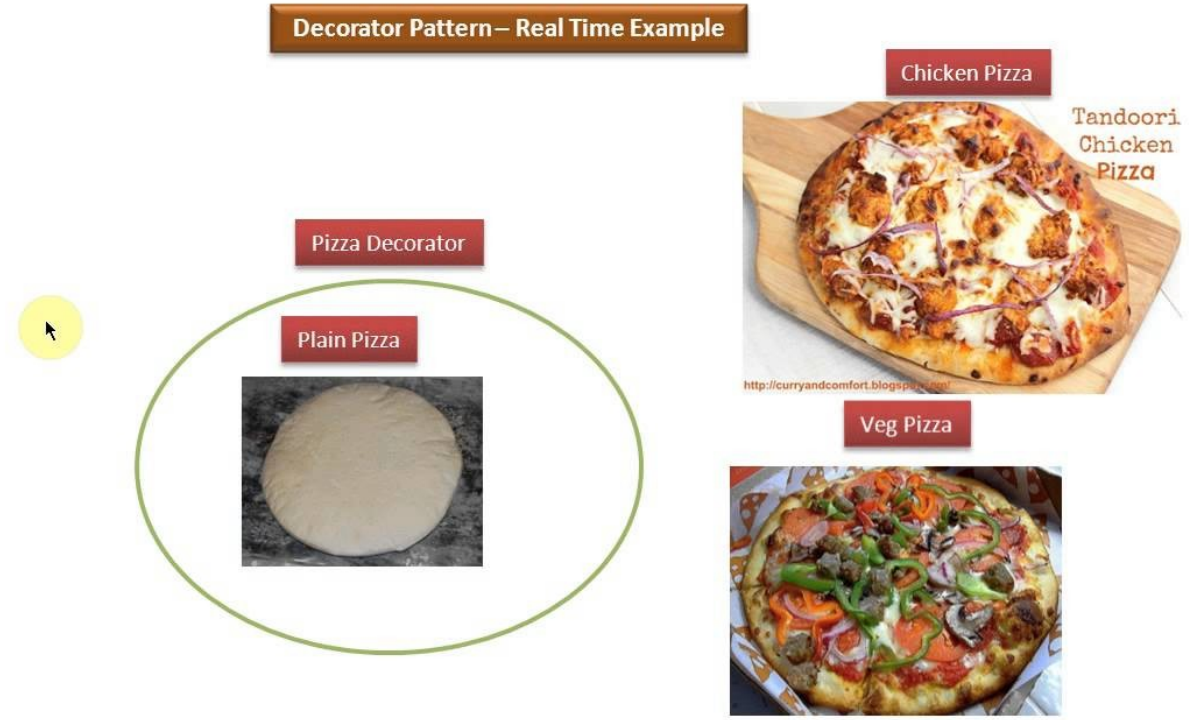
Гарантирует, что класс имеет только один экземпляр и предоставляет глобальную точку доступа к нему.



Singleton
—instance : Singleton
—Singleton() +getInstance() : Singleton

Decorator

Позволяет подключать к объекту дополнительное поведение (статически или динамически), не влияя на поведение других объектов того же класса.
Является гибкой альтернативой порождению подклассов с целью расширения функциональности.



Strategy

Позволяет переключаться между алгоритмами или стратегиями в зависимости от ситуации (независимо от клиентов, их использующих).

Альтернатива наследованию (вместо расширения абстрактного класса).