# Java Design Patterns

## Abstract Factory

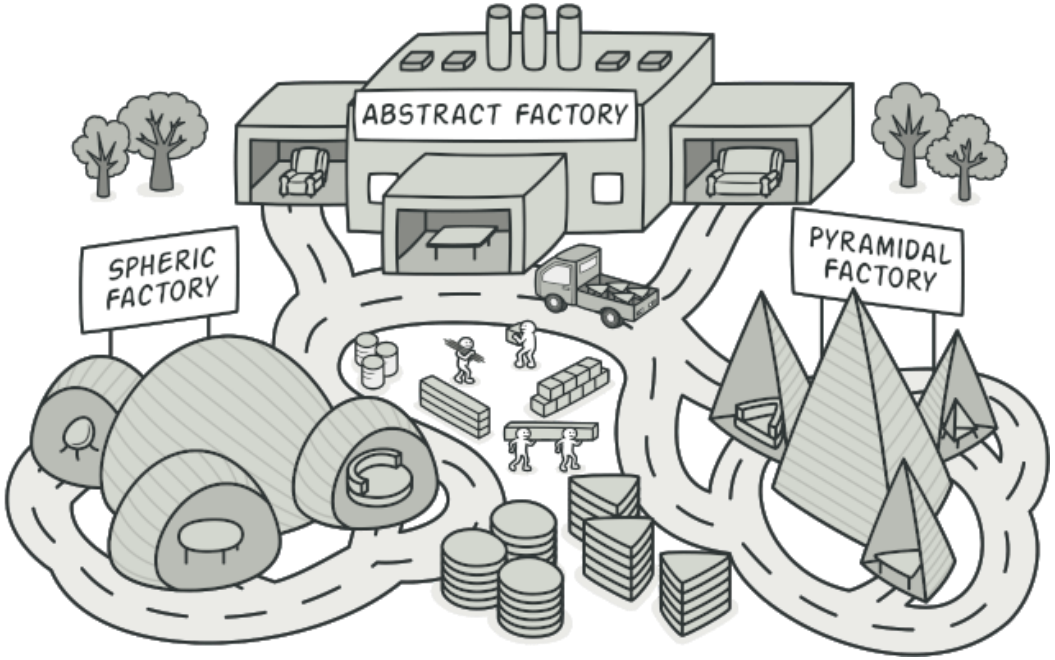
Тема

# Abstract Factory

# Суть паттерна

## Абстрактная фабрика

Абстрактная фабрика — это порождающий паттерн проектирования, который позволяет создавать семейства связанных объектов, не привязываясь к конкретным классам создаваемых объектов.



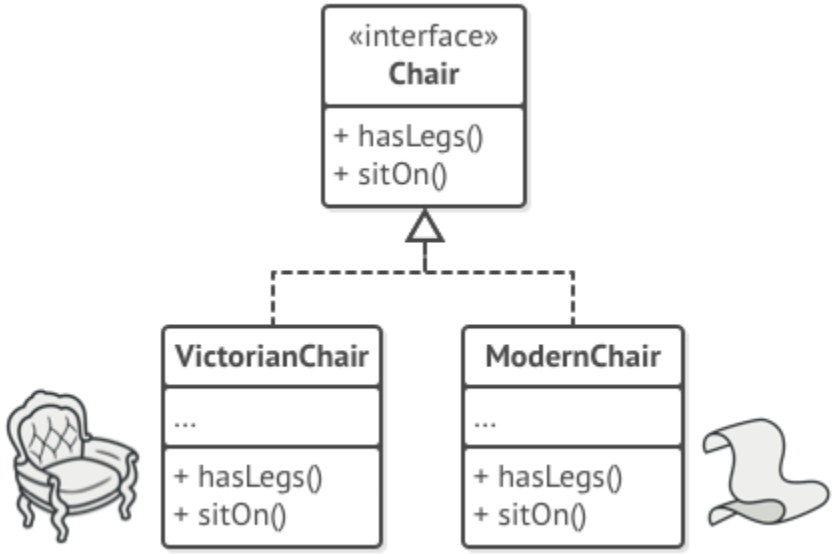
# Проблема

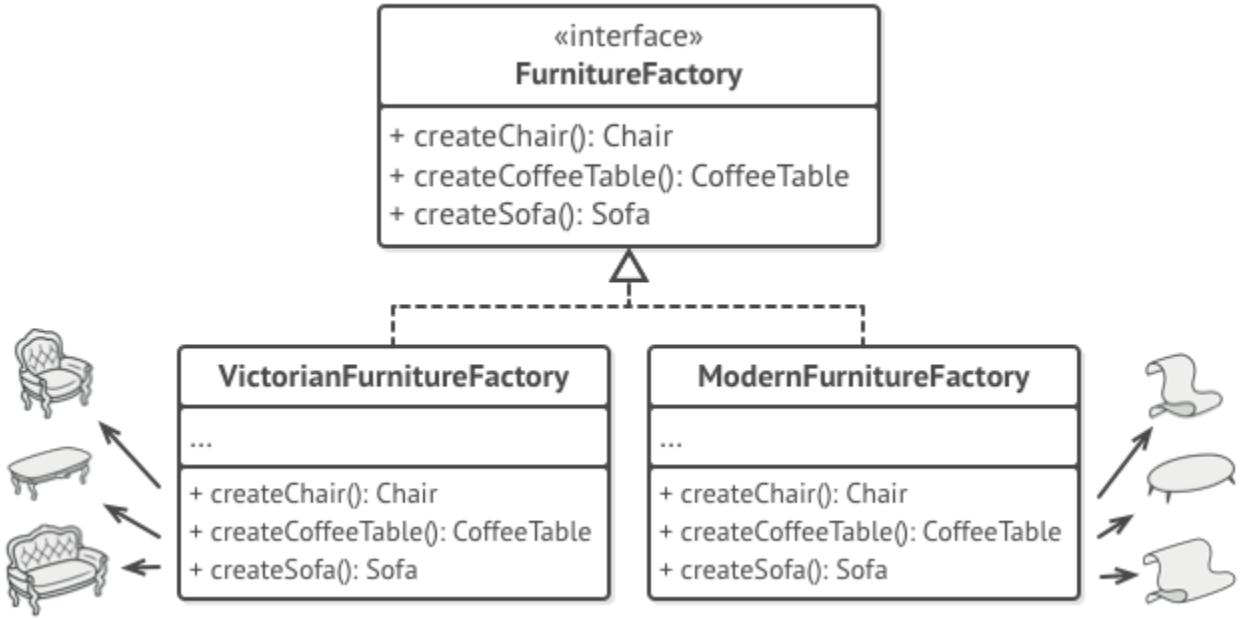
## Постановка задачи

Представьте, что вы пишете симулятор мебельного магазина. Ваш код содержит:
1. Семейство зависимых продуктов. Скажем, Кресло + Диван + Столик.
2. Несколько вариаций этого семейства. Например, продукты Кресло, Диван и Столик представлены в трёх разных стилях: Ар-деко, Викторианском и Модерне.

## Решение задачи

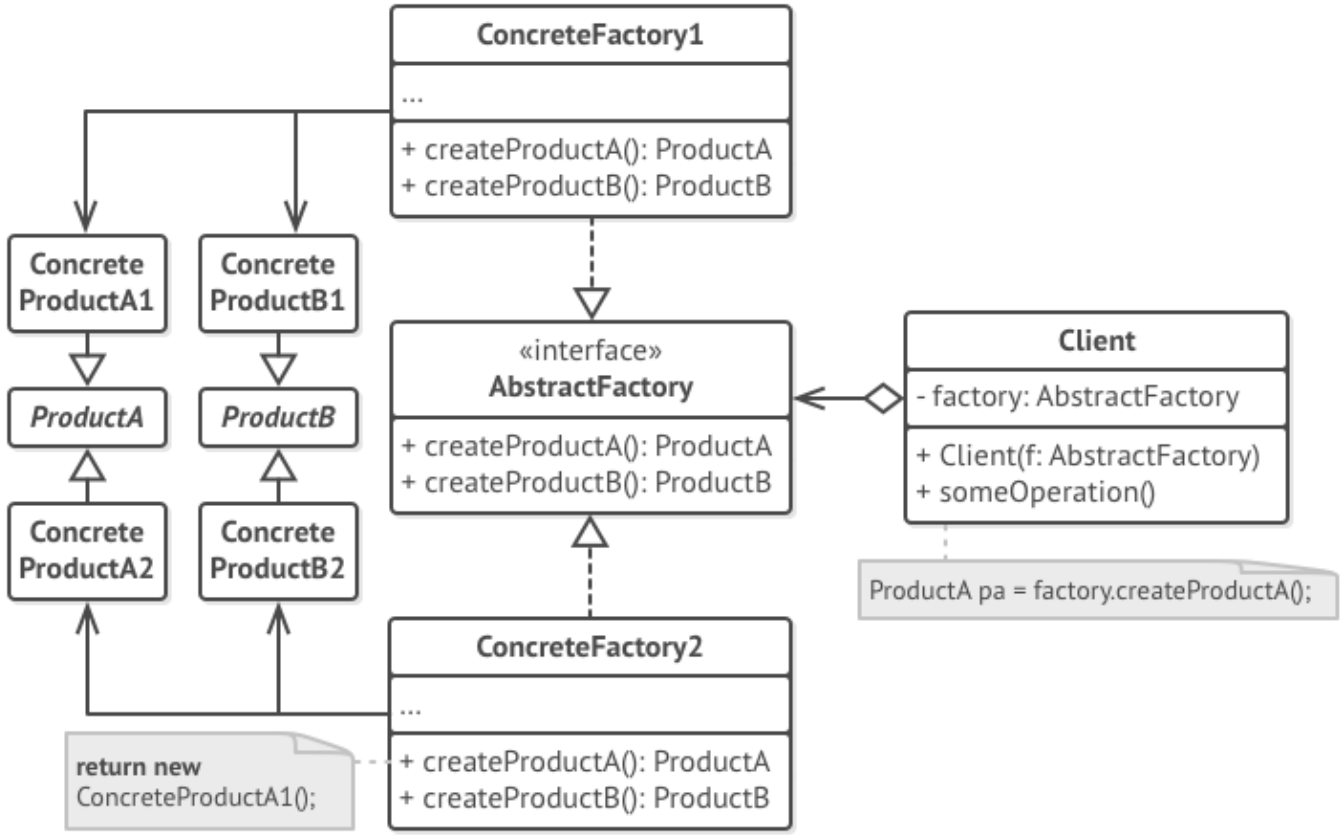
Для начала, паттерн Абстрактная фабрика предлагает выделить общие интерфейсы для отдельных продуктов, составляющих семейства. Так, все вариации кресел получат общий интерфейс Кресло, все диваны реализуют интерфейс Диван и так далее.



## Структура паттерна

1. Абстрактные продукты объявляют интерфейсы продуктов, которые связаны друг с другом по смыслу, но выполняют разные функции.
2. Конкретные продукты — большой набор классов, которые относятся к различным абстрактным продуктам (кресло/столик), но имеют одни и те же вариации (Викториан./Модерн).
3. Абстрактная фабрика объявляет методы создания различных абстрактных продуктов (кресло/столик).
4. Несмотря на то, что конкретные фабрики порождают конкретные продукты, сигнатуры их методов должны возвращать соответствующие абстрактные продукты. Это позволит клиентскому коду, использующему фабрику, не привязываться к конкретным классам продуктов. Клиент сможет работать с любыми вариациями продуктов через абстрактные интерфейсы.



# Применимость

## Применение паттерна

1. Когда бизнес-логика программы должна работать с разными видами связанных друг с другом продуктов, не завися от конкретных классов продуктов.
2. Когда в программе уже используется Фабричный метод, но очередные изменения предполагают введение новых типов продуктов.

# Шаги реализации

## Алгоритм реализации паттерна

1. Создайте таблицу соотношений типов продуктов к вариациям семейств продуктов.
2. Сведите все вариации продуктов к общим интерфейсам.
3. Определите интерфейс абстрактной фабрики. Он должен иметь фабричные методы для создания каждого из типов продуктов.
4. Создайте классы конкретных фабрик, реализовав интерфейс абстрактной фабрики. Этих классов должно быть столько же, сколько и вариаций семейств продуктов.
5. Измените код инициализации программы так, чтобы она создавала определённую фабрику и передавала её в клиентский код.
6. Замените в клиентском коде участки создания продуктов через конструктор вызовами соответствующих методов фабрики.

# Преимущества и недостатки

## Плюсы и недостатки

Плюсы:
- Гарантирует сочетаемость создаваемых продуктов.
- Избавляет клиентский код от привязки к конкретным классам продуктов.
- Выделяет код производства продуктов в одно место, упрощая поддержку кода.
- Упрощает добавление новых продуктов в программу.
- Реализует *принцип открытости/закрытости*.

Минусы:
- Усложняет код программы за счёт множества дополнительных классов.
- Требует наличия всех типов продуктов в каждой вариации.

## Отношение с другими паттернами

- Многие архитектуры начинаются с применения Фабричного метода (более простого и расширяемого через подклассы) и эволюционируют в сторону Абстрактной фабрики, Прототипа или Строителя (более гибких, но и более сложных).
- Строитель концентрируется на постройке сложных объектов шаг за шагом. Абстрактная фабрика специализируется на создании семейств связанных продуктов. *Строитель* возвращает продукт только после выполнения всех шагов, а *Абстрактная фабрика* возвращает продукт сразу же.
- Классы Абстрактной фабрики чаще всего реализуются с помощью Фабричного метода, хотя они могут быть построены и на основе Прототипа.
- Абстрактная фабрика может быть использована вместо Фасада для того, чтобы скрыть платформо-зависимые классы.
- Абстрактная фабрика может работать совместно с Мостом. Это особенно полезно, если у вас есть абстракции, которые могут работать только с некоторыми из реализаций. В этом случае фабрика будет определять типы создаваемых абстракций и реализаций.
- Абстрактная фабрика, Строитель и Прототип могут быть реализованы при помощи Одиночки.

# Информационный видеосервис для разработчиков программного обеспечения

ITVDN

IT VIDEO DEVELOPERS NETWORK

TestProvider PROMETRIC TEST CENTER CyberBionic systematics Microsoft Partner Silver Learning Windows Azure Cloud Partner EBA European Business Association