

Абстрактная фабрика (Abstract Factory)

Суть паттерна

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

Проблема

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.

Вам нужен такой способ создавать объекты продуктов, чтобы они сочетались с другими продуктами того же семейства. Это важно, так как клиенты расстраиваются, если получают несочетающуюся мебель.

Кроме того, вы не хотите вносить изменения в существующий код при добавлении новых продуктов или семейств в программу. Поставщики часто обновляют свои каталоги, и вы бы не хотели менять уже написанный код каждый раз при получении новых моделей мебели.

Решение

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

Далее вы создаёте *абстрактную фабрику* — общий интерфейс, который содержит методы создания всех продуктов семейства (например, создать Кресло, создать Диван и создать Столик). Эти операции должны возвращать **абстрактные** типы продуктов, представленные интерфейсами, которые мы выделили ранее — Кресла, Диваны и Столики.

Как насчёт вариаций продуктов? Для каждой вариации семейства продуктов мы должны создать свою собственную фабрику, реализовав абстрактный интерфейс. Фабрики создают продукты одной вариации. Например, Фабрика Модерн будет возвращать только Кресла Модерн, Диваны Модерн и Столики Модерн.

Клиентский код должен работать как с фабриками, так и с продуктами только через их общие интерфейсы. Это позволит подавать в ваши классы любой тип фабрики и производить любые продукты, ничего не ломая.

Например, клиентский код просит фабрику сделать стул. Он не знает, какого типа была эта фабрика. Он не знает, получит викторианский или современный стул. Для него важно, чтобы на стуле можно было сидеть, и чтобы этот стул отлично смотрелся с диваном той же фабрики.

Осталось прояснить последний момент: кто создаёт объекты конкретных фабрик, если клиентский код работает только с интерфейсами фабрик? Обычно программа создаёт конкретный объект фабрики при запуске, причём тип фабрики выбирается, исходя из параметров окружения или конфигурации.

Структура

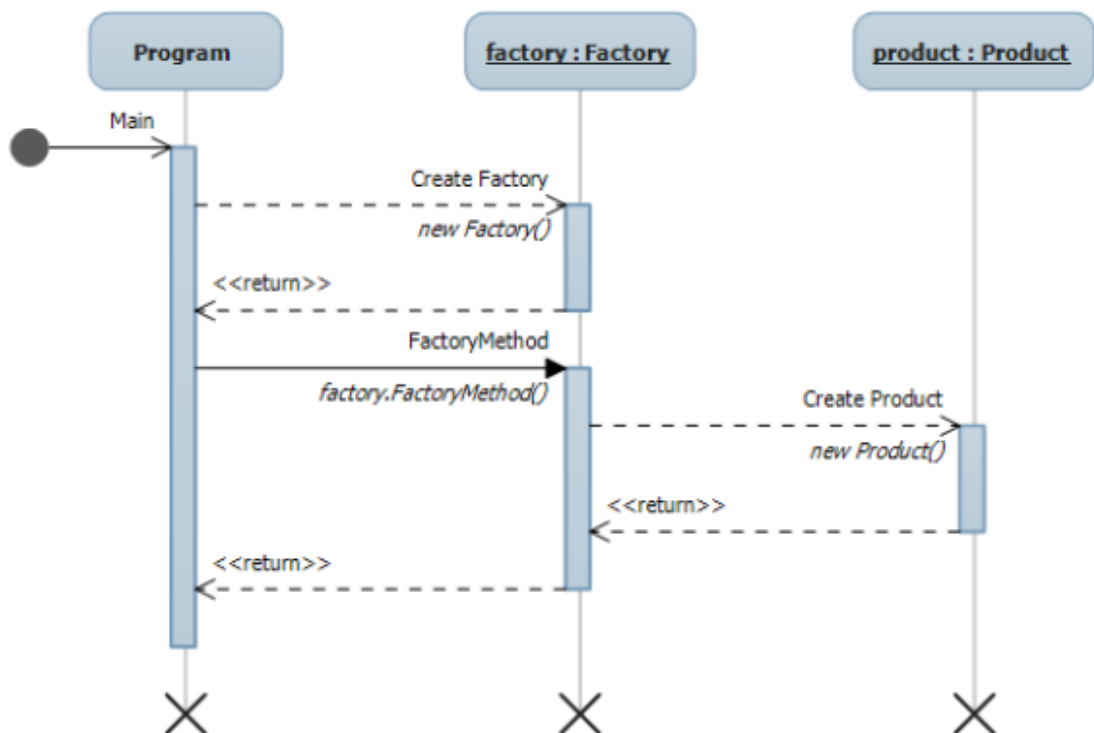
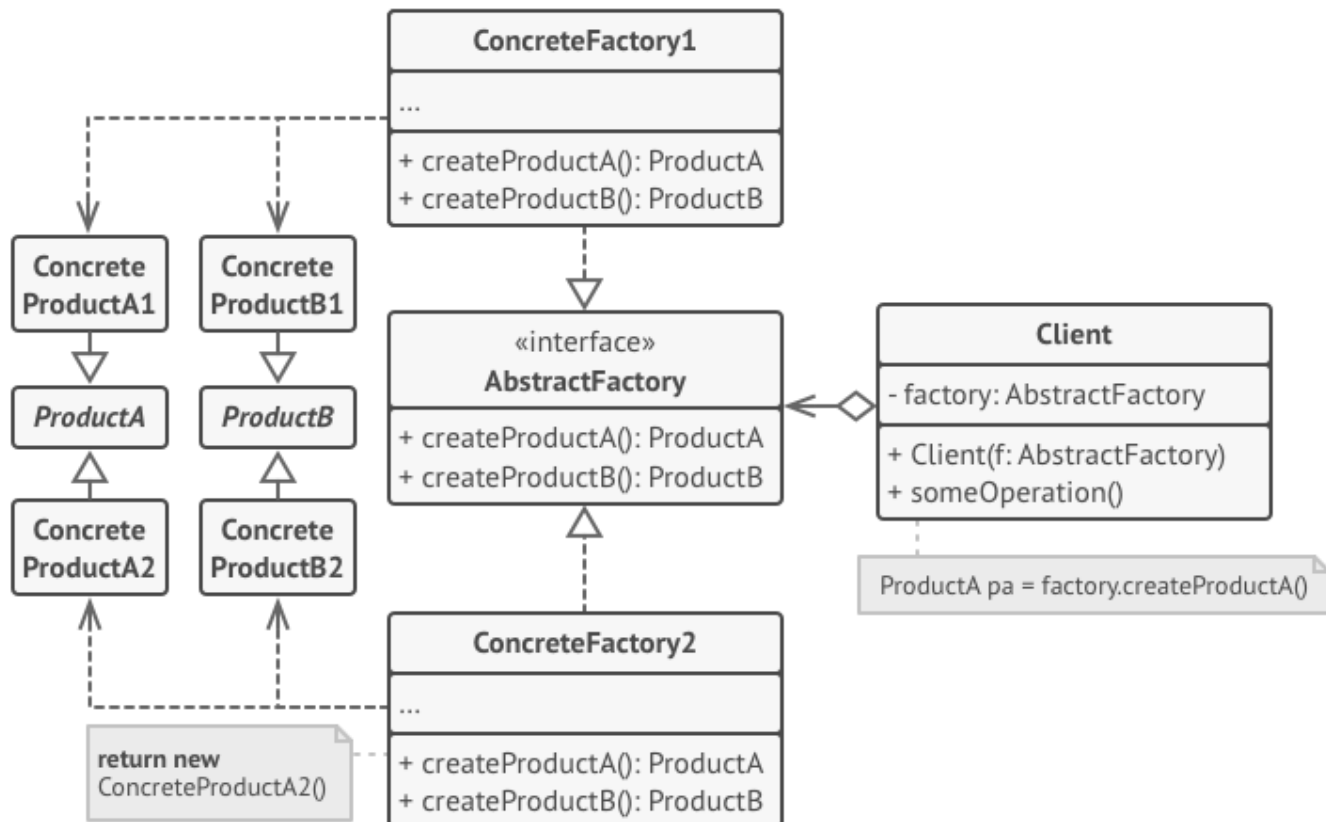
Абстрактные продукты объявляют интерфейсы продуктов, которые связаны друг с другом по смыслу, но выполняют разные функции.

Конкретные продукты — большой набор классов, которые относятся к различным абстрактным продуктам (кресло/столик), но имеют одни и те же вариации (Викторианский/Модерн).

Абстрактная фабрика объявляет методы создания различных абстрактных продуктов (кресло/столик).

Конкретные фабрики относятся каждая к своей вариации продуктов (Викторианский/Модерн) и реализуют методы абстрактной фабрики, позволяя создавать все продукты определённой вариации.

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.



Преимущества:

- 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*.

Недостатки:

- Усложняет код программы из-за введения множества дополнительных классов.
- Требуется наличие всех типов продуктов в каждой вариации.