Java Design Patterns

Builder

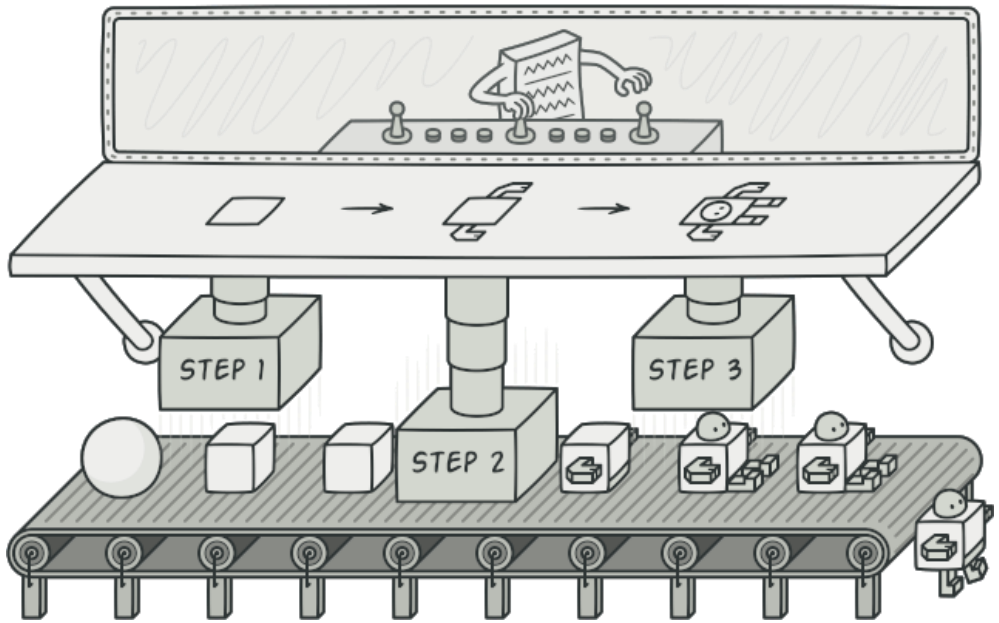
Тема

Builder

Суть паттерна

Строитель

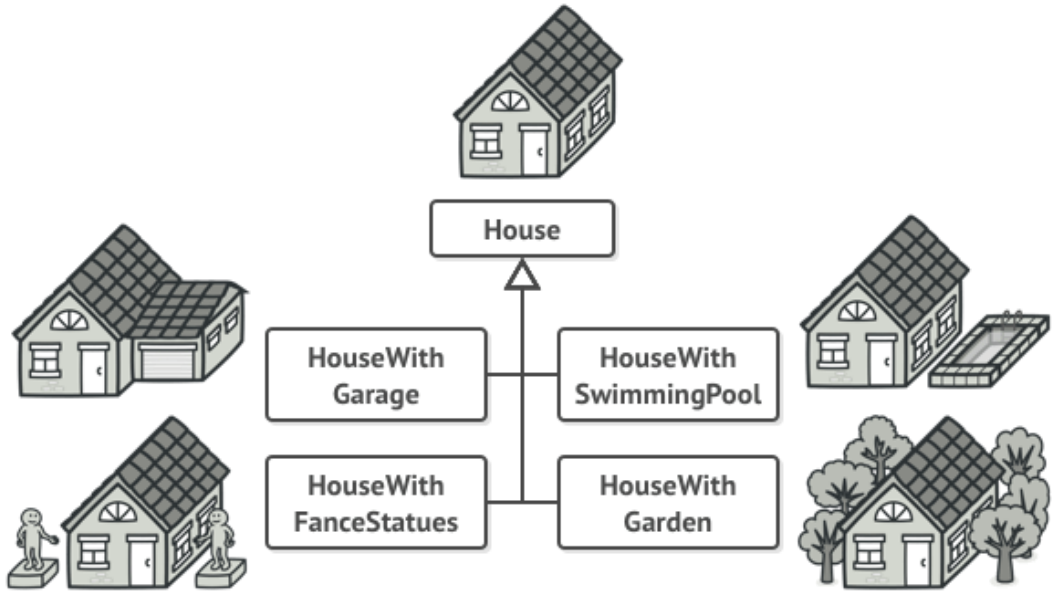
Строитель — это порождающий паттерн проектирования, который позволяет создавать сложные объекты пошагово.

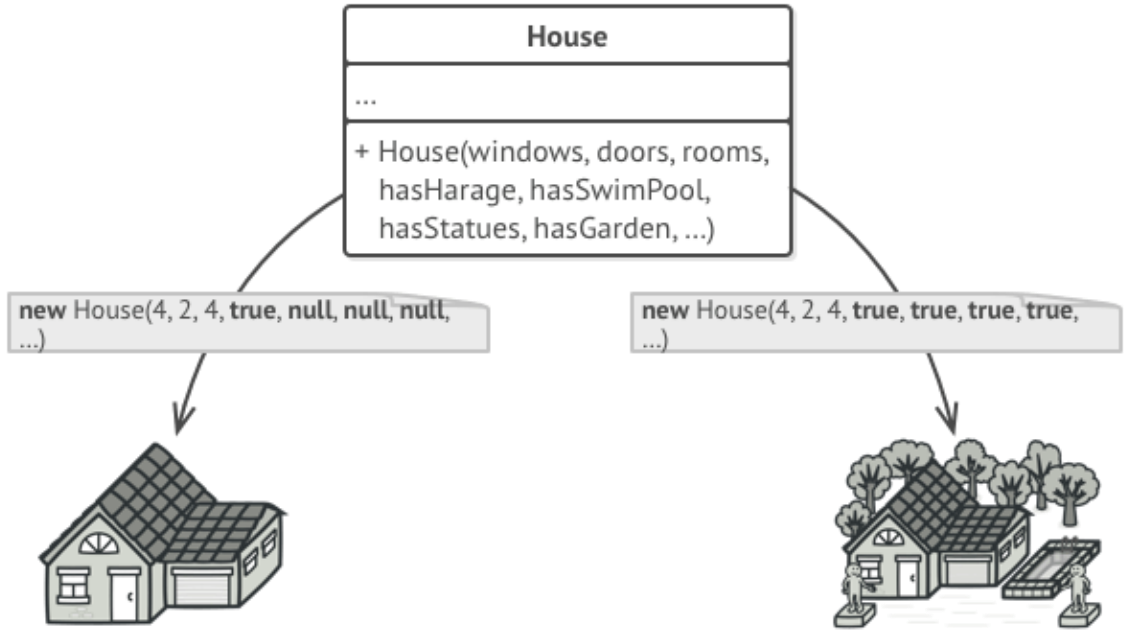


Проблема

Постановка задачи

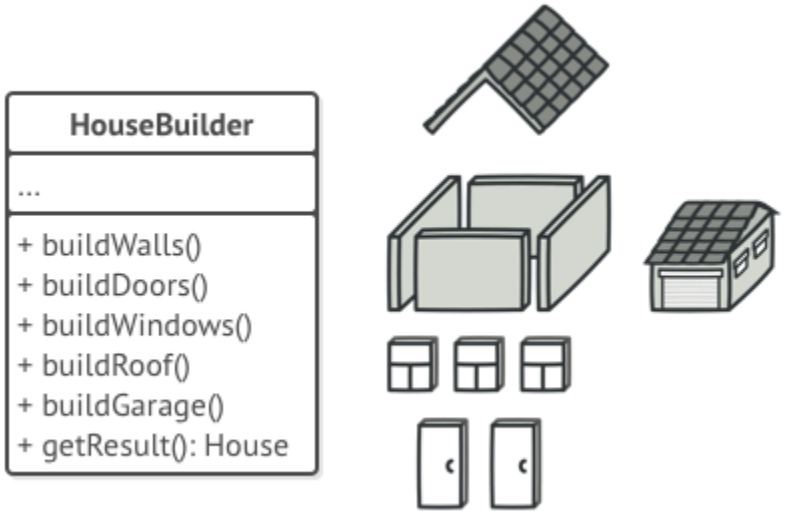
Представьте сложный объект, требующий кропотливой пошаговой инициализации множества полей и вложенных объектов. Код инициализации обычно спрятан внутри монструозного конструктора с десятком параметров, либо ещё хуже — распылён по всему клиентскому коду.





Решение задачи

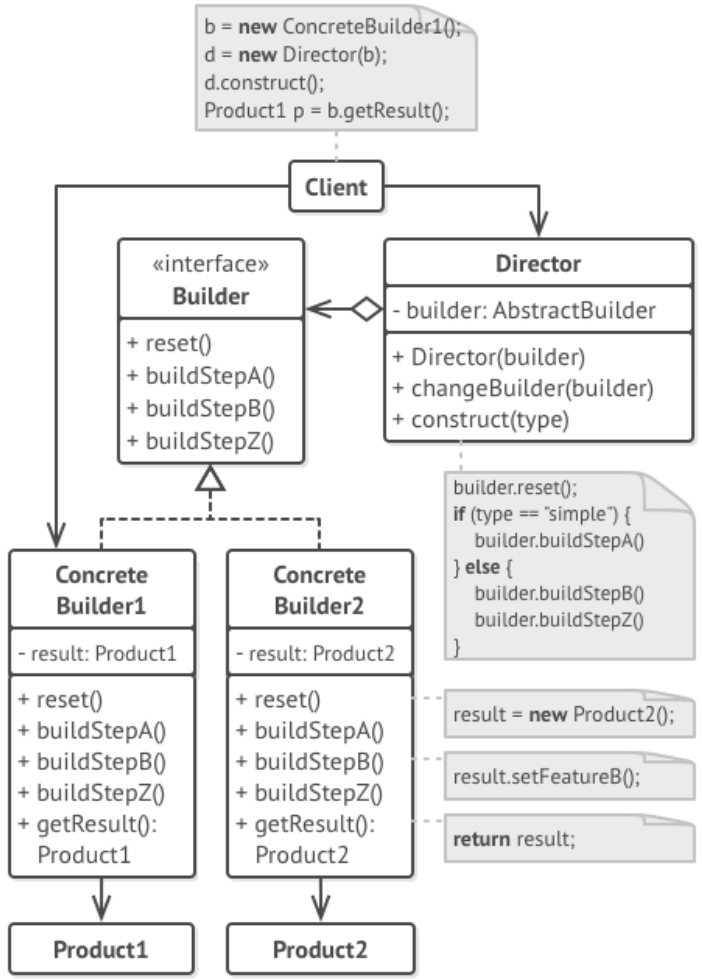
Паттерн Строитель предлагает вынести конструирование объекта за пределы его собственного класса, поручив это дело отдельным объектам, называемым *строителями*.



Структура

Структура паттерна

1. Интерфейс строителя объявляет шаги конструирования продуктов, общие для всех видов строителей.
2. Конкретные строители реализуют строительные шаги, каждый по-своему. Конкретные строители могут производить разнородные объекты, не имеющие общего интерфейса.
3. Продукт — создаваемый объект. Продукты, сделанные разными строителями, не обязаны иметь общий интерфейс.
4. Директор определяет порядок вызова строительных шагов для производства той или иной конфигурации объектов.
5. Обычно, Клиент подаёт в конструктор директора уже готовый объект-строитель, и в дальнейшем данный директор использует только его. Но возможен и другой вариант, когда клиент передаёт строителя через параметр строительного метода директора. В этом случае можно каждый раз применять разных строителей для производства различных представлений объектов.



Применимость

Применение паттерна

1. Когда вы хотите избавиться от «телескопического конструктора».
2. Когда ваш код должен создавать разные представления какого-то объекта. Например, деревянные и железобетонные дома.
3. Когда вам нужно собирать сложные составные объекты, например, деревья Компоновщика.

Шаги реализации

Алгоритм реализации паттерна

1. Убедитесь в том, что создание разных представлений объекта можно свести к общим шагам.
2. Опишите эти шаги в общем интерфейсе строителей.
3. Для каждого из представлений объекта-продукта создайте по одному классу-строителю и реализуйте их методы строительства.
4. Подумайте о создании класса директора. Его методы будут создавать различные конфигурации продуктов, вызывая разные шаги одного и того же строителя.
5. Клиентский код должен будет создавать и объекты строителей, и объект директора. Перед началом строительства, клиент должен связать определённого строителя с директором. Это можно сделать либо через конструктор, либо через сеттер, либо подав строителя напрямую в строительный метод директора.
6. Результат строительства можно вернуть из директора, но только если метод возврата продукта удалось поместить в общий интерфейс строителей. Иначе, вы жёстко привяжете директора к конкретным классам строителей.

Преимущества и недостатки

Плюсы и недостатки

Плюсы:

- Позволяет создавать продукты пошагово.
- Позволяет использовать один и тот же код для создания различных продуктов.
- Изолирует сложный код сборки продукта от его основной бизнес-логики.

Минусы:

- Усложняет код программы за счёт дополнительных классов.
- Клиент будет привязан к конкретным классам строителей, так как в интерфейсе строителя может не быть метода получения результата.

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.
• Строитель позволяет пошагово сооружать дерево Компоновщика.
• Паттерн Строитель может быть построен в виде Моста: *директор* будет играть роль абстракции, а *строители* — реализации.
• 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