Factory Method

Um padrão criacional que fornece uma interface para criar objetos em uma superclasse, mas permite que subclasses alterem o tipo de objetos criados.

Quando usar o Factory Method?

Tipos desconhecidos

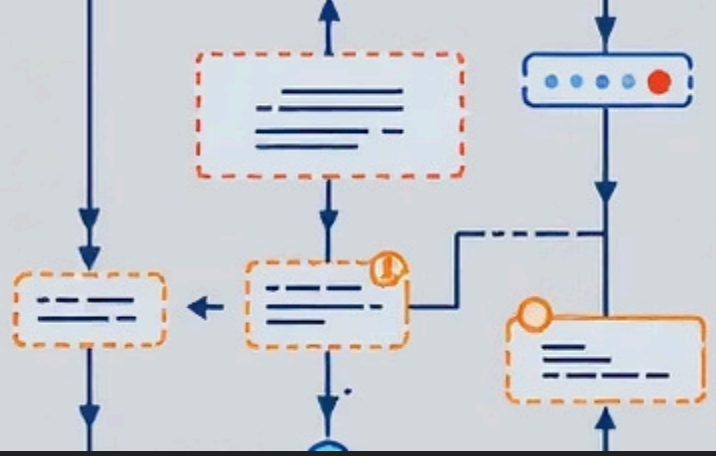
Quando você não sabe de antemão os tipos exatos e dependências dos objetos que seu código deve trabalhar. O padrão separa a construção do produto do código que o utiliza.

Extensibilidade

Para fornecer aos usuários da sua biblioteca uma forma de estender componentes internos. A herança permite estender comportamentos padrão facilmente através de factory methods.

Reutilização de recursos

Quando você quer economizar recursos do sistema reutilizando objetos existentes em vez de reconstruí-los. Ideal para objetos pesados como conexões de banco de dados.



Vantagens

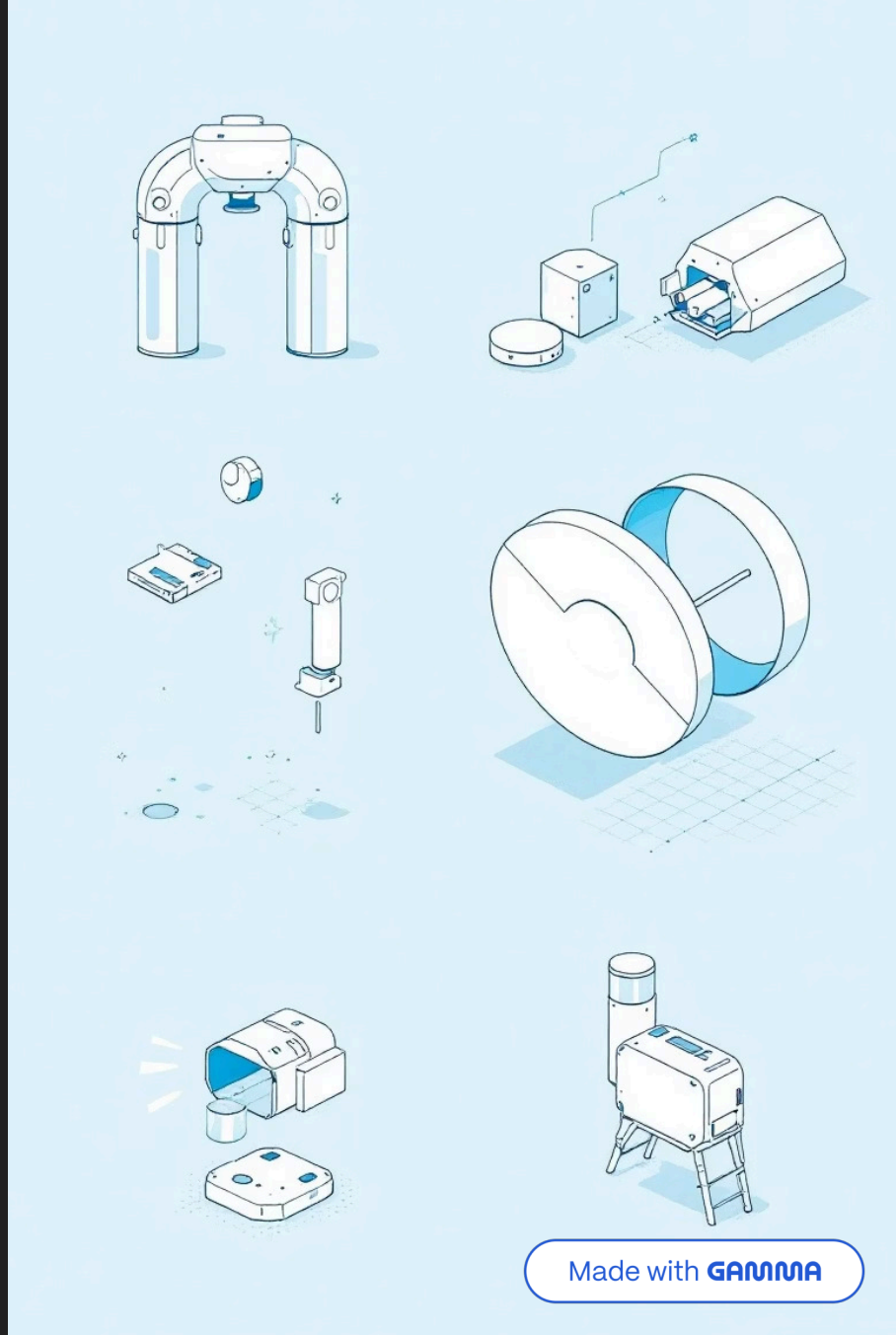
- Evita acoplamento forte entre criador e produtos concretos
- **Princípio da Responsabilidade Única:** código de criação centralizado e fácil de manter
- **Princípio Aberto/Fechado:** novos produtos sem quebrar código existente

Desvantagens

- O código pode se tornar mais complicado com muitas subclasses
- Requer hierarquia de classes bem planejada

Builder

Padrão criacional que permite construir objetos complexos passo a passo. O padrão permite produzir diferentes tipos e representações de um objeto usando o mesmo código de construção.



Quando usar o Builder?

Objetos Complexos

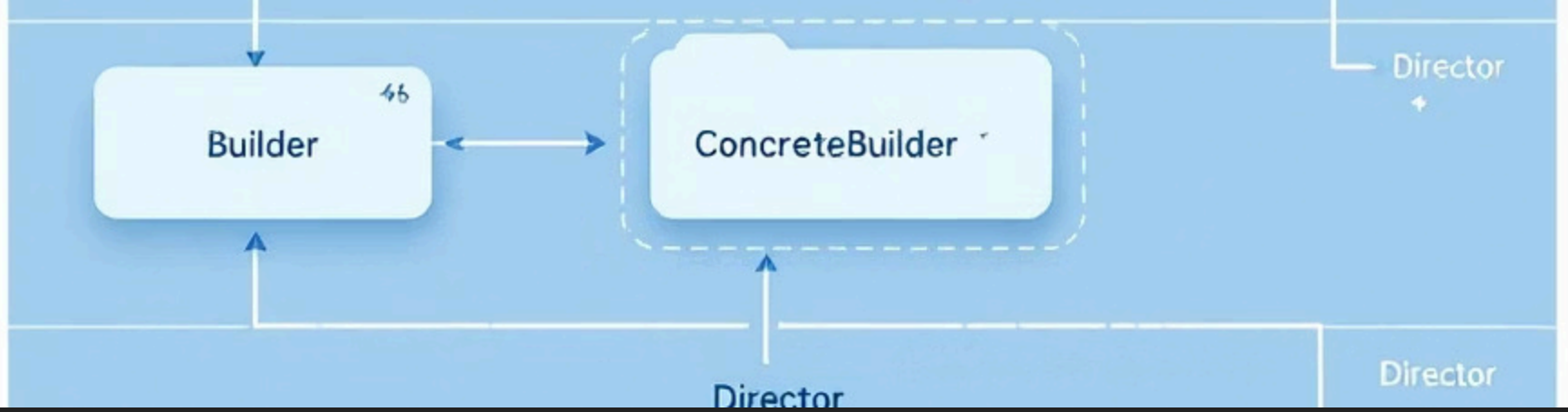
Quando você precisa construir objetos com muitos parâmetros opcionais. Evita construtores gigantes com dezenas de parâmetros (telescoping constructor).

Múltiplas Representações

Para criar diferentes representações do mesmo produto usando o mesmo código de construção. Cada builder implementa os passos de forma diferente.

Estruturas Aninhadas

Ideal para construir árvores de objetos ou estruturas compostas complexas. Permite construir passo a passo e até chamar passos recursivamente.



Vantagens e Desvantagens do Builder

Vantagens

- Constrói objetos passo a passo, permitindo adiar ou executar passos recursivamente
- Reutiliza o mesmo código de construção para diferentes representações
- **Princípio da Responsabilidade Única:** código de construção isolado da lógica do produto
- Evita construtores gigantes com muitos parâmetros

Desvantagens

- Aumenta a complexidade do código com muitas classes builders
- Requer mais linhas de código comparado a construtores simples
- Overhead para objetos simples que não precisam de construção complexa