Curso: Sistemas de Informação	
Disciplina: Padrões de projeto	**Período:** 8
Professor: Luiz Gustavo Dias	**Tipo:** Design Patterns
Objetivo: Builder	

DESIGN PATTERNS: BUILDER

O Builder é um padrão de projeto criacional que permite a você construir objetos complexos passo a passo. Ele permite que você produza diferentes tipos e representações de um objeto usando o mesmo código de construção.

Intenção

Separar a construção de um objeto complexo da sua representação de modo que o mesmo processo de construção possa criar diferentes representações.

Este padrão pode ser utilizado para:

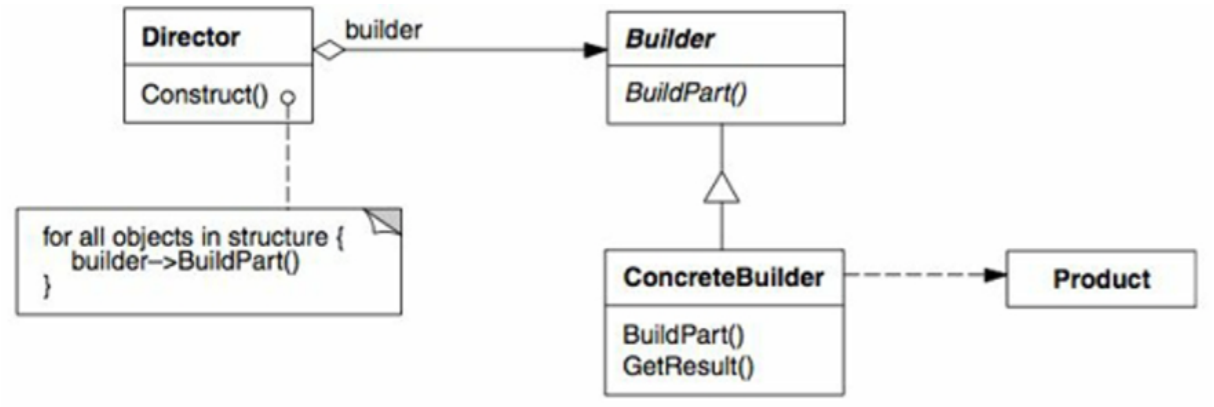
- Separar o código que cria e o código que usa o objeto, deixando o código cliente bem mais limpo;
- Trata da criação de objetos complexos:
  - Construtores muito complexos e extensos;
  - Quando há a composição de vários objetos;
  - Quando necessário executar algum algoritmo de criação do objeto complexo.
- Permite a criação de um objeto em etapas. Exemplo:
  - Pessoa:
    - Etapa 1: crie nome;
    - Etapa 2: crie endereço.
- O objeto final pode variar. Exemplo:
  - Pessoa apenas com nome e sobrenome;
  - Pessoa apenas com nome e endereço.
- É um padrão complexo.

Aplicabilidade

Use o padrão Builder quando o algoritmo para criação de um objeto complexo deve ser independente das partes que compõem o objeto e de como elas são montadas.

Além disso, ele pode ser aplicado quando o processo de construção deve permitir diferentes representações para o objeto que é construído.

Estrutura



Builder: especifica uma interface abstrata para criação de partes de um objeto produto.

ConcreteBuilder: constrói e monta partes do produto pela implementação da interface de Builder.

Director: constrói um objeto usando a interface de Builder. Essa classe é opcional, será necessária caso precise definir uma ordenação das etapas de construção.

Product: representa o objeto complexo em construção.

Consequências

Positivas:

1. Separa criação da utilização;
2. O cliente não precisa criar objetos diretamente – código desacoplado;
3. O mesmo código pode construir objetos diferentes;
4. Ajuda na aplicação dos princípios Single Responsability Principle (SRP) e Open Closed Principle (OCP).

Negativas:

1. O código final pode se tornar muito complexo.