Builder 💻

Análise e Desenvolvimento de Sistemas - 6º Período

Cenário

- Separa a contrução de objetos complexos de sua representação
- O processo de criação fica separado da sua representação
- Usamos quando queremos que a construção de um objeto permita diferentes representações
- As diferentes representações são criadas usando o mesmo código de construção

Cenário

- Imagine que estamos criando um script que faça o **provisionamento de uma Compute Instance**
- Podemos ter um **objeto simples "Compute"**, que representa um recurso computacional
- No entanto, essa compute pode ser muito personalizada, adicionando a quantidade de RAM, qual processador, GPU etc

Cenário

- Uma maneira de fazer isso é **estender a classe** do objeto simples criando um conjunto de subclasses
- Mas já sabemos o problema disso: **Explosão de classes**

Cenário

- Outra forma seria usar um **construtor na classe base com todos os parâmetros possíveis**
- No entanto, é possível que a **maioria dos objetos não utilizem esses parâmetros**
- Pode ser que algumas Compute Instances não utilizem GPU ou Subnet, por exemplo

Builder

- O padrão **Builder sugere a separação da construção de um objeto de sua própria classe**
- Dessa maneira, vamos ter um objeto "contrutor (NÃO É O MÉTODO)" que será responsável por construir esses objetos
- Portanto, vamos ter o objeto que representa e os objetos construtores

Builder

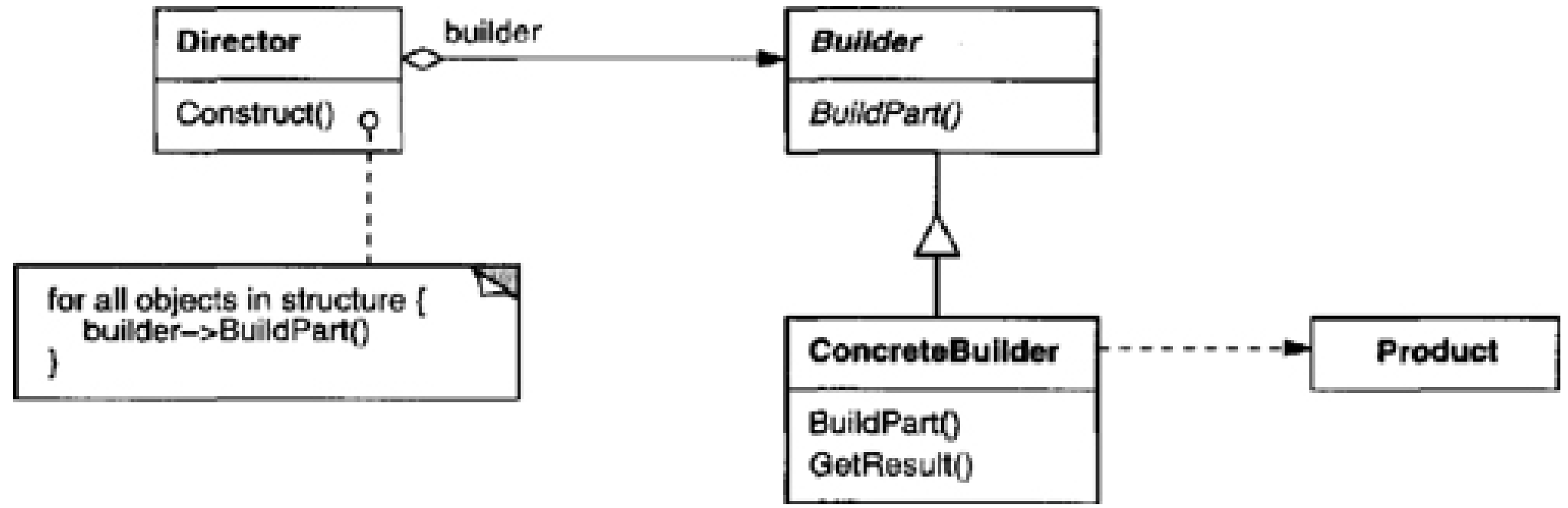
- A construção dos objetos com o Builder é **feita em etapas**. Executamos várias etapas do objeto Builder
- Podemos chamar apenas as etapas que nos interessa, permitindo uma **construção personalizada**

Builder

- Podemos criar Builders que executam as mesmas etapas de construção mas produzem objetos diferentes
- Um Builder pode ser "especialista" em construir Compute Insances simples. Outro pode ser especialista em contruir Computes para treinar Machine Learning
- Eles vão ter as mesmas etapas: Selecionar quantidade de RAM, processamento etc
- Mas o objeto será diferente

Builder

- Um elemento comum no Builder é o **Diretor**
- É uma classe que será responsável por usar as etapas de construção dos objetos Builder e **definir uma ordem correta**.
- Não é obrigatório o uso do Diretor


Director
builder
Builder
Construct()
BuildPart()
for all objects in structure {
builder->BuildPart()
}
ConcreteBuilder
Product
BuildPart()
GetResult()