Padrões de criação

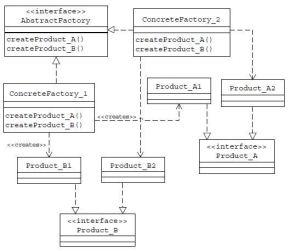
- **Abstract Factory:**
  - O padrão Abstract Factory leva o conceito de Method Factory para o próximo nível.
  - Em termos simples, uma fábrica abstrata é uma classe que fornece uma interface para produzir uma família de objetos. Na linguagem de programação Java, ele pode ser implementado como uma interface ou como uma classe abstrata.
  - No contexto de Abstract Factory existem:
    - Famílias de classes relacionadas, dependentes.
    - Um grupo de classes Concrete Factory que implementa a interface fornecida pela classe Abstract Factory.
    - Cada uma dessas fábricas controla ou fornece acesso a um conjunto particular de objetos relacionados dependentes e implementa a interface de Abstract Factory de uma maneira específica para a família de classes que controla.

Padrões de criação

- **Abstract Factory**:
  - Por exemplo, quando acessamos um banco de dados via JDBC utilizamos um conjunto de classes que representam processos do SGBD como: Connection, Statement, ResultSet, entre outros.
  - Porém estas classes são abstratas, precisam de implementações. Cada SGBD terá sua implementação para esta família de classes;
  - Ou seja, cada SGBD poderá ter a sua Concrete Factory, que implementa a Abstract Factory;

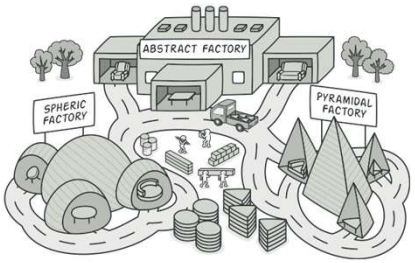
Padrões de criação

- **Abstract Factory**:



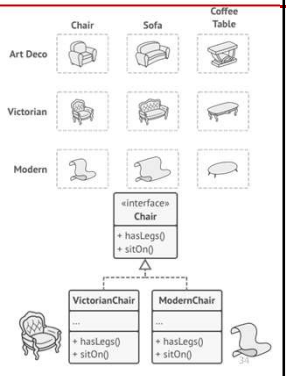
Padrões de criação

- **Abstract Factory**:



Padrões de criação

- **Abstract Factory**:

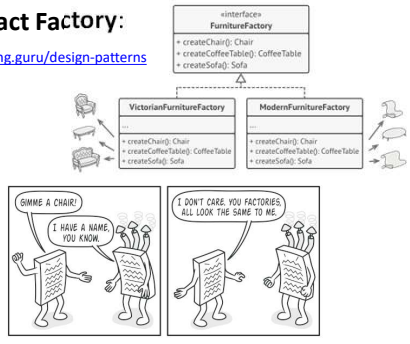


Solução:

Padrões de criação

- **Abstract Factory**:

https://refactoring.guru/design-patterns/abstract-factory



Padrões de criação

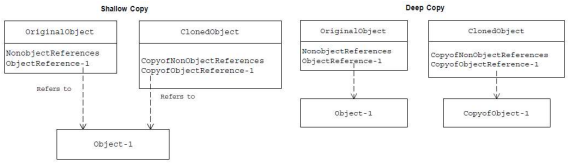
- **Prototype**:
  - Quando um cliente precisa criar um conjunto de objetos que são iguais ou diferem uns dos outros apenas em termos de seu estado e é caro criar tais objetos em termos do tempo e do processamento envolvido.
  - Como alternativa à construção de inúmeras fábricas que espelham as classes a serem instanciadas (como em Factory Method).
  - O padrão Prototype sugere:
    - Criar um objeto antecipadamente e designe-o como um objeto "protótipo".
    - Crie outros objetos simplesmente fazendo uma cópia do objeto protótipo e fazendo as modificações necessárias.

Padrões de criação

- **Prototype**:
  - No mundo real, usamos o padrão Prototype em muitas ocasiões para reduzir o tempo e o esforço gastos em tarefas diferentes. Dois exemplos são os seguintes:
    1. **Criação de Novo Programa de Software** - Normalmente os programadores tendem a fazer uma cópia de um programa existente com estrutura semelhante e modificá-lo para criar novos programas.
    2. **Cartas de Apresentação** - Ao se candidatar a cargos em diferentes organizações, um candidato não pode criar cartas de apresentação para cada organização individualmente a partir do zero. Em vez disso, o requerente iria criar uma carta de apresentação no formato mais atraente, fazer uma cópia dele e personalizá-lo para cada organização.

Padrões de criação

- **Prototype**:
  - Cópia rasa vs. Cópia profunda:



  - No Java o padrão é cópia rasa. A cópia profunda pode ser implementada sobrescrevendo-se o método clone().

Padrões de criação

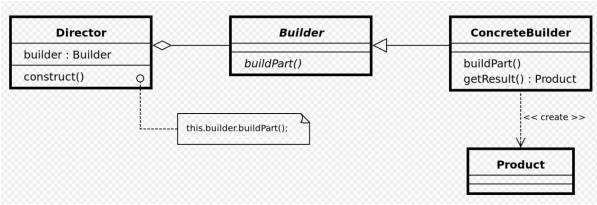
- **Builder**:
  - Em geral, detalhes de construção de objeto são mantidos dentro do objeto, muitas vezes como parte de seu construtor. Esse tipo de projeto vincula o processo de construção com os componentes que o compõem.
    - Esta abordagem é adequada enquanto o objeto em construção é simples e o processo de construção do objeto é definido e sempre produz a mesma representação do objeto.
  - Este design pode não ser eficaz quando o objeto que está sendo criado é complexo.
  - Como diferentes implementações do processo de construção são todas mantidas dentro do objeto, o objeto pode se tornar volumoso e pouco modular.

Padrões de criação

- **Builder:**
  - *Objetivo: Separar o processo de construção de um objeto de sua representação e permitir a sua criação passo-a-passo. Diferentes tipos de objetos podem ser criados com implementações distintas de cada passo.*
  - Largamente utilizado para construção de interfaces gráficas;
    - Um painel principal, que é composto por outros painéis, composto por botões, imagens, inputs, etc.
    - A criação ainda pode depender de preferências do usuário...

Padrões de criação

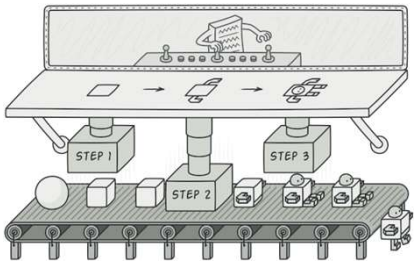
- **Builder**:



Padrões de criação

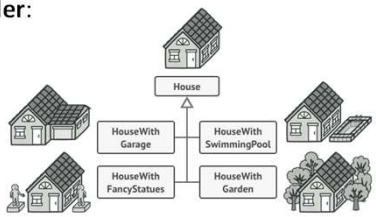
- **Builder**:

Dividir a criação de um objeto complexo em várias etapas.



Padrões de criação

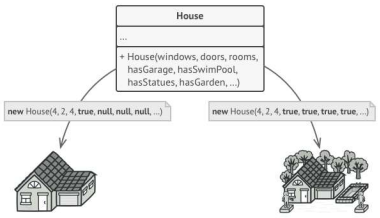
- **Builder**:



Talvez criar uma enorme hierarquia de classes não seja a melhor solução.

Padrões de criação

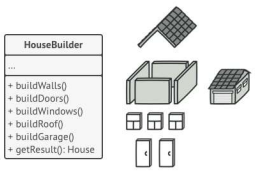
- **Builder**:



O construtor com muitos parâmetros tem sua desvantagem: nem todos os parâmetros são necessários o tempo todo.

Padrões de criação

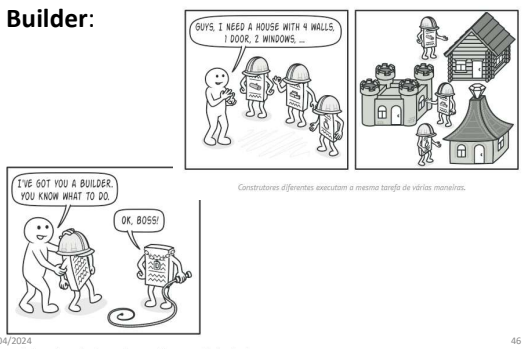
- **Builder**:



O padrão Builder permite construir objetos complexos passo a passo. O Builder não permite que outros objetos acessem o produto enquanto ele está sendo construído.

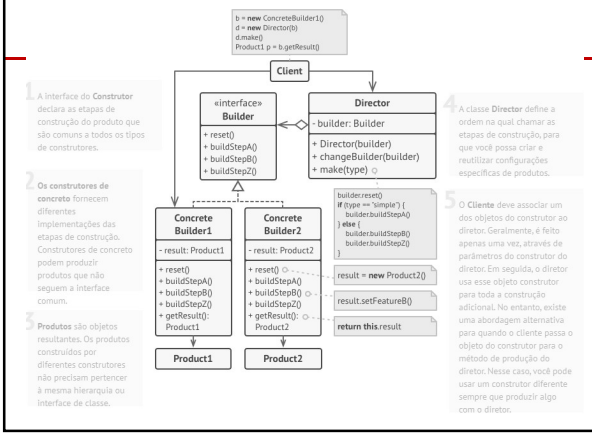
Padrões de criação

- **Builder**:



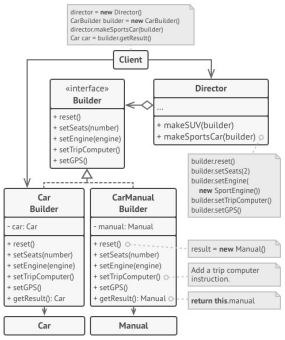
Construtores diferentes executam a mesma tarefa de várias maneiras.

O diretor sabe quais etapas de construção executar para obter um produto funcional.



Padrões de criação

- **Builder**:



O exemplo da construção passo a passo dos carros e as guias do usuário que se encaixam nesses modelos de carros.

Padrões de criação

- Resumo:
  - **FactoryMethod:** Encapsular a escolha da classe concreta a ser utilizada na criação de objetos de um determinado tipo.
  - **Abstract Factory:** Encapsular a escolha das classes concretas a serem utilizadas na criação dos objetos de diversas famílias.
  - **Builder:** Separar o processo de construção de um objeto de sua representação e permitir a sua criação passo-a-passo. Diferentes tipos de objetos podem ser criados com implementações distintas de cada passo.
  - **Prototype:** Possibilitar a criação de novos objetos a partir da cópia de objetos existentes.
  - **Singleton:** Permitir a criação de uma única instância de uma classe e fornecer um modo para recuperá-la.
  - **Multiton** Permitir a criação de uma quantidade limitada de instâncias de determinada classe e fornecer um modo para recuperá-las.
  - **Object Pool** Possibilitar o reaproveitamento de objetos.

Padrões de criação

- Exercício 2:
  d. Defina uma interface Prototype usando genéricos (class: Prototype<T>, method: T clone()). Crie uma classe ContaBancaria que implementa Prototype e uma classe para teste.
  e. Implemente um Builder para fazer pizzas, com métodos para acrescentar diferentes ingredientes permitindo fazer muitos sabores.
    - Construa também uma classe diretora que utiliza um builder para construir os diferentes sabores.
    - O Builder deve implementar um interface, o que permitiria fazer uma implementação do mesmo builder também para calzones e construir calzones com os mesmos sabores das pizzas reaproveitando o diretor.