# Abstract Factory

O padrão Abstract Factory é o primeiro Pattern descrito no livro Design Patterns do **GoF**. Ela faz parte da categoria de **Patterns Criacionais**, cujo objetivo é a instanciação de objetos. Essa categoria é importante pois ela sustenta o princípio mais importante do livro: **"programe para interfaces e não para implementações"**.

Atualmente, os **Patterns Criacionais** estão em desuso, sendo substituídos pelos frameworks de **Injeção de Dependência**, que fazem exatamente isso: instanciam para você as classes das quais você é dependente. De toda forma, conhecer os patterns, a sua motivação e entender as suas consequências é um bom exercício de design de software.

https://medium.com/@gbbigardi/arquitetura-e-desenvolvimento-de-software-parte-2-abstract-factory-f603ccc6a1ea

# Use Cases - Factory Method

---

1. Quando há uma série de objetos a serem criados e que seguem uma mesma interface, podemos usar o **factory method.** Por exemplo, *cadeira*, *sofa*, *banquinho*. Todos eles possuem o método *canSit()*.

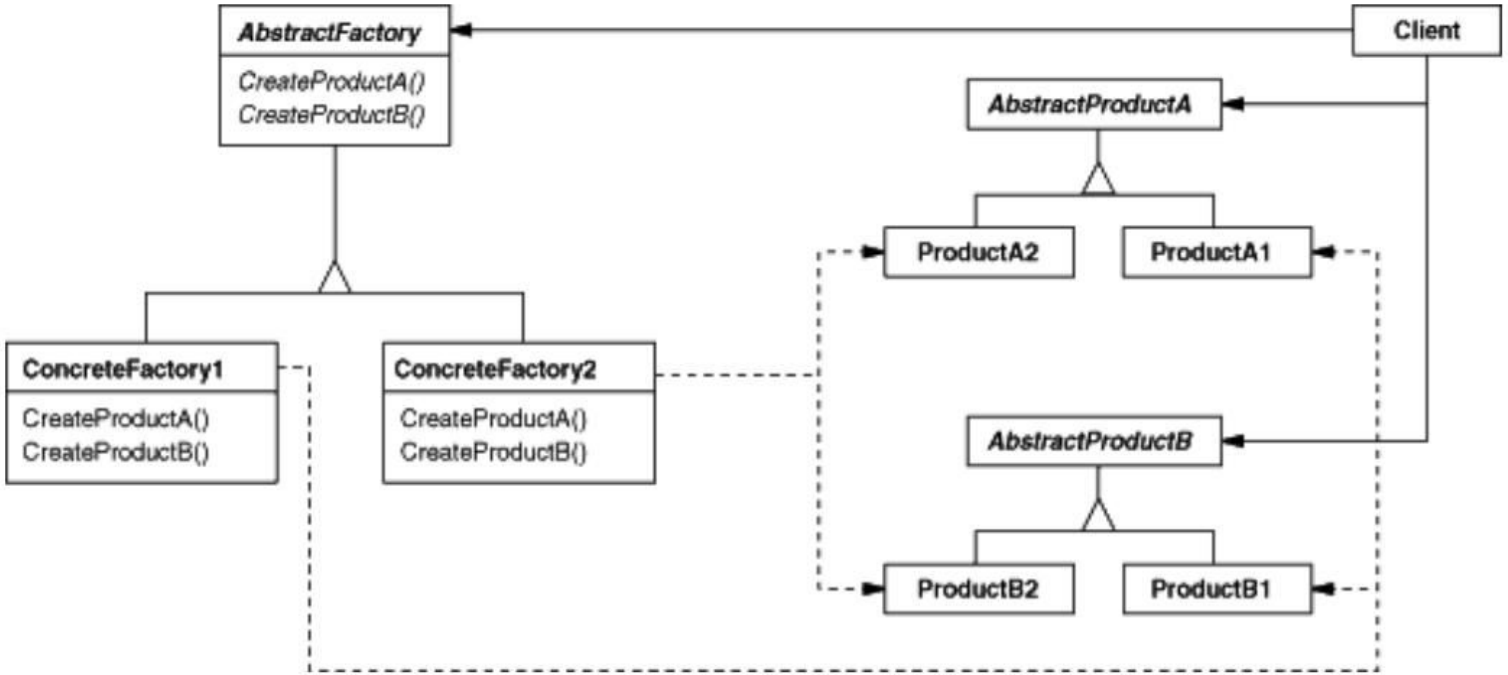
   Com o **factory method**, você irá criar esses objetos a partir de métodos dentro de sua classe.

# Use Cases - Abstract Factory

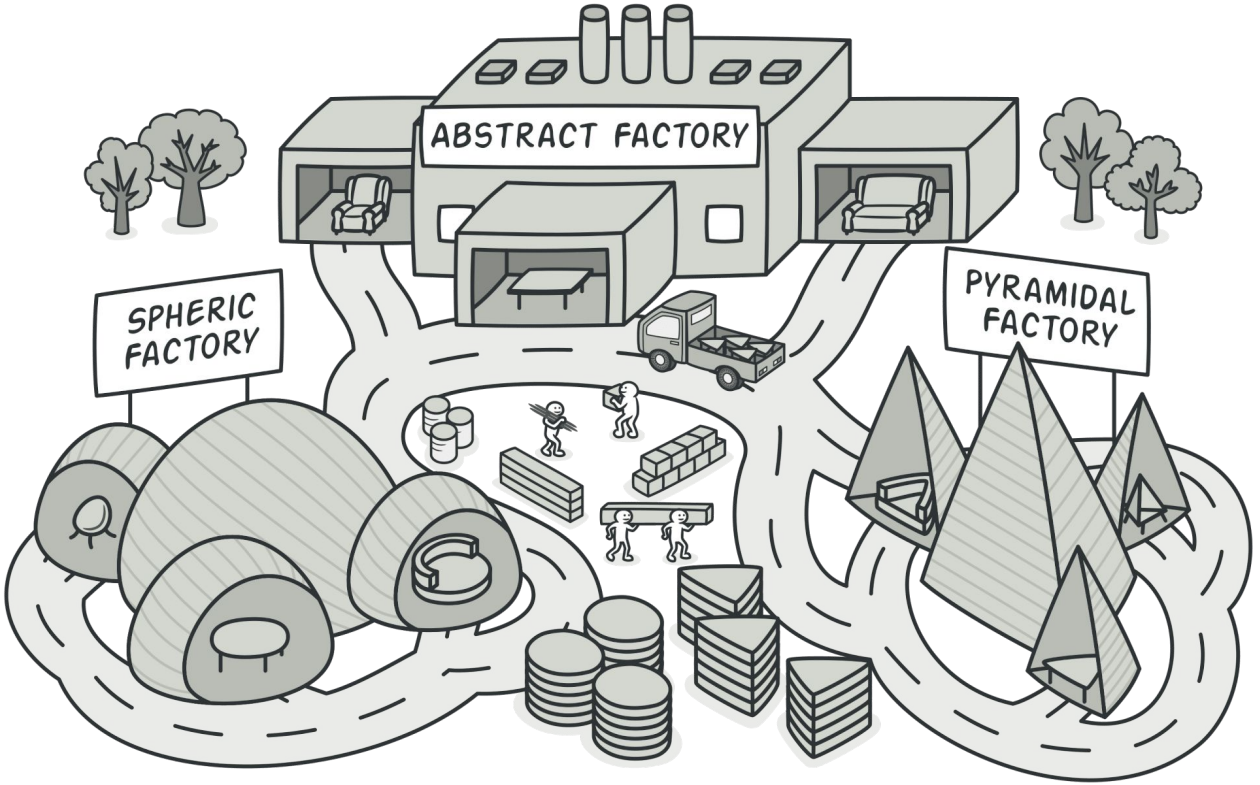
---

1. Mas e se essa série de objetos a serem criados possuem cada uma, uma variante?
2. E se um sofá puder ser um "Sofá Moderno" e um "Sofá Vintage"? E cada um dos móveis puder ter essas variações?
3. Agora, nesse caso, pode ser interessante fazer uso de uma *abstract factory*.
4. Uma abstract factory nada mais é do que uma *interface* para várias *factories* relacionadas.

# Use Cases - Abstract Factory



# Use Cases - Abstract Factory



# Use Cases - Abstract Factory

**1.** Crie **interfaces padrões** para os diferentes produtos dessa família (como Window, ScrollBar e Menu). E todo o seu sistema vai trabalhar **apenas** com essas **interfaces** que você definiu.

**2.** Defina uma **Abstract Factory**, que tem os métodos de instanciação para cada uma dessas interfaces padrões definidas acima, no caso, os métodos para o exemplo seriam: **CreateWindow, CreateScrollBar, CreateMenu**. Sempre que o **sistema precisar de instâncias** dos produtos ele irá conseguir as mesmas através dessa **Abstract Factory**.

# Use Cases - Abstract Factory

Mas de onde virá essa Factory? Em algum lugar, provavelmente na **inicialização** do seu sistema, você define qual **Abstract Factory** você vai fornecer ao sistema, e repassa essa **Factory** a todos os objetos que dependem dela. Veja o esquema:

# Vantagens - Abstract Factory

---

1. Uma família de produtos e variantes pode ser instanciado e definidos em *runtime*, sem dependência de suas classes concretas.
2. Facilita a inserção de novos objetos relacionados / derivados
3. Todos os produtos criados pelas factories são compatíveis entre eles.

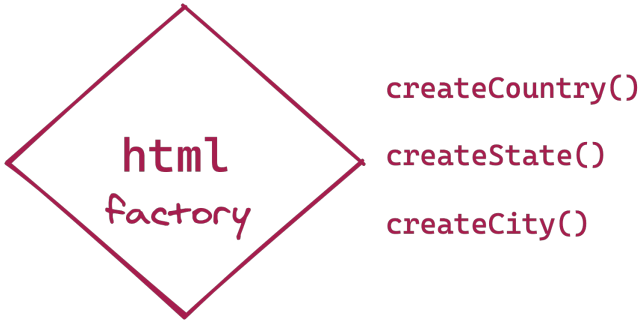
**Pontos Negativos**

1. Muito trabalho, muito código.

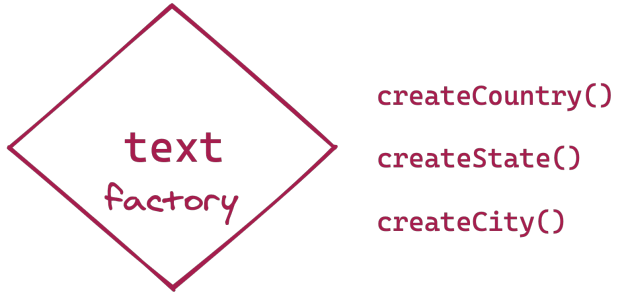
# Vamos Codar!

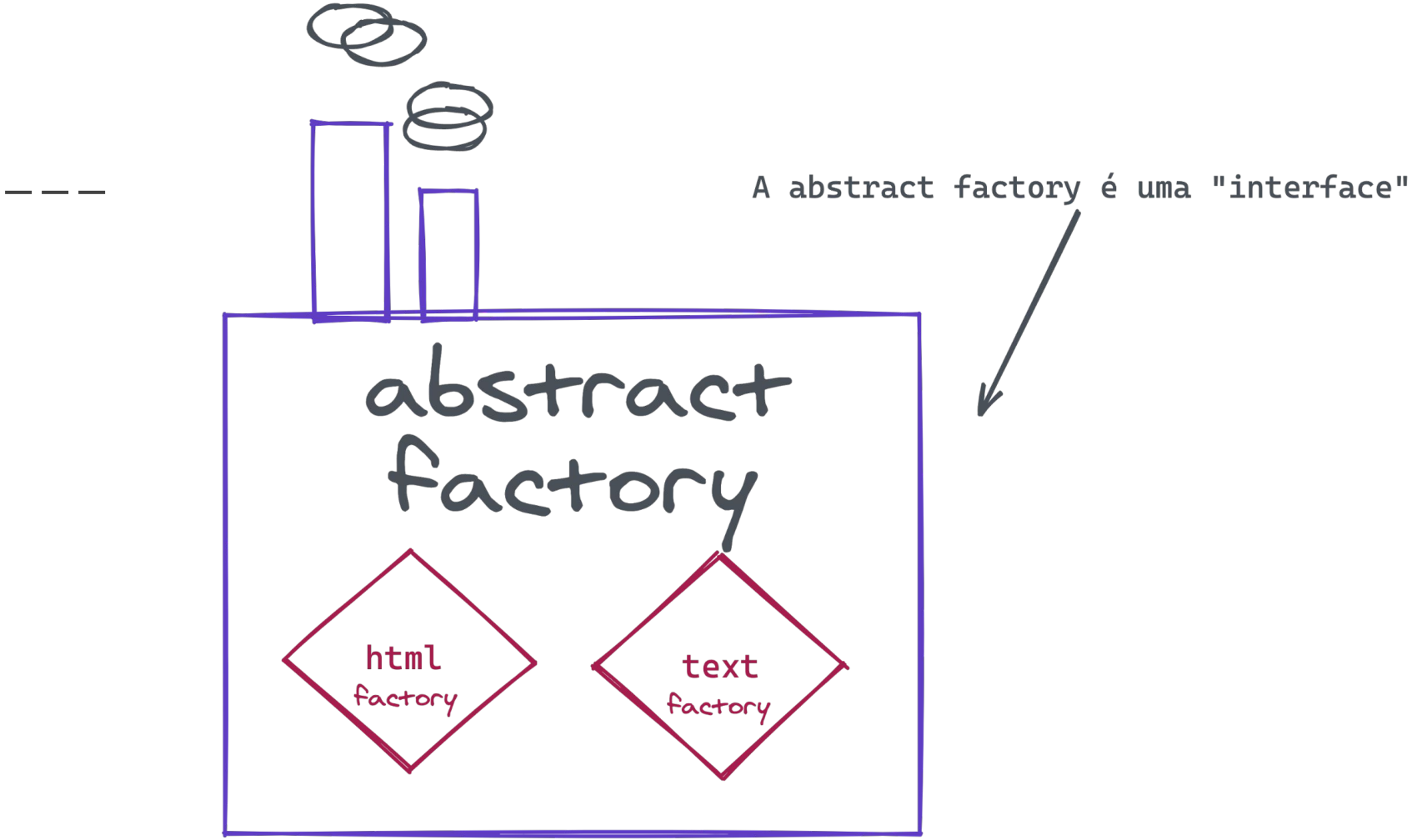
# Exemplo: Wikipedia Geográfica

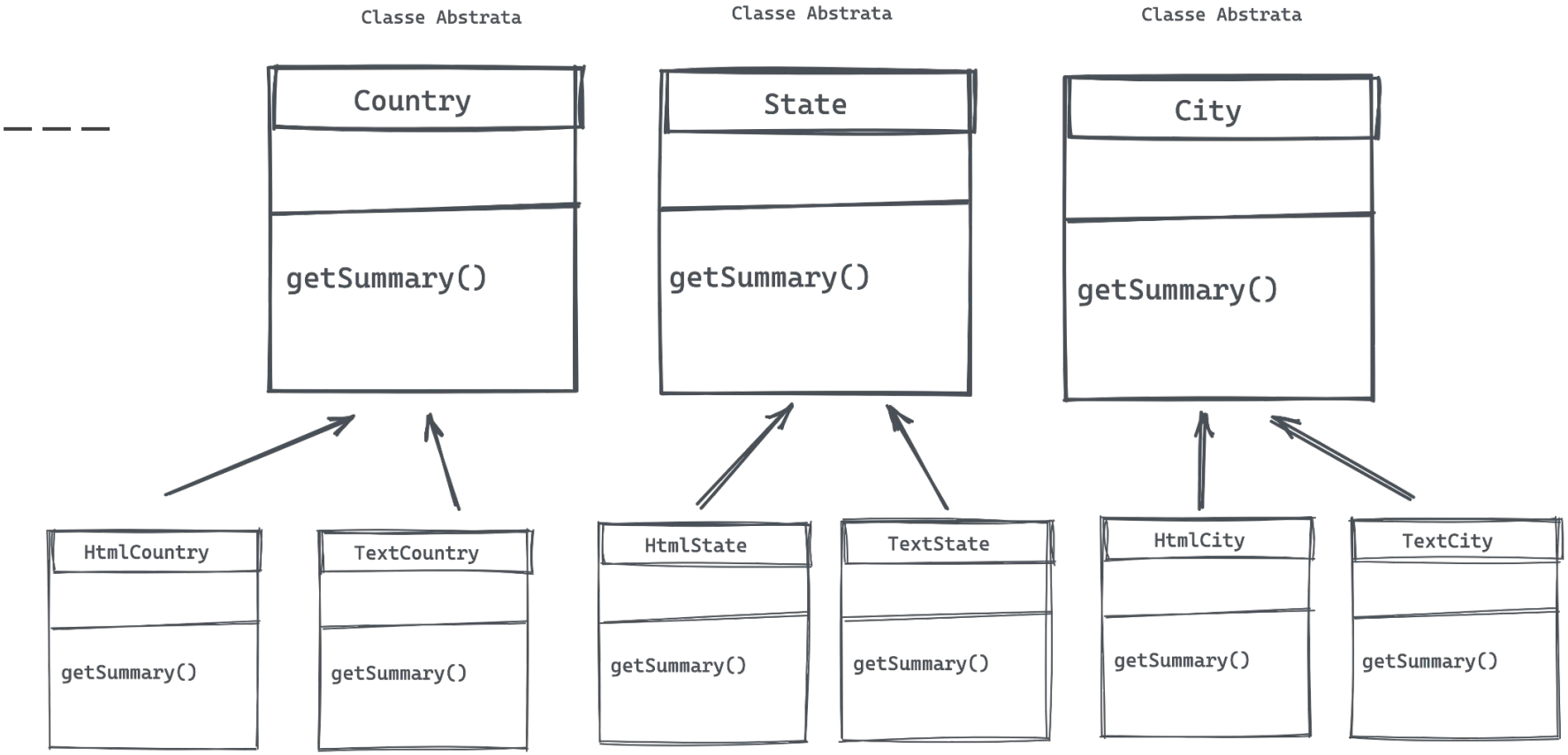
Factory para Widgets Html




Factory para widgets texto




A abstract factory é uma "interface"
abstract
factory
html
factory
text
factory


Classe Abstrata
Classe Abstrata
Classe Abstrata
Country
getSummary()
State
getSummary()
City
getSummary()
HtmlCountry
getSummary()
TextCountry
getSummary()
HtmlState
getSummary()
TextState
getSummary()
HtmlCity
getSummary()
TextCity
getSummary()