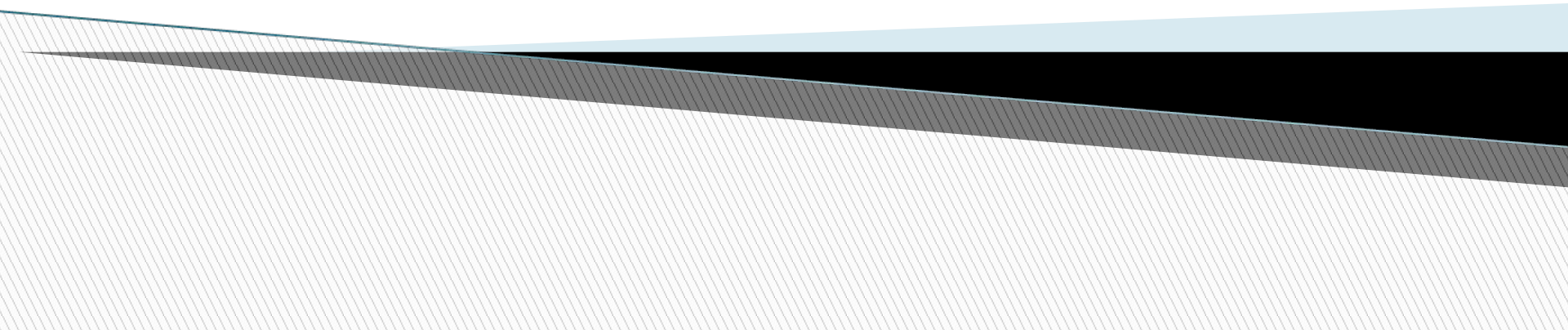


# PADRÕES DE CRIAÇÃO

ABSTRACT FACTORY



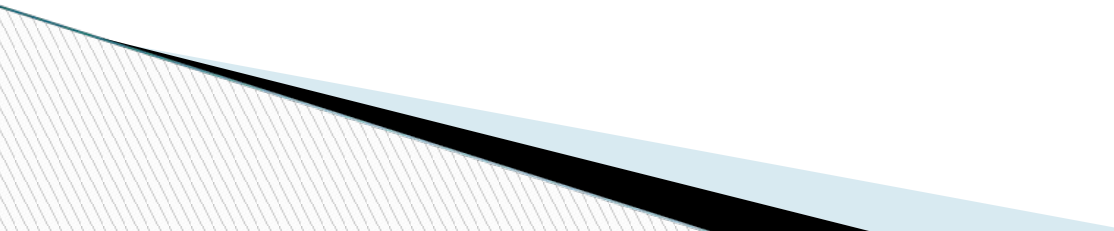


# INTENÇÃO

- Fornecer uma interface para criar famílias de objetos relacionados ou dependentes sem especificar suas classes concretas.



# APLICABILIDADE

- ❑ um sistema deve ser independente da forma como os seus produtos são criados, composto, e representados.
  - ❑ um sistema deve ser configurado com uma das várias famílias de produtos.
  - ❑ uma família de objetos de produtos relacionados é projetado para ser usado em conjunto, e você precisa para fazer cumprir essa restrição.
  - ❑ você deseja fornecer uma biblioteca de classes de produtos, e que pretende revelar apenas as suas interfaces, não suas implementações.
- 

# PARTICIPANTES

## ▣ **AbstractFactory**

- declara uma interface para as operações que criam objetos abstratos produtos.

## ▣ **ConcreteFactory**

- implementa as operações para criar objetos de produtos concretos.

## ▣ **Abstract**

- declara uma interface para um tipo de objeto produto.

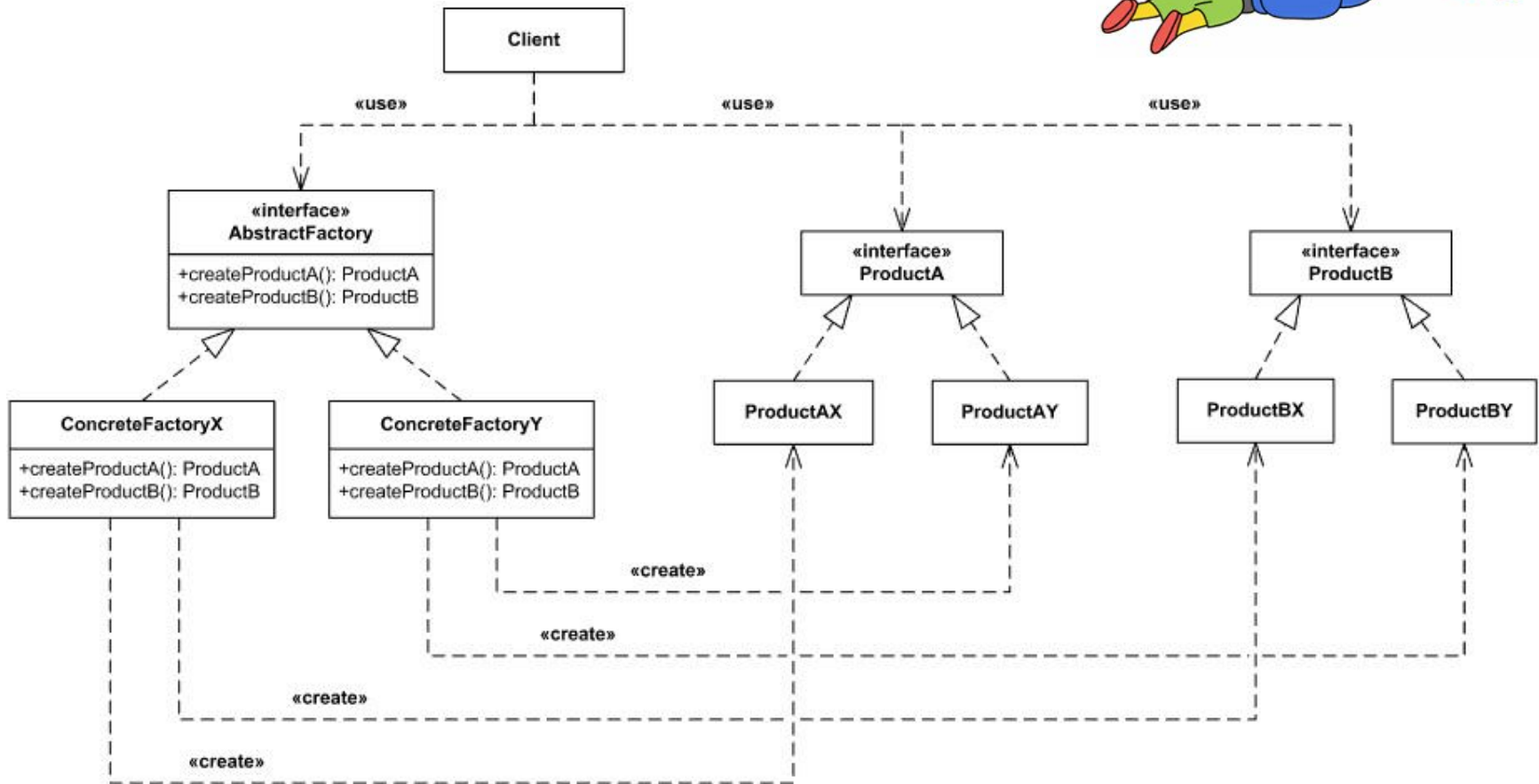
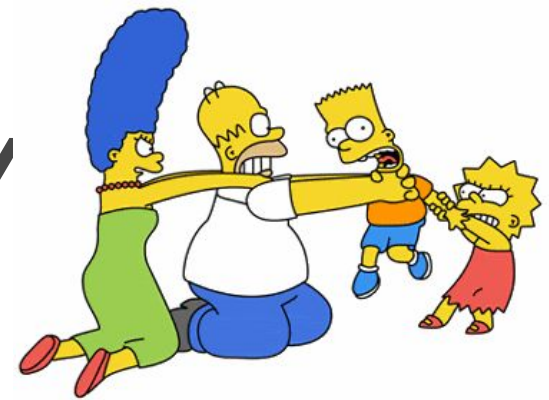
## ▣ **ConcreteProduct**

- define um objeto de produto para ser criado pela fábrica correspondente.
- implementa a interface.

## ▣ **Cliente**

- usa apenas interfaces declaradas por classes AbstractFactory e Produto.

# ABSTRACT FACTORY

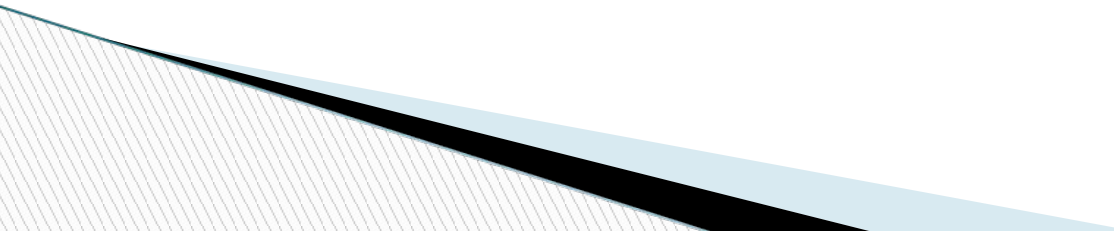


# CONSEQUÊNCIAS

- ▣ *Ela isola classes concretas.*
- ▣ *Faz fácil troca de famílias de produtos*
- ▣ *Ela promove a consistência entre os produtos*
- ▣ *Apoiar novos tipos de produtos é difícil.*



# ABSTRACT FACTORY

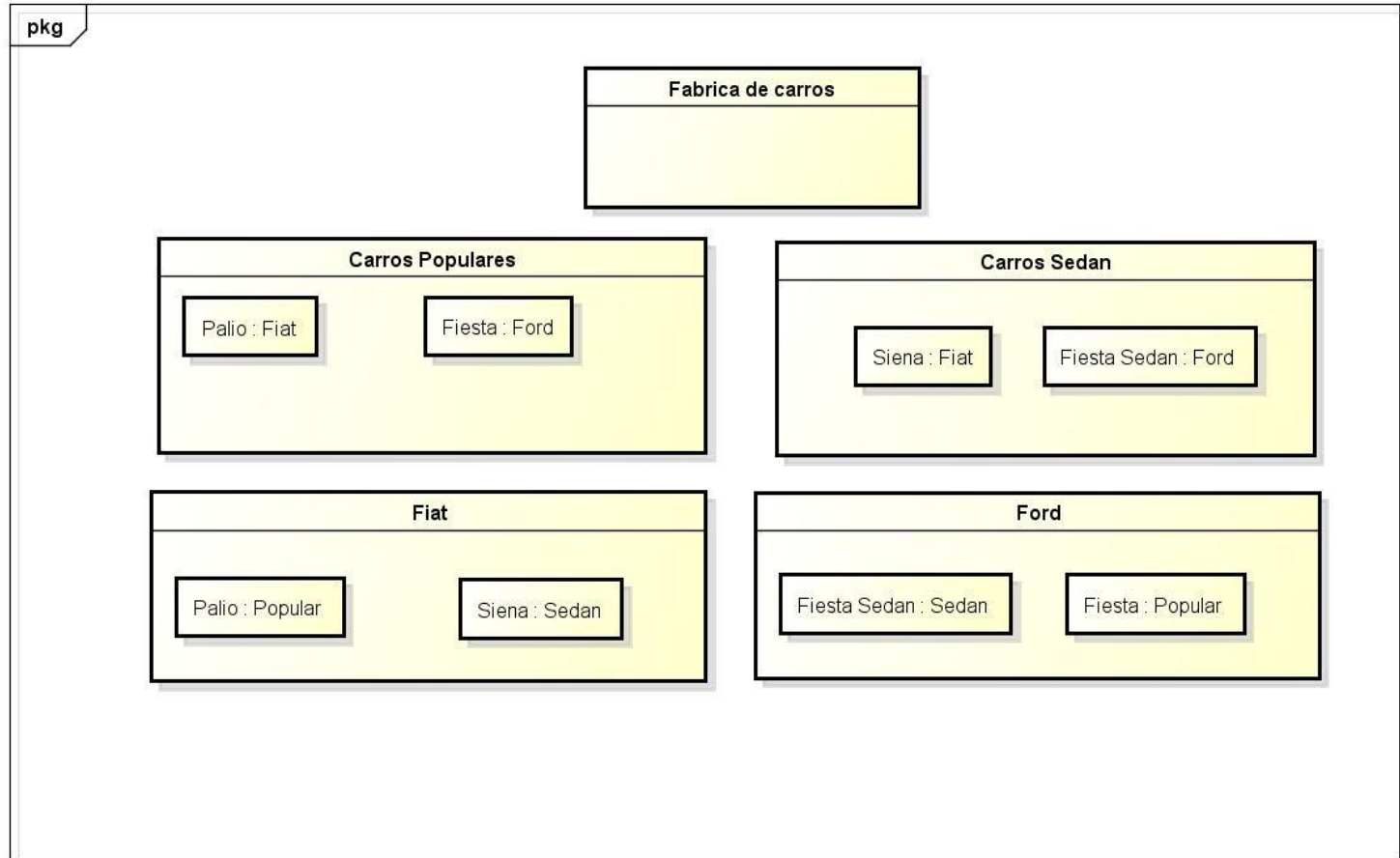
- Este padrão permite a criação de famílias de objetos relacionados ou dependentes, através de uma única interface e sem que a classe concreta seja especificada.
  - Um exemplo bem simplista seria um projeto com interface para Mobile e para Desktop, uma boa opção para reaproveitar os mesmos controles de interface seria criar pacotes com classes abstratas e os pacotes com as classes concretas implementando apenas as diferenças.
  - Esse padrão também se aplica na padronização de ambientes, por exemplo, tamanhos de botões, fontes, cores de fundo, largura de bordas. Com isso e havendo uma política que exija que os desenvolvedores usem essas classes em vez das nativas da linguagem, ajudará a padronizar a aparência e comportamento das aplicações.
- 



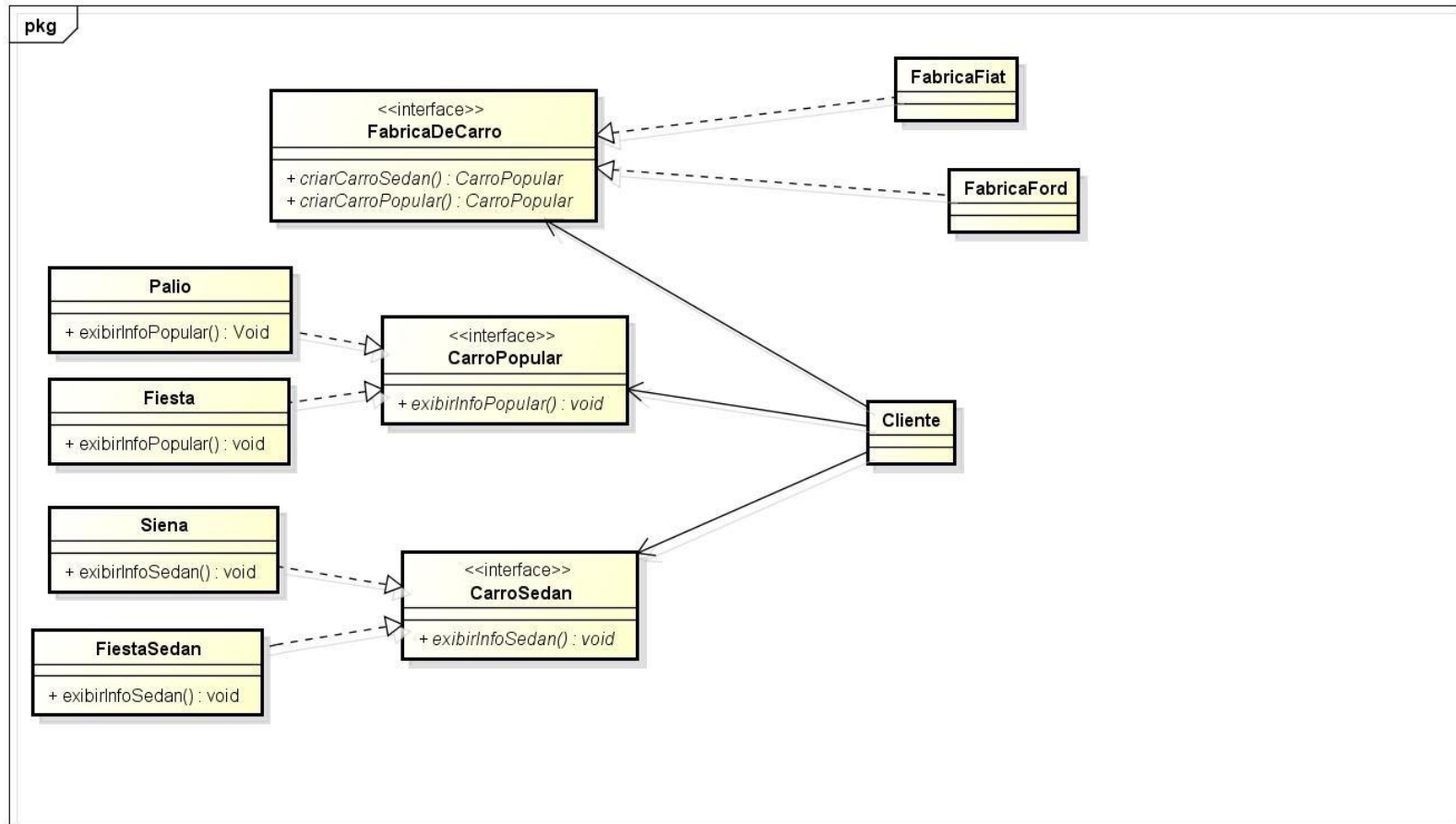
# ABSTRACT FACTORY

- Representar um sistema que, dado um conjunto de carros deve manipulá-los. A diferença é que, desta vez, precisamos agrupar os carros em conjuntos. A ideia de conjuntos é agrupar objetos que tem comportamentos parecidos.
- Como por exemplo em dois conjuntos : populares e sedan.
- Um conjunto de carros populares e outro para sedan.
- Poderíamos criar uma classe produto para cada novo carro e uma classe fábrica para cada novo carro, dessa forma teríamos que ter uma fábrica específica para cada fabricante, por exemplo uma fábrica para populares da Fiat e outro para populares da Ford.
- “Fornecer uma interface para criação de famílias de objetos relacionados ou dependentes sem especificar suas classes concretas.”
- Dessa forma construiríamos famílias de carros Sedan e famílias de carros Populares, sendo assim precisaremos de uma fábrica para cada tipo de carro.

# ABSTRACT FACTORY



# ABSTRACT FACTORY



# VIDA DE PROGRAMADOR

.COM.BR

/" HISTÓRIA REAL  
ENVIADA POR  
JOÃO NEVIADUNSKI "/



#378

VOCÊ DISSE QUE  
PROGRAMA EM  
JAVA, NÉ?

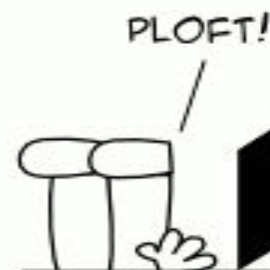


AH, SIM...  
EU PROGRAMO EM  
JAVA HÁ 17 ANOS!



DEPOIS...

CAÇAROLA! EU  
NUNCA VI ISSO ANTES!  
ESTOU TENTANDO COMPILAR  
AQUI E TÁ PEDINDO UM TAL  
DE JDK. VOCÊ TEM AÍ?



PLOFT!