Abstract Factory

Instituto Federal do Espirito Santo – Campus Colatina
Disciplina: Projeto de Sistemas
Professor: Giovany Frossard Teixeira

Elementos Essenciais

- **Nome:** *Abstract Factory (Kit)*
- **Problema:** Desconhecer o tipo do objeto que será criado de forma prévia e estática. Mas o problema aqui possui a característica de agrupar famílias de objetos. Trata de um contexto similar ao do *Factory Method*, mas ao invés da Fábrica ter um método de criação passa a ter 2 ou mais.
- **Solução:** Fornecer uma interface para criação de famílias de objetos relacionados ou dependentes sem especificar suas classes concretas.
- **Consequências:** As mesmas do *Factory Method*.

Problema

- Vamos utilizar como o problema o mesmo discutido na aula sobre o *Factory Method*. Queremos 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. Para exemplificar veja como os objetos devem ser organizados:
  - Sedan:
    - Siena – Fiat
    - Fiesta Sedan – Ford
  - Popular:
    - Palio – Fiat
    - Fiesta – Ford

Fonte: https://brizeno.wordpress.com/category/padroes-de-projeto/abstract-factory/

Abstract Method

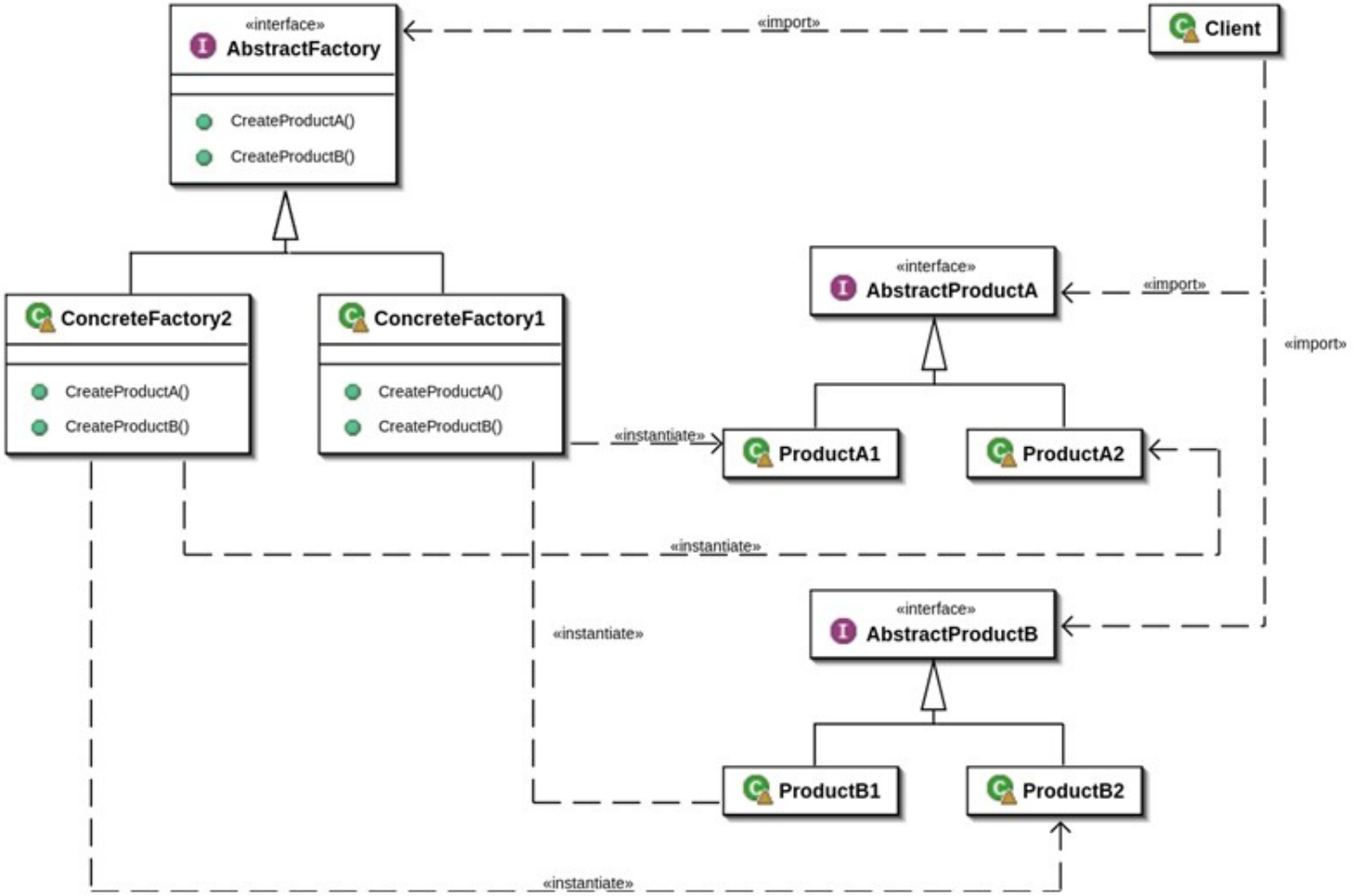
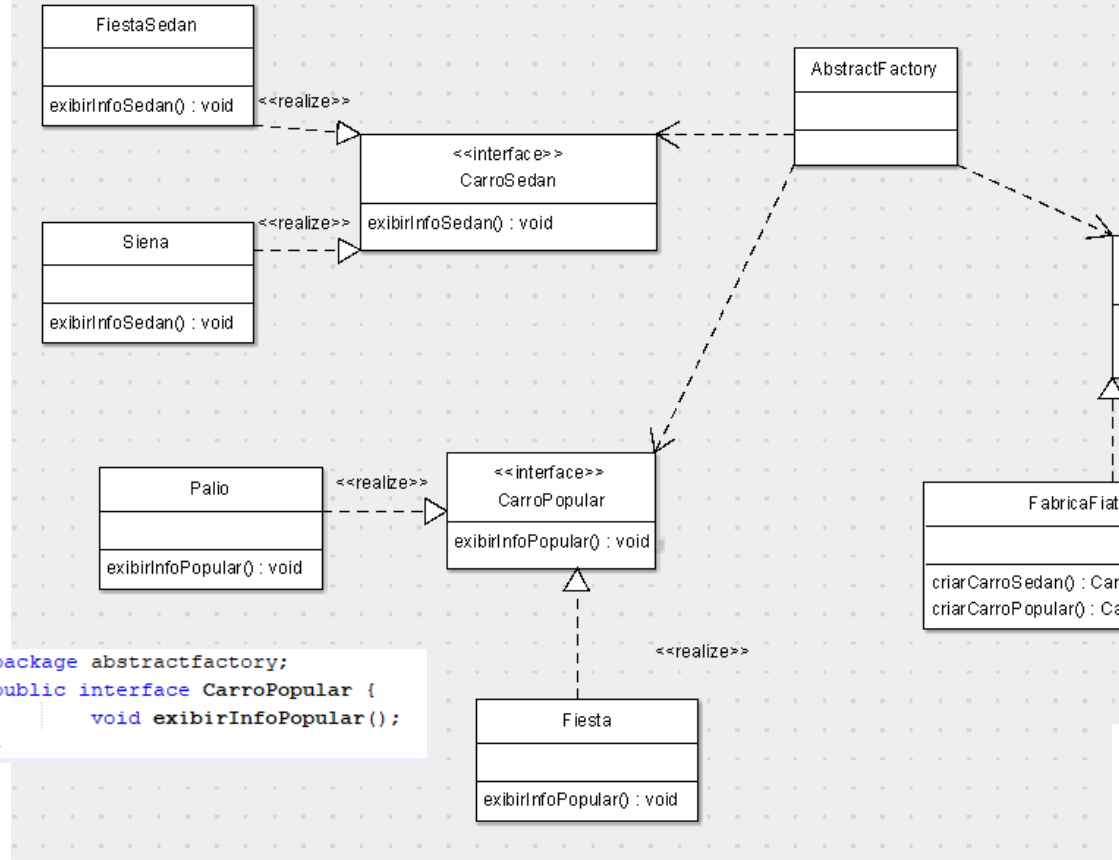


Diagrama de classes do Exemplo

```java
package abstractfactory;
public class FiestaSedan implements CarroSedan {
        @Override
        public void exibirInfoSedan() {
                System.out.println("Modelo: Fiesta\nFábrica:Ford\nCategoria:Sedan\n\n");
        }
}
```




```java
package abstractfactory;
public class AbstractFactory {
        public static void main(String[] args)
        {
                FabricaDeCarro fabrica;
                CarroSedan sedan;
                CarroPopular popular;

                fabrica = new FabricaFiat();

                sedan       = fabrica.criarCarroSedan();
                sedan.exibirInfoSedan();

                popular     = fabrica.criarCarroPopular();
                popular.exibirInfoPopular();

                fabrica = new FabricaFord();

                sedan       = fabrica.criarCarroSedan();
                sedan.exibirInfoSedan();

                popular     = fabrica.criarCarroPopular();
                popular.exibirInfoPopular();
        }
}
```

```java
package abstractfactory;
public interface FabricaDeCarro
{
    CarroSedan   criarCarroSedan();
    CarroPopular criarCarroPopular();
}
```

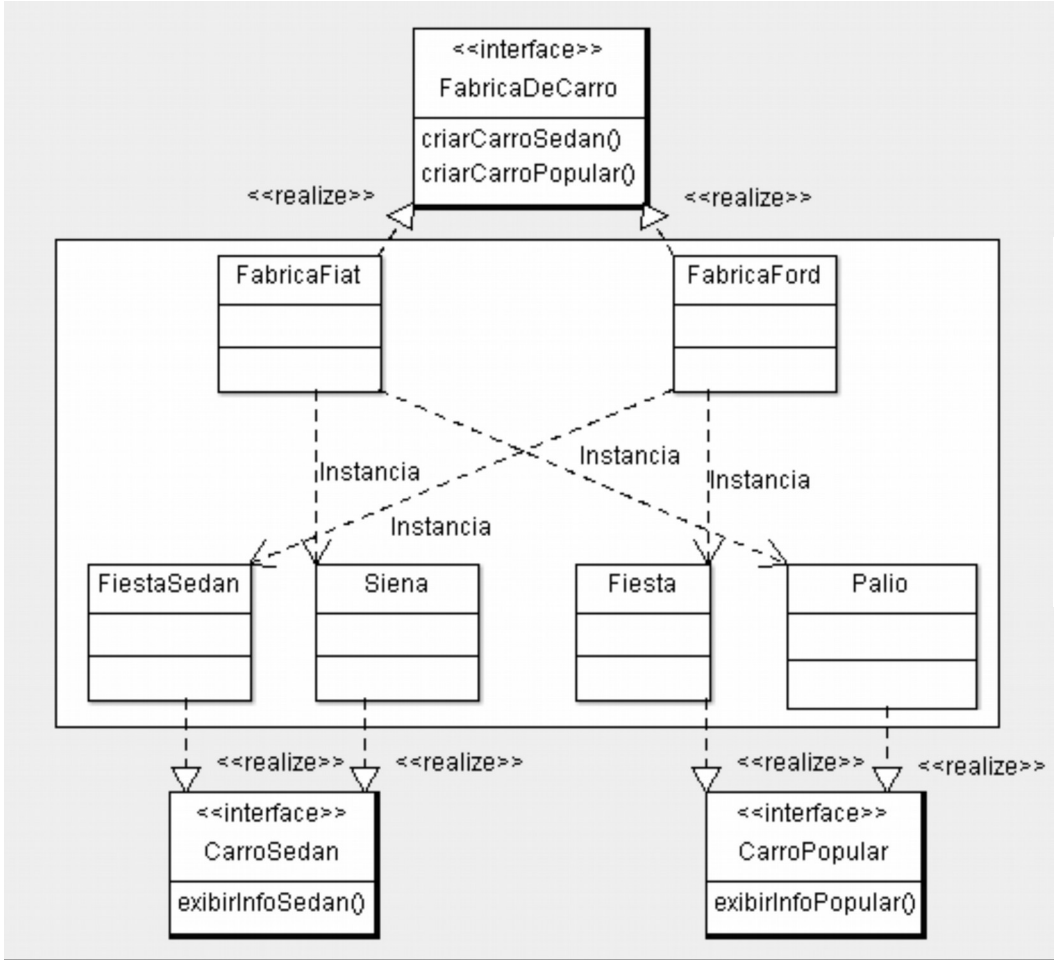
```java
package abstractfactory;
public interface CarroPopular {
        void exibirInfoPopular();
}
```

```java
package abstractfactory;
public class FabricaFord implements FabricaDeCarro {
        @Override
        public CarroSedan criarCarroSedan() {
                return new FiestaSedan();
        }

        @Override
        public CarroPopular criarCarroPopular() {
                return new Fiesta();
        }
}
```


Instituto Federal do Espirito Santo – Campus Colatina
Disciplina: Projeto de Sistemas
Professor: Giovany Frossard Teixeira

Diagrama de classes do Exemplo – Mostrando instanciações




```
package abstractfactory;
public interface FabricaDeCarro
{
    CarroSedan   criarCarroSedan();
    CarroPopular criarCarroPopular();
}
```

```
package abstractfactory;
public class FabricaFord implements FabricaDeCarro {
        @Override
        public CarroSedan criarCarroSedan() {
                return new FiestaSedan();
        }

        @Override
        public CarroPopular criarCarroPopular() {
                return new Fiesta();
        }
}
```

Abstract Factory
Quando Usar?

- ✓ Quando um sistema deve ser independente de como seus produtos são criados
- ✓ Quando um sistema deve ser configurado com uma entre várias famílias de produtos
- ✓ Quando uma família de produtos relacionados foi projetada para uso conjunto e você deve implementar essa restrição
- ✓ Quando deseja fornecer uma biblioteca de classes e quer revelar sua interface e não sua implementação
- ✓ Não permitir que objetos sejam diretamente criados com new

Dúvidas?