Códigos

Exemplo de *Abstract Factory* Config.

Arquivo *1.yaml*:

```yaml
#
# Fornecer uma interface para criação de famílias de objetos relacionados ou
# dependentes sem especificar suas classes concretas.
#

Family:                     # Família de tipos
- Family                    # Nome do tipo
- FamilyTypes:              # Nome da Família de tipos
    - Type1                 # primeiro tipo definido
    - Type2

AbstractFactory:            # Classe *AbstractFactory*
- AbstractFactory:          # Nome da Classe *AbstractFactory*
    - get                   # Método chamador
    - CreateProductA:       # Primeiro método abstrato de *AbstractFactory*
        - AbstractProductA: #
            - doIt
            - ProductA1
            - ProductA2

ConcreteFactory:
- ConcreteFactory1          # Cria o Produto A1
- ConcreteFactory2          # Cria o Produto A2

Client:
- Cliente
```

Arquivo *2.yaml*:

```yaml
#
# Fornecer uma interface para criação de famílias de objetos relacionados ou
# dependentes sem especificar suas classes concretas.
#

Family:                       # Família de tipos
- Family                      # Nome do tipo
- FamilyTypes:                # Nome da Família de tipos
    - Type1                   # primeiro tipo definido
    - Type2                   # segundo tipo definido
    - Type3                   # terceiro tipo definido

AbstractFactory:              # Classe *AbstractFactory*
- AbstractFactory:            # Nome da Classe *AbstractFactory*
    - get                     # Método chamador
    - CreateProductA:         # Primeiro método abstrato de *AbstractFactory*
        - AbstractProductA: #
            - doIt
            - ProductA1
            - ProductA2
            - ProductA3
    - CreateProductB:         # Segundo método abstrato de *AbstractFactory*
        - AbstractProductB:
            - doTi
            - ProductB1
            - ProductB2
            - ProductB3
#    - CreateProductC:        # Terceiro método abstrato de *AbstractFactory*
#            - AbstractProductC:
#                - doTI
#                - ProductC1
#                - ProductC2

ConcreteFactory:
- ConcreteFactory1            # Cria o Produto A1 e produto B1
- ConcreteFactory2            # CriaçãoProduto A2 e primeiroB2
- ConcreteFactory3            # CriaçãoProduto A3 e primeiroB3

Client:
- Client
```

Arquivo *3.yaml*:

```yaml
#
# Fornecer uma interface para criação de famílias de objetos relacionados ou
# dependentes sem especificar suas classes concretas.
#

Family:                       # Família de tipos
- Family                      # Nome do tipo
- FamilyTypes:                # Nome da Família de tipos
    - Type1                   # primeiro tipo definido
    - Type2                   # segundo tipo definido
    - Type3                   # terceiro tipo definido

AbstractFactory:              # Classe *AbstractFactory*
- AbstractFactory:            # Nome da Classe *AbstractFactory*
    - get                     # Método chamador
    - CreateProductA:         # Primeiro método abstrato de *AbstractFactory*
        - AbstractProductA:   #
            - doIt
            - ProductA1
            - ProductA2
            - ProductA3
    - CreateProductB:         # Segundo método abstrato de *AbstractFactory*
        - AbstractProductB:
            - doTi
            - ProductB1
            - ProductB2
            - ProductB3
    - CreateProductC:         # Terceiro método abstrato de *AbstractFactory*
        - AbstractProductC:
            - doTI
            - ProductC1
            - ProductC2

ConcreteFactory:
- ConcreteFactory1            # Cria o Produto A1, B1 e C1
- ConcreteFactory2            # CriaçãoProduto A2, B2 e C2
- ConcreteFactory3            # CriaçãoProduto A3, B3 e C3

Client:
- Client                      # Nome do arquivo principal
```

Arquivo *4.yaml*:

```yaml
#
# Fornecer uma interface para criação de famílias de objetos relacionados ou
# dependentes sem especificar suas classes concretas.
#

Family:                         # Família de tipos
- Api                           # Nome do tipo
- FamilyTypes:                  # Nome da Família de tipos
    - TypeQt                    # primeiro tipo definido
    - TypeMotif                 # segundo tipo definido

AbstractFactory:                # Classe *AbstractFactory*
- AbstractFactory:              # Nome da Classe *AbstractFactory*
    - get                       # Método chamador
    - CreateWindow:             # Primeiro método abstrato de *AbstractFactory*
        - AbstractWindow:       #
            - doIt
            - WindowQt
            - WindowMotif
    - CreateScrollBar:          # Segundo método abstrato de *AbstractFactory*
        - AbstractScrollBar:
            - doTi
            - ScrollBarQt
            - ScrollBarMotif
    - CreateColor:              # Terceiro método abstrato de *AbstractFactory*
        - AbstractColor:
            - doTI
            - ColorQt
            - ColorMotif

ConcreteFactory:
- FactoryQt                     # Cria o Produto AQt, BQt e CQt
- FactoryMotif                  # CriaçãoProduto AMotif, BMotif e CMotif

Client:
- Client                        # Nome do arquivo principal
```

Arquivo *5.yaml*:

```yaml
#
# Fornecer uma interface para criação de famílias de objetos relacionados ou
# dependentes sem especificar suas classes concretas.
#

Family:                         # Família de tipos
- Api                           # Nome do tipo
- ApiTypes:                     # Nome da Família de tipos
    - Qt                        # primeiro tipo definido
    - Motif                     # segundo tipo definido

AbstractFactory:                # Classe *AbstractFactory*
- Widget:                       # Nome da Classe *AbstractFactory*
    - get                       # Método chamador
    - CreateWindow:             # Primeiro método abstrato de *AbstractFactory*
        - Window:               #
            - doIt
            - WindowQt
            - WindowMotif
    - CreateScrollBar:          # Segundo método abstrato de *AbstractFactory*
        - ScrollBar:
            - doTi
            - ScrollBarQt
            - ScrollBarMotif
    - CreateColor:              # Terceiro método abstrato de *AbstractFactory*
        - Color:
            - doTI
            - ColorQt
            - ColorMotif

ConcreteFactory:
- FactoryQt                     # Cria o Produto AQt, BQt e CQt
- FactoryMotif                  # CriaçãoProduto AMotif, BMotif e CMotif

Client:
- Client                        # Nome do arquivo principal
```

Arquivo *Calculadora.yaml*:

```yaml
#
# Fornecer uma interface para criação de famílias de objetos relacionados ou
# dependentes sem especificar suas classes concretas.
#

Family:                     # Família de tipos
- Operacoes                 # Nome do tipo
- TiposDeOperacoes:         # Nome da Família de tipos
    - Soma                  # primeiro tipo definido
    - Subtracao
    - Divisao
    - Multiplicacao

AbstractFactory:            # Classe *AbstractFactory*
- Calculadora:              # Nome da Classe *AbstractFactory*
    - get                   # Método chamador
    - Solucao:              # Primeiro método abstrato de *AbstractFactory*
        - OperacaoMatematica: #
            - calcula
            - Soma
            - Subtracao
            - Divisao
            - Multiplicacao

ConcreteFactory:
- oSoma             # Cria o Produto A1
- oSubtracao        # Cria o Produto A2
- oDivisao          # Cria o Produto A2
- oMultiplicacao    # Cria o Produto A2

Client:
- Cliente
```

Arquivo *fabricaX.yaml*:

```yaml
#
# Fornecer uma interface para criação de famílias de objetos relacionados ou
# dependentes sem especificar suas classes concretas.
#

Family:                     # Família de tipos
- Materiais                 # Nome do tipo
- TiposDeMateriais:         # Nome da Família de tipos
    - Ferro                 # primeiro tipo definido
    - Madeira
    - Plastico

AbstractFactory:            # Classe *AbstractFactory*
- FabricaX:                 # Nome da Classe *AbstractFactory*
    - get                   # Método chamador
    - CreateCadeira:        # Primeiro método abstrato de *AbstractFactory*
        - Cadeira:          #
            - doIt
            - CadeiraDeFerro
            - CadeiraDeMadeira
            - CadeiraDePlastico

ConcreteFactory:
- Ferro      # Cria o Produto A1
- Madeira    # Cria o Produto A2
- Plastico   # Cria o Produto A2

Client:
- Cliente
```

Arquivo *fabricaY.yaml*:

```
#
# Fornecer uma interface para criação de famílias de objetos relacionados ou
# dependentes sem especificar suas classes concretas.
#

Family:                     # Família de tipos
- Materiais                 # Nome do tipo
- TiposDeMateriais:         # Nome da Família de tipos
    - Ferro                 # primeiro tipo definido
    - Madeira
    - Plastico

AbstractFactory:            # Classe *AbstractFactory*
- FabricaX:                 # Nome da Classe *AbstractFactory*
    - get                   # Método chamador
    - CreateCadeira:        # Primeiro método abstrato de *AbstractFactory*
        - Cadeira:          #
            - doIt
            - CadeiraDeFerro
            - CadeiraDeMadeira
            - CadeiraDePlastico
    - CreatePorta:          # Primeiro método abstrato de *AbstractFactory*
        - Porta:            #
            - doIt
            - PortaDeFerro
            - PortaDeMadeira
            - PortaDePlastico
    - CreateCabide:         # Primeiro método abstrato de *AbstractFactory*
        - Cabide:           #
            - doIt
            - CabideDeFerro
            - CabideDeMadeira
            - CabideDePlastico
    - CreateCamas:          # Primeiro método abstrato de *AbstractFactory*
        - Camas:            #
            - doIt
            - CamasDeFerro
            - CamasDeMadeira
            - CamasDePlastico

ConcreteFactory:
- Ferro      # Cria o Produto A1
- Madeira    # Cria o Produto A2
- Plastico   # Cria o Produto A2
```

```
Client:
- Cliente
```

Arquivo *onOff.yaml*:

```
#
# Fornecer uma interface para criação de famílias de objetos relacionados ou
# dependentes sem especificar suas classes concretas.
#

Family:                     # Família de tipos
- EstadoLampada             # Nome do tipo
- Estados:                  # Nome da Família de tipos
    - Ligada                # primeiro tipo definido
    - Desligada

AbstractFactory:            # Classe *AbstractFactory*
- MinhaLampada:             # Nome da Classe *AbstractFactory*
    - get                   # Método chamador
    - SelecioneLampada:     # Primeiro método abstrato de *AbstractFactory*
        - Lampada:          #
            - apertaPlug
            - Liga
            - Desliga

ConcreteFactory:
- ConcreteFactory1          # Cria o Produto A1
- ConcreteFactory2          # Cria o Produto A2

Client:
- Cliente
```

Arquivo *Perfumes.yaml*:

```yaml
#
# Fornecer uma interface para criação de famílias de objetos relacionados ou
# dependentes sem especificar suas classes concretas.
#

Family:                     # Família de tipos
- Perfume                   # Nome do tipo
- sexo:                     # Nome da Família de tipos
    - Masculinos            # primeiro tipo definido
    - Femininos

AbstractFactory:            # Classe *AbstractFactory*
- Boticario:                # Nome da Classe *AbstractFactory*
    - get                   # Método chamador
    - PerfumesDeVerao:      # Primeiro método abstrato de *AbstractFactory*
        - Colonias:         #
            - doIt
            - ColoniaMasculina
            - ColoniaFeminina

ConcreteFactory:
- ConcreteFactory1          # Cria o Produto A1
- ConcreteFactory2          # Cria o Produto A2

Client:
- Cliente
```

Arquivo *simple.yaml*:

```yaml
#
# Fornecer uma interface para criação de famílias de objetos relacionados ou
# dependentes sem especificar suas classes concretas.
#

Family:                     # Família de tipos
- Family                    # Nome do tipo
- FamilyTypes:              # Nome da Família de tipos
    - Type1                 # primeiro tipo definido

AbstractFactory:            # Classe *AbstractFactory*
- AbstractFactory:          # Nome da Classe *AbstractFactory*
    - get                   # Método chamador
    - CreateProductA:       # Primeiro método abstrato de *AbstractFactory*
        - AbstractProductA: #
            - doIt
            - ProductA1

ConcreteFactory:
- ConcreteFactory1          # Cria o Produto A1

Client:
- Cliente
```

Arquivo *widget.yaml*:

```
#
# Fornecer uma interface para criação de famílias de objetos relacionados ou
# dependentes sem especificar suas classes concretas.
#

Family:                         # Família de tipos
- Api                           # Nome do tipo
- FamilyTypes:                  # Nome da Família de tipos
    - TypeQt                    # primeiro tipo definido
    - TypeMotif                 # segundo tipo definido

AbstractFactory:                # Classe *AbstractFactory*
- AbstractFactory:              # Nome da Classe *AbstractFactory*
    - get                       # Método chamador
    - CreateWindow:             # Primeiro método abstrato de *AbstractFactory*
        - AbstractWindow:       #
            - doIt
            - WindowQt
            - WindowMotif
    - CreateScrollBar:          # Segundo método abstrato de *AbstractFactory*
        - AbstractScrollBar:
            - doTi
            - ScrollBarQt
            - ScrollBarMotif
    - CreateColor:              # Terceiro método abstrato de *AbstractFactory*
        - AbstractColor:
            - doTI
            - ColorQt
            - ColorMotif

ConcreteFactory:
- FactoryQt                     # Cria o Produto AQt, BQt e CQt
- FactoryMotif                  # CriaçãoProduto AMotif, BMotif e CMotif

Client:
- Client                        # Nome do arquivo principal
```

Arquivo *Widget.yaml*:

```
#
# Fornecer uma interface para criação de famílias de objetos relacionados ou
# dependentes sem especificar suas classes concretas.
#

Family:                         # Família de tipos
- Api                           # Nome do tipo
- FamilyTypes:                  # Nome da Família de tipos
    - TypeQt                    # primeiro tipo definido
    - TypeMotif                 # segundo tipo definido

AbstractFactory:                # Classe *AbstractFactory*
- AbstractFactory:              # Nome da Classe *AbstractFactory*
    - get                       # Método chamador
    - CreateWindow:             # Primeiro método abstrato de *AbstractFactory*
        - Window:               #
            - draw
            - WindowQt
            - WindowMotif
    - CreateScrollBar:          # Segundo método abstrato de *AbstractFactory*
        - ScrollBar:
            - draw
            - ScrollBarQt
            - ScrollBarMotif
    - CreateColor:              # Terceiro método abstrato de *AbstractFactory*
        - Color:
            - setColor
            - ColorQt
            - ColorMotif

ConcreteFactory:
- FactoryQt                     # Cria o Produto AQt, BQt e CQt
- FactoryMotif                  # CriaçãoProduto AMotif, BMotif e CMotif

Client:
- Client                        # Nome do arquivo principal
```