Abstract Factory

1. Pesquise sobre a implementação do padrão Abstract Factory em algum problema e obtenha o código que foi usado para a solução.

   O Abstract Factory tem o mesmo objetivo principal das outras duas versões, remover código de criação (aquele com a palavra reservada new) de nossas classes de negócio.

   No nosso exemplo iremos citar uma Pizzaria, onde há duas cidades e mesmos sabores, porém com ingredientes distintos.

```
1  public class PizzaQuatroQueijos extends Pizza {
2      private String cidade;
3      private Massa massa;
4      private Queijo queijo;
5      private Molho molho;
6      private Salsa salsa;
7
8      public PizzaQuatroQueijos( String cidade ){
9          this.cidade = cidade;
10     }
11
12     public void prepara(){
13
14         if( cidade.equals(sao-paulo) ){
15             massa = new MassaGrossa();
16             queijo = new QueijoMinas();
17             molho = new MolhoMarroquino();
18             salsa = new SalsaNobre();
19         }
20         else{
21             massa = new MassaFina();
22             queijo = new QueijoParmesao();
23             molho = new MolhoDaRoca();
24             salsa = new SalsaNobreApimentada();
25         }
26     }
27     ...
28 }
```

   Para um melhor entendimento: o queijo utilizado em todas as pizzas que precisam de queijo, na filial de São Paulo, é sempre o QueijoMinas, a massa, se necessária, sempre é MassaGrossa e assim por diante.

Diferente de ter apenas uma pizza ou paulista ou carioca para cada sabor, ou seja, algo que seria facilmente atendido por um Factory Method. Temos uma família de objetos quando é uma pizza paulista e uma outra família quando é uma carioca.

Agora podemos partir para a solução. Vamos a nossa classe abstrata que representa a interface de implementação do padrão Abstract Factory, **PizzaIngredientesFactory:**

```
1  public abstract class PizzaIngredientesFactory {
2      public abstract Massa criarMassa();
3
4      public abstract Queijo criarQueijo();
5
6      public abstract Molho criarMolho();
7
8      public abstract Salsa criarSalsa();
9  }
```

Agora como fábricas concretas teremos uma factory para São Paulo e outra para Rio de Janeiro. Começando com a **SPPizzaIngredientesFactory:**

```
1  public class SPPizzaIngredientesFactory extends PizzaIngredientesFactory {
2      @Override
3      public Massa criarMassa() {
4          return new MassaGrossa();
5      }
6
7      @Override
8      public Queijo criarQueijo() {
9          return new QueijoMinas();
10     }
11
12     @Override
13     public Molho criarMolho() {
14         return new MolhoMarroquino();
15     }
16
17     @Override
18     public Salsa criarSalsa() {
19         return new SalsaNobre();
20     }
21 }
```

E então a **RJPizzaIngredientesFactory:**

```
1  public class RJPizzaIngredientesFactory extends PizzaIngredientesFactory {
2      @Override
3      public Massa criarMassa() {
4          return new MassaFina();
5      }
6
7      @Override
8      public Queijo criarQueijo() {
9          return new QueijoParmesao();
10     }
11
12     @Override
13     public Molho criarMolho() {
14         return new MolhoDaRoca();
15     }
16
17     @Override
18     public Salsa criarSalsa() {
19         return new SalsaNobreApimentada();
20     }
21 }
```

Agora nossas fábricas já podem ser utilizadas nas subclasses de Pizza. Vamos passá-las como dependências nos construtores dessas subclasses de Pizza. Como exemplo vamos ao código da classe **PizzaQuatroQueijos:**

```
1  public class PizzaQuatroQueijos extends Pizza {
2      private PizzaIngredientesFactory ingredientes;
3      private Massa massa;
4      private Queijo queijo;
5      private Molho molho;
6      private Salsa salsa;
7
8      public PizzaQuatroQueijos( PizzaIngredientesFactory ingredientes ){
9          this.ingredientes = ingredientes;
10     }
11     public void prepara(){
12         massa = ingredientes.criarMassa();
13         queijo = ingredientes.criarQueijo();
14         molho = ingredientes.criarMolho();
15         salsa = ingredientes.criarSalsa();
16     }
17 }
```

Agora podemos voltar ao código de Pizzaria, porém com a versão mais atual, utilizando as **fábricas** de ingredientes:

```
1  public class Pizzaria {
2      private Pizza pizza;
3
4      public void criarPizza( String cidade, String tipo ){
5
6          if( tipo.equals(queijo) ){
7              pizza = new PizzaQuatroQueijos( getIngredientes(cidade) );
8          }
9          else if( tipo.equals(portuguesa) ){
10             pizza = new PizzaPortuguesa( getIngredientes(cidade) );
11         }
12         else if( tipo.equals(calabresa) ){
13             pizza = new PizzaCalabresa( getIngredientes(cidade) );
14         }
15     }
16
17     private PizzaIngredientesFactory getIngredientes( String cidade ){
18         if( cidade.equals(sao-paulo) ){
19             return new SPPizzaIngredientesFactory();
20         }
21         return new RJPizzaIngredientesFactory();
22     }
23     ...
24 }
```

2. Informe qual a linguagem de programação o padrão foi criado.

   A linguagem utilizada foi **JAVA**.

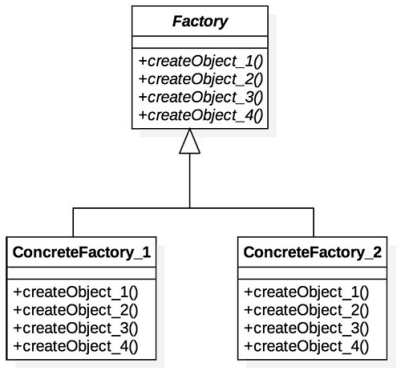

Diagrama Abstract Factory

Referências

[1] Vinícius Thiengo. Padrão de Projeto: Abstract Factory. https://www.thiengo.com.br/padrao-de-projeto-abstract-factory. [Online; acessado 18 de abril de 2020].