Departamento de Computação – DComp - IFMA
Padrões de Software – 7º Período
Entrega: 03-09-2021

Atividade Etapa 02 – Parte 03

Padrão de Projeto Abstract Factory

Preâmbulo

Se em algum momento tivermos objetos fortemente relacionados em nosso sistema, podemos separar a criação deles em uma única classe, para garantir que eles sejam criados sempre em conjunto com o seu "par" correto. Essa família de classes estendem o que é conhecida como Abstract Factory, por ser uma fábrica abstrata, que pode criar famílias de objetos, e não apenas um objeto específico.

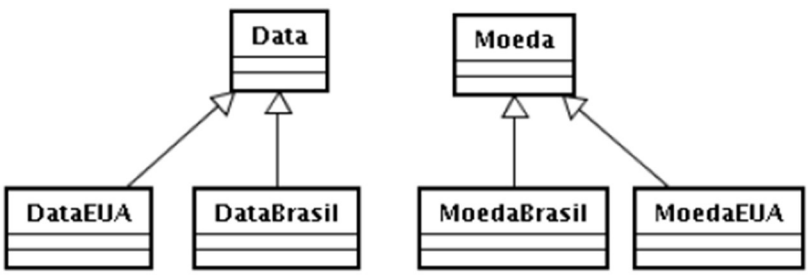
Veja exemplos de implementação nesse link:

https://refactoring.guru/design-patterns/abstract-factory

Questão

A implementação deste padrão envolve o emprego de várias classes, em um cenário relativamente complexo de interação entre as classes. Suponha a existência de uma classe Data e a classe Moeda. A primeira representa uma data qualquer, por exemplo, 7 de setembro de 2004. A outra representa uma quantidade em dinheiro, por exemplo, R$25,00. Em ambos os casos você deve ter observado que a forma empregada para exibir a data e a quantia em dinheiro são típicas do Brasil. Para um país de língua inglesa outra forma deveria ser utilizada. Ou seja, September 07, 2004 para a data e $25.00 para a moeda (estou assumindo a paridade de um dólar para um real).

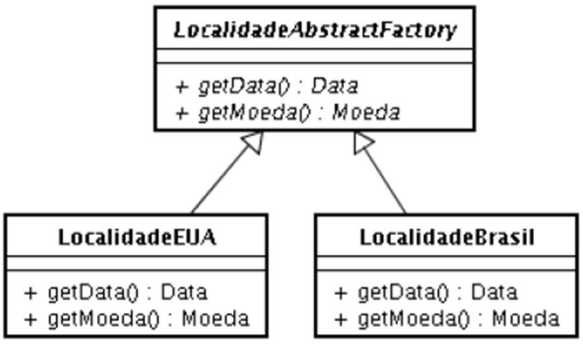
O que é preciso neste caso, é a implementação das classes Data e Moeda conforme a localidade. Para o Brasil, por exemplo, tem-se implementações distintas daquelas para países de língua inglesa, conforme a figura abaixo.



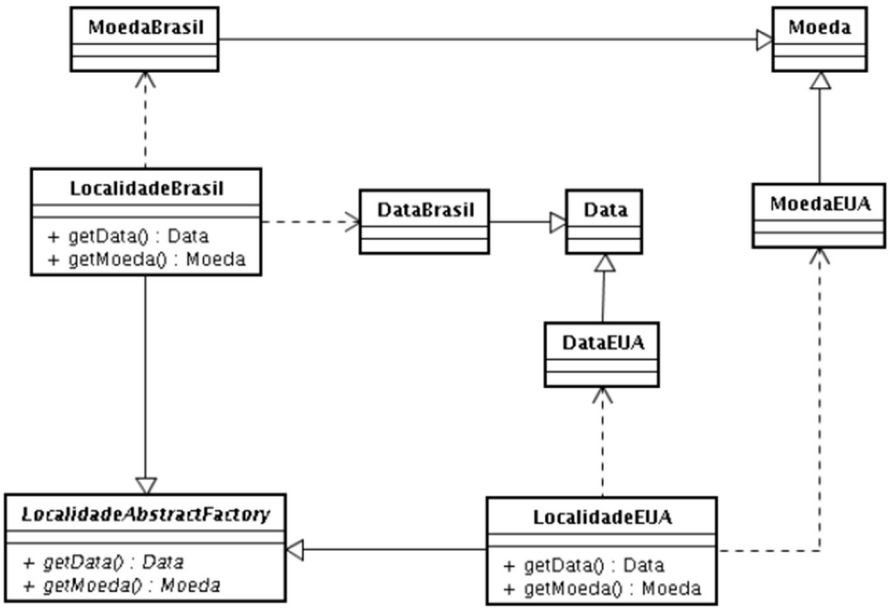
Para uma aplicação cliente, o padrão Abstract Factory pode ser empregado para evitar que a aplicação use, por exemplo, a implementação para o Brasil da classe Data e a implementação para um país de língua inglesa para a classe Moeda, misturando data de um com moeda de outro. O que é preciso, neste caso, é recuperar instâncias de classes correlatas.

Uma classe que implemente os métodos getMoeda e getData podem identificar a localidade, através de um teste, por exemplo, e retornar a instância da classe identificada. Contudo, esta abordagem é inapropriada, principalmente à medida que o número de opções crescer. Dois países não oferecem tantos desafios, mas não podemos dizer o mesmo para centenas deles. A manutenção seria dificultada, mesmo todo este código estando devidamente confinado em uma única classe.

Uma abordagem mais atrativa envolve o emprego de uma classe abstrata, LocalidadeAbstractFactory, nossa fábrica abstrata de objetos. Esta classe deve definir os métodos getMoeda e getData, mas não implementá-los, tarefa que seria delegada para as subclasses concretas, tantas quantas forem as opções existentes. Para o nosso exemplo e para a localidade do Brasil pode-se criar a classe LocalidadeBrasil, para a localidade EUA pode-se criar a classe LocalidadeEUA, conforme ilustra a figura abaixo



A implementação dos métodos getMoeda e getData é trivial, simplesmente retornando uma instância da classe MoedaBrasil e DataBrasil, respectivamente. Naturalmente, MoedaBrasil herda da classe Moeda e DataBrasil herda da classe Data. Estas classes de sufixo Brasil, convém relembrar, devem implementar o formato empregado no Brasil para a exibição de valores monetários e de datas. Algo similar seria obtido com outra localidade qualquer, conforme ilustra a figura.



A solução apresentada no diagrama acima faz uso do padrão Abstract Factory. Para a aplicação cliente, convém ressaltar, há dependências para as classes LocalidadeAbstractFactory, Moeda e Data. O cliente depende de LocalidadeAbstractFactory porque é a partir desta que as instâncias adequadas de Moeda e Data são obtidas. Naturalmente, código cliente depende das classes Moeda e Data, pois são estas que são efetivamente fornecem a funcionalidade desejada. Obserque que o código cliente, ao fazer uso de uma instância de Moeda não sabe se se trata de uma instância da classe MoedaBrasil ou MoedaEUA, que foi retornada por uma classe derivada de LocalidadeAbstractFactory, também desconhecida da aplicação.

Implemente o modelo comentado acima, onde as seguintes restrições deverão ser satisfeitas.

i. Uma classe Cliente deverá representar código cliente.

ii. Crie a classe Factory a partir da qual o método newLocalidade() deverá retornar uma instância de LocalidadeAbstractFactory. A decisão deverá vir da configuração de um arquivo de propriedades, conhecido pela classe Factory. A propriedade localidade deverá indicar Brasil para a localidade do Brasil ou EUA para a localidade dos EUA.

iii. A classe Cliente, ao fazer uso de Data e Moeda, não deverá conhecer a implementação da classe LocalidadeAbstractFactory assim como também não conhecerá quem implementa Data e Moeda.

iv. A funcionalidade depositada em Data é apresentar, conforme a localidade, a dia da semana corrente. Por exemplo, para o Brasil pode ser “Seg”, enquanto para a localidade EUA seria “Mon”.

v. A funcionalidade de Moeda é apresentar um valor qualquer na moeda em questão. Por exemplo, para o Brasil, R$10,0 é uma saída correta, enquanto para os EUA um valor correto seria $5.00. Esta funcionalidade deverá ser obtida, tanto da classe Moeda quanto da classe Data através do método toString(), herdado de Object.