UNIESP - CENTRO UNIVERSITÁRIO

SISTEMAS PARA INTERNET

GABRIEL CAVALCANTE MELO

HENRIQUE MELO LEAL

JARDEL PRAXEDES DA COSTA

LEONARDO DO NASCIMENTO PEIXOTO DA SILVA

ENTENDENDO O PADRÃO DE PROJETO ABSTRACT FACTORY

CABEDELO - PB,

2023

Gabriel Cavalcante Melo
Henrique Melo Leal
Jardel Praxedes Da Costa
Leonardo do Nascimento Peixoto da Silva

ENTENDENDO O PADRÃO DE PROJETO ABSTRACT FACTORY

Trabalho apresentado na disciplina de Padrões de Projeto, no curso de Sistemas para Internet, para obtenção da 2ª nota parcial.

Orientador(a): Prof. Angelo Francescoly Dias Gonçalves.

Cabedelo - PB,
2023

SUMÁRIO

1 INTRODUÇÃO

O presente trabalho tem como finalidade discutir sobre o padrão de projeto Abstract Factory que pertence a família Factory, composta por três tipos (Simple Factory, Factory Method e Abstract Factory).

Os Padrões de Projeto (Design Pattern) podem ser entendidos com modelos de soluções para problemas comuns dentro do desenvolvimento de software. Esses são adaptáveis e consegue-se realizar alterações para as diversas situações do dia a dia que exigem alguma implementação por meio da programação. Tais padrões funcionam como um verdadeiro kit de ferramentas para as diversas soluções já anteriormente testadas em problemas comuns. Os Padrões de Projetos podem ser classificados em três tipos:

1. Os **padrões criacionais** são focados em soluções para criação de objetos que aumentam a flexibilidade e a reutilização de código.
2. Os **padrões estruturais** explicam como montar objetos e classes em estruturas maiores, enquanto ainda mantém as estruturas flexíveis e eficientes.
3. Os **padrões comportamentais** cuidam da comunicação eficiente e da assinalação de responsabilidades entre objetos.

Nesse contexto, abordar-se-á o padrão de projeto Abstract Factory, que pertence ao tipo criacional, sendo um padrão amplamente utilizado para fornecer uma maneira de criar famílias de objetos relacionados ou dependentes, sem especificar suas classes concretas.

2 DESENVOLVIMENTO

2.1 SOBRE O PADRÃO ABSTRACT FACTORY

Uma Fábrica (Factory), dentro de padrões de projetos, é a implementação de uma classe concreta que cria os objetos dependentes dessa classe. Nesse cenário, o Abstract Factory trata-se de um padrão de projeto desse grupo que lida com a criação de objetos a fim de agrupar as famílias de produtos ou classes relacionadas.

Tal padrão funciona de uma maneira mais complexa do que um Simple Factory e Factory Method. Nele, o Abstract Factory Interface pode dispor de várias classes 'Concrete Factory' - ao contrário da Factory Method por exemplo - e tais podem produzir diferentes versões de um mesmo produto. Por exemplo, em uma Factory, do tipo Abstract Factory, é possível produzir cadeiras, sofás e mesas de centro (lembre-se do conceito inicial desse padrão). Cada um desses produtos possuem uma interface (Cadeira, Sofá e Mesa de Centro) que pode ser utilizada para uma ou mais diferentes classes concretas desse objeto. Diante desse contexto, cada classe Concrete Factory pode estar relacionada com um ConcreteProdut de Cadeira, Sofá e Mesa de Centro.

Resumidamente, o propósito desse padrão é criar famílias de objetos relacionados ou dependentes por meio de uma única interface Factory.

2.2 VANTAGENS DO PADRÃO

Dentro das Vantagens de se utilizar tal projeto, pode-se elencar as seguintes:

- Criação de famílias de objetos que se relacionam entre si., de forma que não seja necessário a criação de várias classes referindo-se a mesma coisa;
- Uma melhor funcionalidade aos objetos relacionados de maneira mais contextualizada;
- Flexibilidade de novas funcionalidades sem alterar o código, criando novas implementações da fábrica abstrata.

2.3 DESVANTAGENS DO PADRÃO

As desvantagens de utilizar esse tipo de padrão são poucos mas muito prejudiciais em códigos de grande complexidade. Exemplos de problemas que podem ocorrer são:

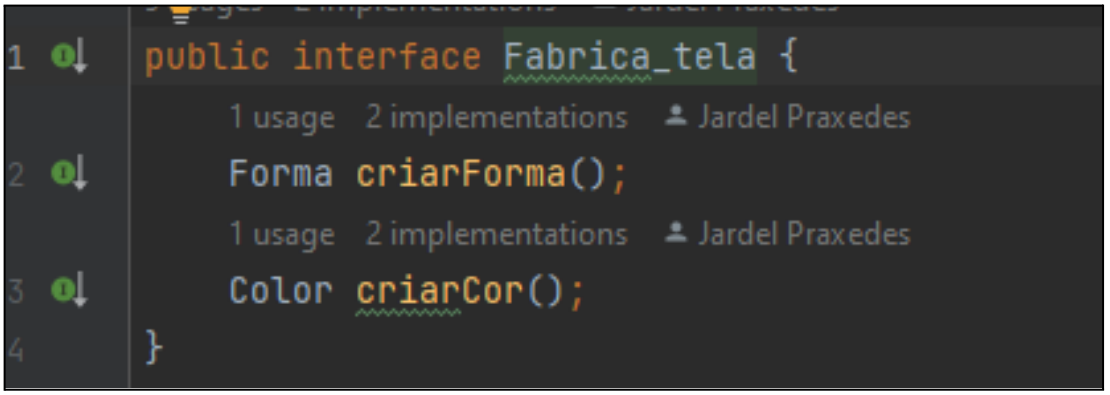
- Implementar o abstract Factory quando é mais simples usar o Simple Factory ou Factory Method seria suficiente, pode piorar a performance do programa;
- Dificuldade na adição e remoção de produtos da família, pois deve-se modificar todas as implementações da Factory e o cliente que usa a Abstract Factory;
- O código pode se tornar complicado, uma vez que muitas interfaces e classes são adicionadas junto com o padrão.

2.4 APLICAÇÃO PRÁTICA

Seguindo a lógica estabelecida pelo conceito de Abstract Factory, a aplicação prática trabalha com uma fábrica de criação de figuras. Seu funcionamento, baseado nos aspectos já conhecidos desse modelo, é composto por interfaces e classes concretas.

Em relação às interfaces, há três delas: Fabrica_tela, Color e Forma. Elas são a base de como as classes concretas relacionadas a elas agirão. Sobre a Fabrica_tela, essa tem em seu “contrato” os seguintes métodos (Figura 1):

Figura 1 - Composição da interface Fabrica_tela



Fonte: própria, 2023.

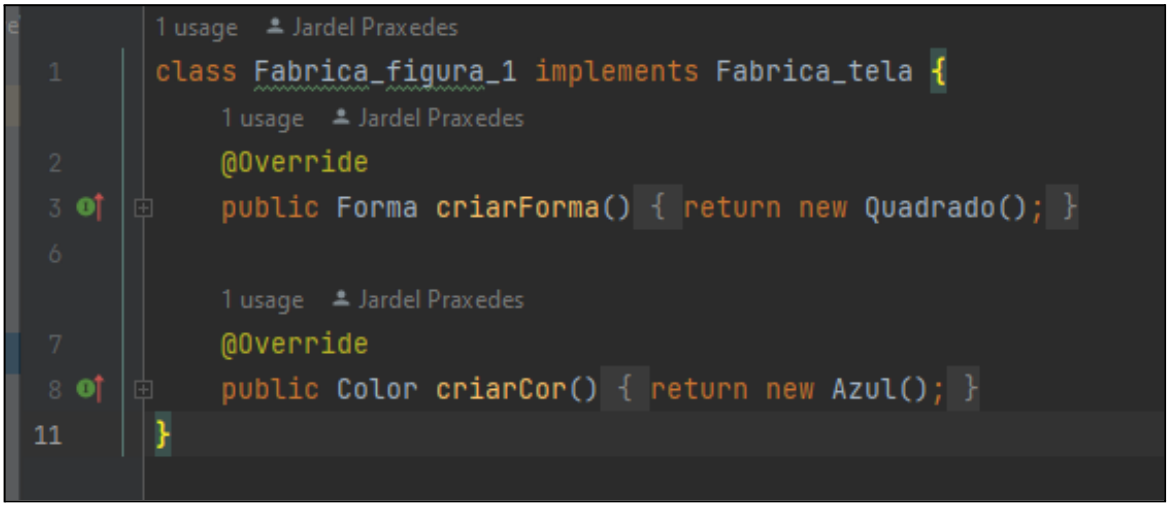
Como se pode notar, um desses métodos é responsável pela forma do objeto e outro pelo preenchimento de sua cor. A interface Fabrica_tela, sendo assim, acaba se relacionando com as interfaces Forma e Color já que ambas são retornos de seus métodos.

No que diz respeito a essas duas interfaces, ambas possuem apenas um método: void desenhar() em Forma e void preenchimento() em Color.

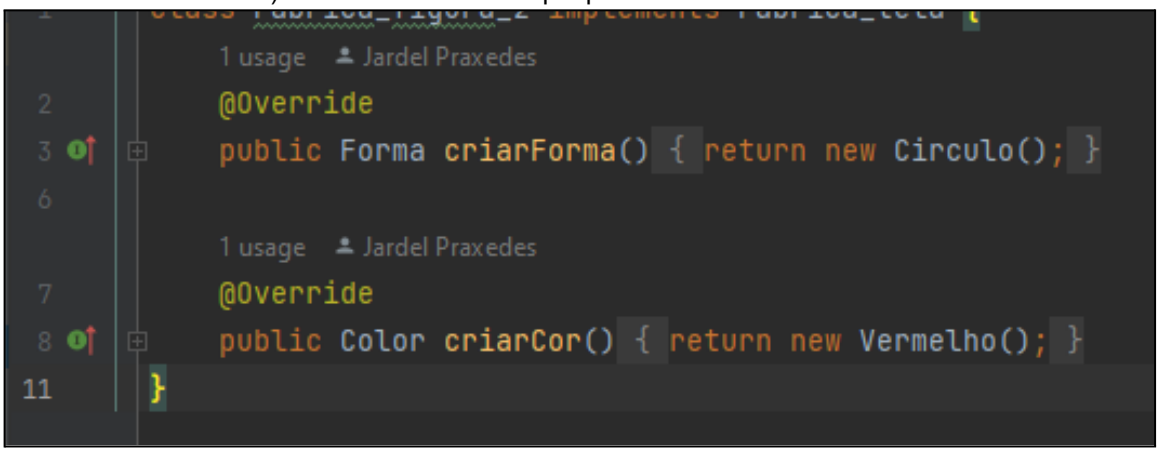
A partir dessas interfaces iniciais, pode-se montar as classes concretas (Figura 2). Acerca das classes concretas vinculadas a Fabrica_tela, tem-se duas: Fabrica_figura_1 e Fabrica_figura_2, responsáveis pela criação de quadrados azuis e círculos vermelhos respectivamente.

Figura 2 - Implementação da Fabrica_tela em classes abstratas.

a) Criando uma fábrica que produz quadrados azuis.



b) Criando uma fábrica que produz círculos vermelhos.

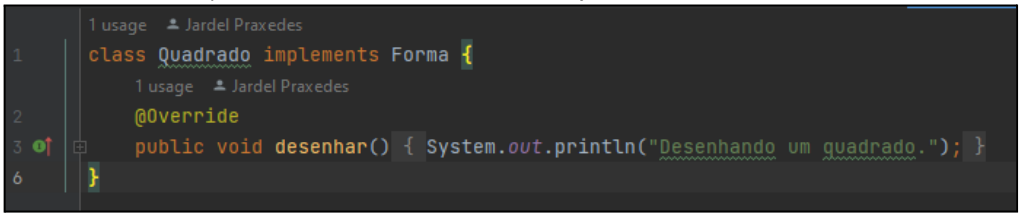


Fonte: própria, 2023.

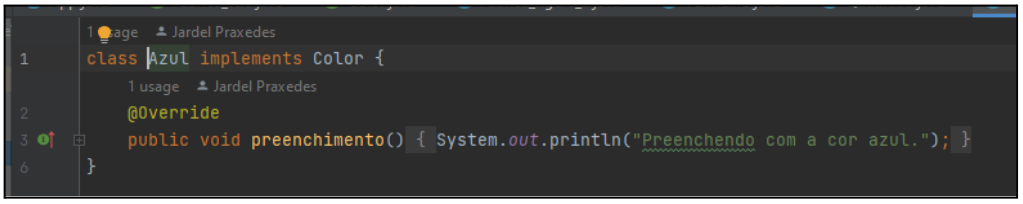
Observando atentamente, pode-se reparar que há presença de quatro objetos antes não debatidos: Quadrado, Azul, Círculo e Vermelho. Essas são classes concretas das interfaces Forma (Quadrado e Círculo) e Color (Azul e Vermelho). A figura 3 mostra o código de implementação de tais classes.

Figura 3 - Classes concretas de Forma e Color.

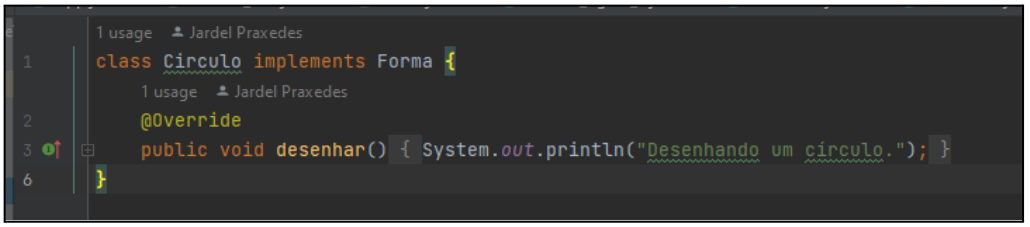
a) Classe concreta Quadrado implementando Forma.



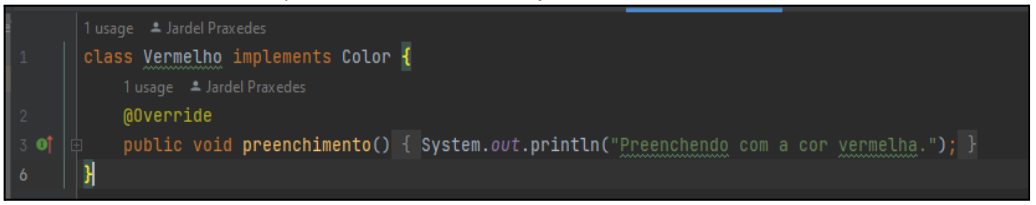
b) Classe concreta Azul implementando Color.



c) Classe Círculo implementando Forma.



d) Classe Vermelho implementando Color



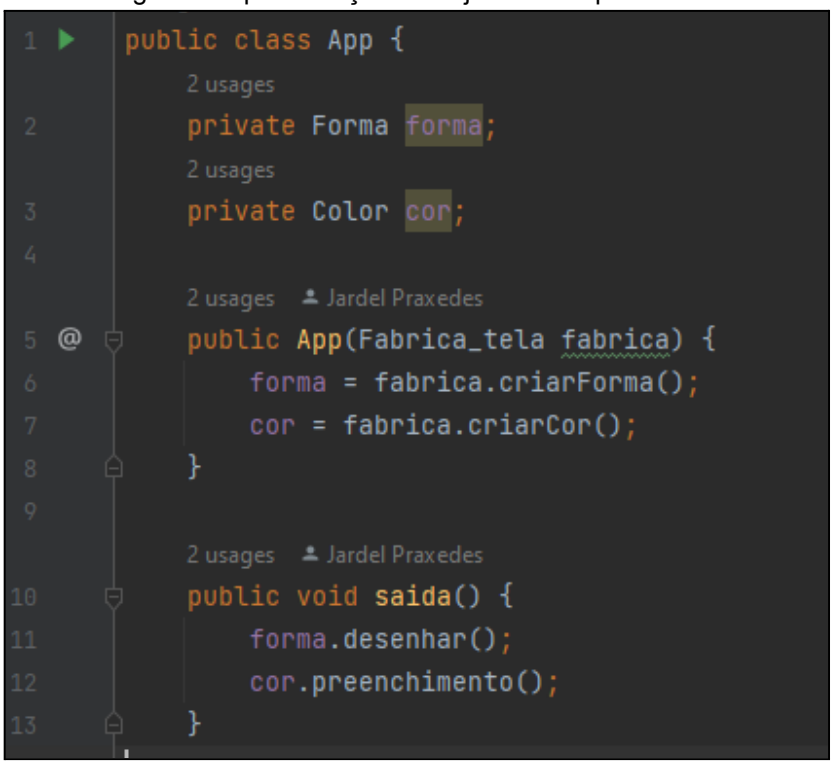
Fonte: própria, 2023.

Com essa base de interfaces e classes concretas montada, consegue-se agora realizar algumas criações de objeto em um método Main - que, nesse caso, está na classe App.

Essa classe principal, que roda o programa, pode ser dividida em duas seções principais: a de preparação, configuração, do padrão de projeto e a do método main.

Sobre essa primeira parte, conforme figura 4, há criação de dois objetos privados chamados *forma* e *cor* do tipo Forma e Color - nessa ordem. Depois, o construtor da classe App é declarado explicitamente e este recebe a fábrica modelo (Inteface Fabrica_tela) e tais objetos criados anteriormente, *forma* e *cor*, recebem atribuição de acordo com os métodos da fábrica recebida como parâmetro. Por último, tem-se o método saida(), que está junta as informações do objeto para ser apresentado para o usuário no console.

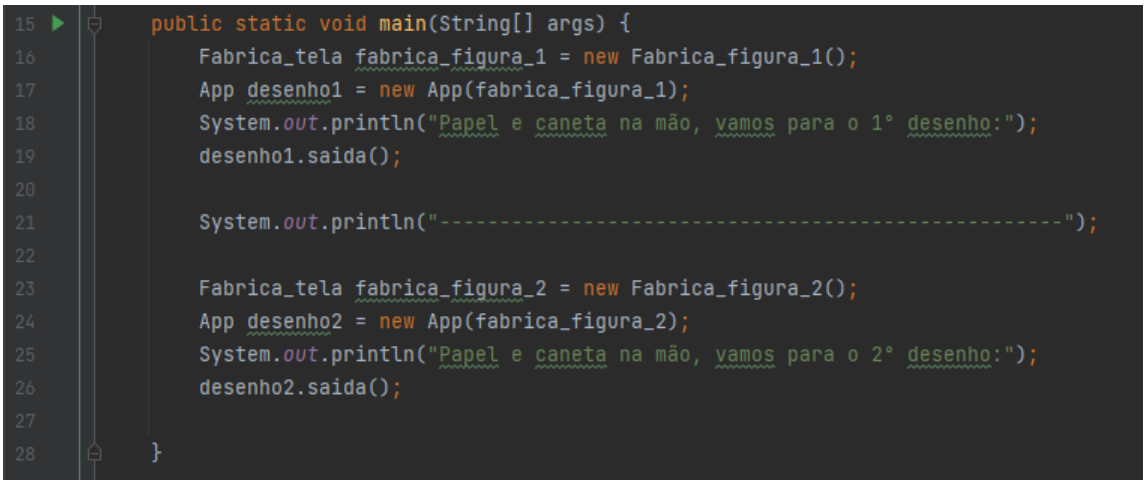
Figura 4 - Primeira parte da classe principal App: atribuição de forma e cor; definição do construtor; e montagem de apresentação do objeto criado para o usuário.



Fonte: própria, 2023.

Com tudo isso estabelecido, o método main é responsável apenas por organizar a linha de raciocínio (Figura 5). Então, primeiramente, cria-se o objeto do tipo da classe concreta que se deseja ser produzido, usando Fabrica_tela (a interface raiz do Abstract Factory) como guia. Depois de criado esse objeto, por exemplo fabrica_figura_1, cria-se um objeto do tipo App que recebe em seu construtor essa fábrica criada anteriormente. E, por último, já sabendo qual objeto deve ser criado, chama-se o método saida() do objeto do tipo App, que monta para apresentação final ao usuário.

Figura 5 - Funcionamento do método main localizado na classe App. Observa-se a conexão presente com todos as outras partes desenvolvidas previamente.



```
15  public static void main(String[] args) {
16      Fabrica_tela fabrica_figura_1 = new Fabrica_figura_1();
17      App desenho1 = new App(fabrica_figura_1);
18      System.out.println("Papel e caneta na mão, vamos para o 1° desenho:");
19      desenho1.saida();
20
21      System.out.println("-----------------------------------------------------");
22
23      Fabrica_tela fabrica_figura_2 = new Fabrica_figura_2();
24      App desenho2 = new App(fabrica_figura_2);
25      System.out.println("Papel e caneta na mão, vamos para o 2° desenho:");
26      desenho2.saida();
27
28  }
```

Fonte: própria, 2023.

3 CONCLUSÃO

O Abstract Factory é uma ferramenta poderosa para criar famílias de objetos relacionados de forma flexível. Ele promove a abstração, encapsulamento e consistência na criação de objetos, trazendo benefícios como facilidade de expansão e manutenção do sistema. Ainda que tal padrão possa aumentar a complexidade do código, especialmente em projetos menores, suas vantagens superam as desvantagens quando aplicado corretamente em sistemas complexos.

No exemplo prático apresentado, a fábrica de criação de figuras diferentes, consegue-se ter uma ideia geral de seu funcionamento e suas características. Ao compreender e aplicar adequadamente o padrão Abstract Factory, os desenvolvedores de software podem criar sistemas mais robustos, flexíveis e fáceis de manter, garantindo a consistência e a eficiência na criação de objetos relacionados.

REFERÊNCIAS

ABSTRACT Factory. **Refactoring Guru**, 2023. Disponível em: <https://refactoring.guru/pt-br/design-patterns/abstract-factory>. Acesso em: 21 de maio de 2023.

BIGARDI, Gustavo B. Arquitetura e desenvolvimento de software - Parte 2 - Abstract Factory. **Medium**, 2023. Disponível em: < https://gbbigardi.medium.com/arquitetura-e-desenvolvimento-de-software-parte-2-abstract-factory-f603ccc6a1ea>. Acesso em: 21 de maio de 2023.

PADRÃO Abstract Factory. **DevMedia**, 2023. Disponível em:<https://www.devmedia.com.br/padrao-abstract-factory/23030>. Acesso em: 21 de maio de 2023.

THIENGO, Vinícius. Padrão de Projeto: Abstract Factory. **Thiengo**, 2023. Disponível em:<https://www.thiengo.com.br/padrao-de-projeto-abstract-factory>. Acesso em: 21 de maio de 2023.