ZEUS
AUTOMAÇÃO COMERCIAL

FACTORY METHOD

Apresentado por: Vinicius Cunha

Design Patterns

- Design patterns ou padrões de design são soluções já testadas para problemas recorrentes no desenvolvimento de software, que deixam seu código mais manutenível e elegante, pois essas soluções se baseiam em um baixo acoplamento.

- **Tipos de padrões:**

  Criacionais: Fornecem meios para criar objetos, encapsulando a lógica de criação deles.

  Estruturais: Demonstram como montar objetos em estruturas maiores, sem perder a eficiência e flexibilidade.

  Comportamentais: Ajudam a trabalhar melhor com os algoritmos e com a delegação de responsabilidades entre os objetos.

Benefícios

- São modelos que já foram utilizados e testados, portanto podem representar um bom ganho de produtividade para os desenvolvedores.

- Seu uso também contribui para a organização e manutenção de projetos, já que esses padrões se baseiam em baixo acoplamento entre as classes e padronização do código.

- Com a padronização dos termos, as discussões técnicas são facilitadas. É mais fácil falar o nome de um design pattern em vez de ter que explicar todo o seu comportamento.

Introdução

- O Factory Method pertence a classe padrão criacional, cuja responsabilidade é de instanciação de objetos.

- De acordo com o livro Design Patterns: Elements of Reusable Object-Oriented Software o factory method trata-se de: "Definir uma interface para criar um objeto, mas deixar as subclasses decidirem que classe instanciar. O Factory Method permite adiar a instanciação para as subclasses.".

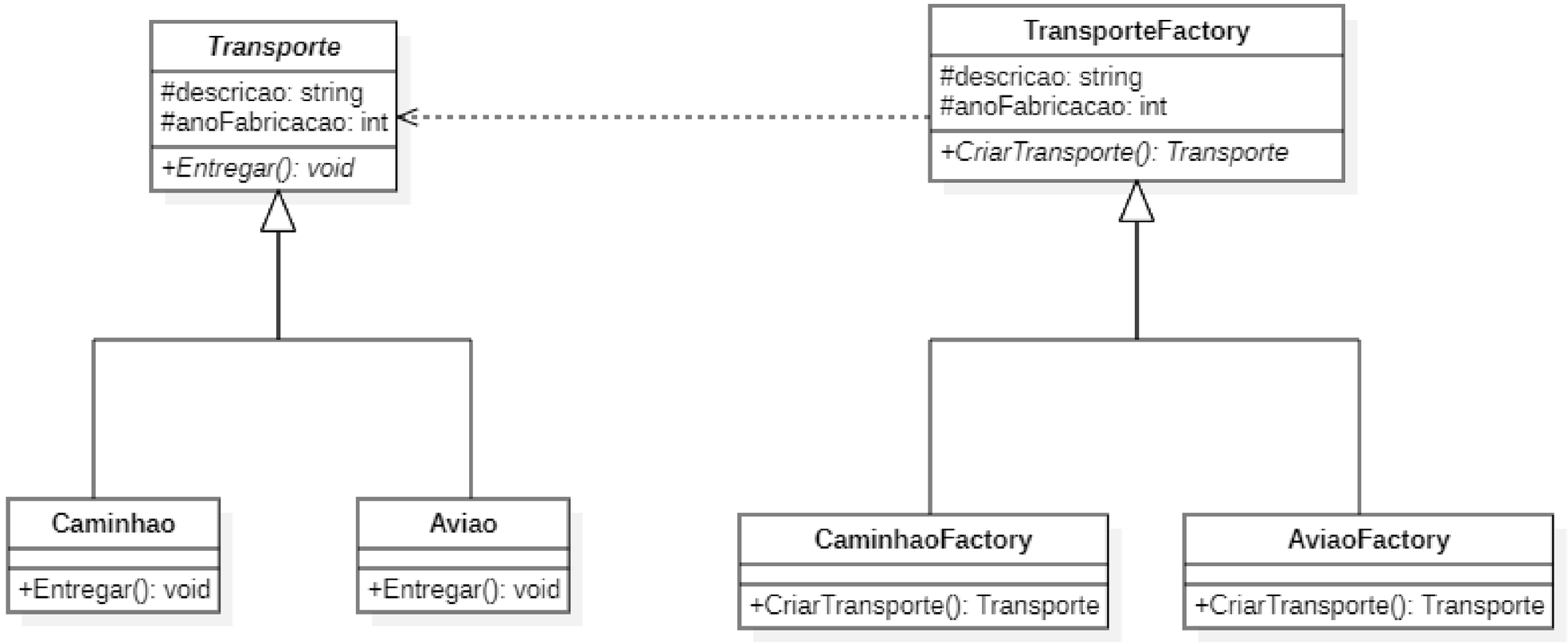
Problema

- Imagine uma aplicação de uma transportadora. A primeira versão da aplicação pode lidar apenas com o transporte via caminhões, portanto a maior parte do seu código está acoplado à classe Caminhão.

- Após um tempo, é necessária a expansão do aplicativo para conter envio por aviões. Isso seria ótimo para a empresa, mas como o aplicativo faria isso de maneira rápida? Já que o acoplamento à classe Caminhão é grande.

Diagrama

Figura 1: Diagrama da implemetação



Fonte: Autor

Prós

- Você evita acoplamentos firmes entre o criador e os transportes concretos.
- Cumpre o Princípio de responsabilidade única. Permitindo mover o código de criação do transporte para um único local do programa, facilitando a manutenção do código.
- Cumpre o Princípio aberto/fechado. Onde é possível introduzir novos tipos de transporte no programa sem quebrar o código cliente existente.

Contras

- O código pode se tornar mais complexo, pois é necessário introduzir muitas subclasses novas para implementar o padrão.

Referências

- Shvets, Alexander. Factory method. Refactoring.Guru, 2014. Disponível em: <https://refactoring.guru/pt-br/design-patterns/factory-method>. Acesso em: 23, setembro e 2022.

ZEUS
AUTOMAÇÃO COMERCIAL

Obrigado!