**Padrão Factory Method**

**Definição:** O Factory Method é um padrão de projeto criacional que fornece uma interface para a criação de objetos em uma classe, mas delega a decisão de quais objetos serão criados para subclasses. Ele é parte dos padrões de projeto descritos no livro "Design Patterns: Elements of Reusable Object-Oriented Software".

**Principais Características:**

* **Encapsula a Criação de Objetos:** Centraliza a criação de objetos em um local, facilitando a manutenção.
* **Desacoplamento:** O código cliente interage com uma interface ou classe abstrata, permitindo flexibilidade e extensibilidade.
* **Uso de Subclasses:** As subclasses concretas determinam o objeto a ser criado, promovendo a reutilização do código.

**Aplicação Prática:** Utilizado em sistemas onde há necessidade de flexibilidade e extensibilidade na criação de objetos, especialmente quando o tipo de objeto pode variar de acordo com o contexto.

**Exemplo em Java: Sistema de Transporte**

**Estrutura do Código:**

1. **Superclasse Transporte:** Classe abstrata que define a interface **criarTransporte()**.
2. **Subclasses Aviao, Caminhao, Navio:** Implementam a superclasse **Transporte**, sobrescrevendo o método **criarTransporte()** para criar instâncias específicas de transporte.
3. **Classe TransporteFactory:** Fábrica que cria instâncias de transportes com base em um parâmetro de entrada.
4. **Classe Main:** Ponto de entrada do programa, onde a **TransporteFactory** é usada para criar diferentes tipos de transportes.

**Funcionalidade do Código:**

* **Abstração e Flexibilidade:** A superclasse **Transporte** fornece uma abstração, enquanto as subclasses concretizam esta abstração, permitindo flexibilidade na criação de diferentes tipos de transportes.
* **Desacoplamento:** O código cliente (na classe **Main**) não precisa conhecer as classes concretas de transporte, apenas a interface **Transporte** e a **TransporteFactory**.
* **Extensibilidade:** Novos tipos de transporte podem ser facilmente adicionados, estendendo a superclasse **Transporte**.

**Vantagens do Padrão:**

* **Reutilização de Código:** Evita duplicação e facilita a manutenção ao centralizar a criação de objetos.
* **Princípio da Responsabilidade Única:** Separa a lógica de criação de objetos da lógica de negócios.
* **Substituição Facilitada:** Facilita a introdução de novos tipos de transporte sem alterar o código existente.