Documentação dos Padrões de Projeto: Observer e Factory Method

Padrão Observer

O **Observer** é um padrão de design comportamental que permite que um objeto (`Sujeito`) notifique automaticamente uma lista de outros objetos (`Observadores`) quando ocorre uma mudança em seu estado.

Como funciona?

1. **Sujeito (Subject):**
    a. É o objeto principal que possui um estado que pode mudar.
    b. Mantém uma lista de Observadores inscritos.
    c. Quando seu estado muda, ele notifica todos os Observadores.
2. **Observadores (Observers):**
    a. São objetos interessados em acompanhar as mudanças no Sujeito.
    b. Se inscrevem no Sujeito para receber notificações.
    c. Implementam um método de atualização que é chamado pelo Sujeito.

Benefícios:

- **Desacoplamento:** O Sujeito não precisa conhecer os detalhes dos Observadores; ele apenas envia notificações.
- **Facilidade de Manutenção:** Novos Observadores podem ser adicionados ou removidos sem alterar o código do Sujeito.
- **Reutilização de Código:** Permite reutilizar o Sujeito com diferentes Observadores.

Adaptei o padrão de design para criar um sistema de LOG de notícias, favorecendo a comunicação entre diferentes partes do sistema sobre a alteração do estado do objeto.

Padrão Factory Method

O **Factory Method** é um padrão de design criacional que fornece uma interface para criar objetos, mas permite que as subclasses decidam qual classe instanciar. Ou seja, a responsabilidade de criar objetos é delegada para subclasses.

Como funciona?

- **Classe Criadora (Creator):** Define um método fábrica que retorna objetos do tipo produto.
- **Subclasses Concretas:** Implementam ou substituem o método fábrica para criar produtos específicos.

Benefícios:

- **Isolamento de Código:** O código que usa os objetos (código cliente) não precisa conhecer as classes concretas dos objetos que está usando.
- **Facilidade de Extensão:** Novos tipos de produtos podem ser adicionados facilmente sem modificar o código existente.
- **Reutilização de Código:** A lógica comum de criação pode ser centralizada na classe criadora.

  Utilizei esse padrão de designer otimizar o processo de criação de diferentes tipos de objetos derivados da classe ‘Noticia’ concentrando a lógica de criação dentro da classe fábrica.