Introdução

Neste documento, vou explorar a escolha dos padrões de projeto **Strategy, Factory** e **Facade** para implementar um sistema de gerenciamento de contatos. Vou abordar as razões por trás dessas escolhas e como cada padrão é aplicado no contexto do projeto.

1. Strategy Pattern (Padrão Comportamental)

Descrição: O Strategy Pattern é um padrão comportamental que permite definir uma família de algoritmos, encapsulando cada um deles e tornando-os intercambiáveis. Ele permite que o algoritmo varie independentemente dos clientes que o utilizam.

Justificativa: Escolhi o Strategy Pattern para implementar a funcionalidade de busca de contatos. Por quê?

- **Flexibilidade:** O padrão Strategy me permite variar a estratégia de busca sem modificar o código existente. Se, no futuro, desejarmos adicionar novos algoritmos de busca (por exemplo, busca por nome, busca por telefone, busca por email), podemos fazê-lo facilmente, criando novas classes de estratégia sem afetar o restante do sistema.
- **Manutenção:** Separar a lógica de busca em estratégias independentes facilita a manutenção e a evolução do código. Cada estratégia é responsável por uma única tarefa, tornando o sistema mais modular e compreensível.

2. Factory Pattern (Padrão Criacional)

Descrição: O Factory Pattern é um padrão criacional que permite delegar a criação de objetos para subclasses. Ele define uma interface para criar objetos, mas permite que as subclasses decidam qual classe concreta instanciar.

Justificativa: Escolhi o Factory Pattern para a criação e gerenciamento de objetos do tipo Contato. Por quê?

- **Abstração da Criação:** O Factory Pattern me permite encapsular a lógica de criação de objetos. Se precisarmos adicionar novos tipos de contatos (por exemplo, ContatoProfissional, ContatoPessoal), podemos estender a fábrica sem modificar o código existente.
- **Desacoplamento:** O Factory Pattern desacopla o código cliente da implementação concreta. O cliente pode criar objetos de contato sem se preocupar com os detalhes específicos da criação.

3. Facade Pattern (Padrão Estrutural)

Descrição: O Facade Pattern é um padrão estrutural que fornece uma interface simplificada para um conjunto complexo de classes. Ele atua como uma fachada (ou interface única) para ocultar a complexidade interna e fornecer uma maneira mais fácil de interagir com o sistema.

Justificativa: Escolhi o Facade Pattern para simplificar a interação com o sistema de gerenciamento de contatos. Por quê?

- **Abstração da Complexidade:** O Facade Pattern permite que os clientes interajam com o sistema através de uma interface única, ocultando os detalhes internos e complexos. Isso simplifica o uso e melhora a legibilidade do código.
- **Organização e Coesão:** Ao agrupar funcionalidades relacionadas em uma fachada, podemos manter uma estrutura organizada e coesa. Isso facilita a manutenção e evita a dispersão de lógica em várias partes do sistema.

Problema:

Objetivo: Deve-se aplicar padrões de projeto em JavaScript, tanto estruturais quanto comportamentais em um contexto prático.

Problema: Você foi contratado para desenvolver um sistema de gerenciamento de contatos em JavaScript. O sistema deve permitir aos usuários adicionar, remover e listar contatos. Além disso, eles devem ser capazes de buscar contatos por nome.

Requisitos Iniciais:

4. Implemente uma classe `Contato` que represente um contato com os seguintes atributos: nome, telefone e email.
5. Implemente uma classe `GerenciadorContatos` que possua métodos para adicionar, remover e listar contatos. Este gerenciador deve manter uma lista de contatos.
6. Implemente uma interface simples de linha de comando (CLI) que permita aos usuários interagirem com o sistema (adicionar, remover, listar e buscar contatos).
7. Utilize um padrão de projeto estrutural para garantir que a adição e remoção de contatos possam ser facilmente estendidas no futuro, sem modificar o código existente.
8. Utilize um padrão de projeto comportamental para implementar a funcionalidade de busca de contatos, garantindo que esta possa variar independentemente dos algoritmos de busca.

Instruções Adicionais:

- Você pode escolher qualquer padrão de projeto estrutural e comportamental para implementar os requisitos.
- Documente o seu código de forma clara, explicando a escolha dos padrões de projeto utilizados.
- Siga as melhores práticas de desenvolvimento JavaScript e padrões de codificação.
- Teste o seu sistema para garantir que ele atende aos requisitos especificados.