Perguntas para Reflexão

Após completar a implementação, responda:

1. **Qual é o papel do ComboFacade neste sistema?**
    - Como ele simplifica a vida do cliente (Main)?
    - Main só chama criarCombo, exibirItens e getPrecoTotal — não precisa conhecer classes internas nem detalhes de construção dos objetos.
    - Menos acoplamento e código no cliente; responsabilidade única em Main.
2. **Por que usamos a interface ItemCombo?**
    - Que benefícios isso traz para o design?

      Polimorfismo e baixo acoplamento.

      Facilita extensão: novas classes de item implementam ItemCombo sem alterar clientes.

      Melhora testabilidade e manutenção.
3. **Compare com o padrão Singleton:**
    - Poderia o ComboFacade também ser um Singleton? Por quê?
    - Sim — tecnicamente é possível tornar ComboFacade um Singleton para compartilhar estado/fonte única de criação.
    - Em que situação isso seria útil?
    - se houver recursos compartilhados/configuração global que precisam ser centralizados (por exemplo, caching, acesso a serviços externos).
4. **Identifique os subsistemas:**
    - Quais são as classes que representam a "complexidade" escondida?
    - As classes que representam a complexidade escondida são a interface br.fastfood.model.ItemCombo e suas implementações br.fastfood.model.Burger, br.fastfood.model.Bebida e br.fastfood.model.Sobremesa, além da classe

br.fastfood.model.Combo que agrega os itens e calcula o preço; o ComboFacade apenas encapsula a criação e uso dessas classes para o cliente.

- O que aconteceria se o cliente (Main) tivesse que criar cada item individualmente?
- Se o Main tivesse que criar cada item individualmente, ele passaria a conhecer preços e composições, gerando acoplamento maior, duplicação de código, responsabilidade excessiva (UI + regras de negócio), maior propensão a erros e dificuldade de manutenção e testes sempre que um combo ou preço mudasse.
-

5. **Extensibilidade:**
   - Como você adicionaria um novo combo (Combo 4) ao sistema?
   - Adicionar o novo case em ComboFacade.criarCombo com os itens e preços.
   - Atualizar o menu em Main.java (mostrar opção 4).
   - Que classes precisariam ser modificadas?
   - ComboFacade e Main
6. **Validação e Tratamento de Erros:**
   - Como o sistema trata um código de combo inválido?
   - O tratamento ocorre em duas camadas: na interface (Main) entradas fora de 0–3 geram " Opção inválida!"; no domínio, ComboFacade imprime "Código de combo inválido!" e define combo = null, de modo que exibirItens mostra "Nenhum combo criado ainda!" e getPrecoTotal retorna 0.0 — evitando NullPointer e mantendo o sistema em um estado previsível.
   - Que melhorias você sugere?
   - Remova prints do ComboFacade e faça criarCombo retornar um resultado claro (boolean/Optional ou lançar exceção), use enums ou configuração externa para definir combos e preços, evite nulls preferindo Optional, trate validação de entrada no Main e exceções de Scanner, padronize nomes de classes/arquivos, substitua System.out por logging, e escreva testes unitários e documentação;

essas mudanças reduzem acoplamento, aumentam a testabilidade e facilitam manutenção.