Relatório - Sistema de Pedidos para Lanchonete

Gabriel da Silva Reboli

CC4M

1 Introdução

Este relatório descreve o desenvolvimento de um sistema de gerenciamento de pedidos para uma lanchonete fictícia. O sistema foi desenvolvido com o objetivo de permitir a personalização de hambúrgueres utilizando o padrão de projeto *Decorator* e a notificação de observadores por meio do padrão *Observer*. Além disso, o cliente pode adicionar itens extras ao pedido, como bebidas e sobremesas.

2 Padrões de Projeto Utilizados

Dois padrões de projeto foram implementados neste sistema: *Decorator* e *Observer*. Cada um foi utilizado de acordo com as necessidades descritas no enunciado do trabalho.

2.1 Decorator

O padrão *Decorator* foi aplicado para permitir a personalização dos hambúrgueres. O hambúrguer base é uma instância de uma classe que implementa a interface `IHamburguer`. Ingredientes adicionais, como bacon, queijo, alface e tomate, foram implementados como decoradores que adicionam uma nova descrição e custo ao hambúrguer base.

- A interface `IHamburguer` define dois métodos principais: `GetDescricao()` e `GetCusto()`.
- A classe `HamburguerSimples` implementa a interface `IHamburguer` e representa o hambúrguer básico.
- As classes de ingredientes, como `Bacon`, `Queijo`, `Alface`, e `Tomate`, implementam o padrão *Decorator*, envolvendo o hambúrguer base e adicionando novos comportamentos.

A principal vantagem do padrão *Decorator* é a flexibilidade para adicionar novos ingredientes ao hambúrguer sem modificar a estrutura básica do objeto, permitindo personalizações dinâmicas.

2.2 Observer

O padrão *Observer* foi utilizado para notificar automaticamente a equipe da lanchonete sempre que um pedido é feito ou atualizado. Existem dois observadores principais:

- `MonitorProducao` é responsável por acompanhar os pedidos em produção.
- `MonitorMontagem` é responsável por acompanhar quando um pedido está pronto para montagem.

A classe `Pedido` funciona como o *Sujeito*, mantendo uma lista de observadores e os notificando sempre que há uma mudança no pedido. Os observadores são atualizados automaticamente, recebendo informações sobre o estado atual do pedido e os ingredientes selecionados.

3 Estrutura do Código

A estrutura do código foi organizada de forma a facilitar a adição de novos ingredientes e o controle do fluxo do pedido. Segue uma breve explicação das principais classes e interfaces:

- `IHamburguer`: Interface que define a estrutura dos hambúrgueres e decoradores.
- `HamburguerSimples`: Implementa a interface `IHamburguer`, representando o hambúrguer básico.
- `IngredienteDecorator`: Classe abstrata que implementa a interface `IHamburguer` e serve de base para todos os decoradores.
- `Bacon`, `Queijo`, `Alface`, `Tomate`: Decoradores concretos que adicionam ingredientes ao hambúrguer.
- `Pedido`: Classe que representa o pedido do cliente e implementa o padrão *Observer*, notificando os observadores conforme o status do pedido muda.
- `MonitorProducao` e `MonitorMontagem`: Observadores que são notificados sobre mudanças no pedido, como ingredientes escolhidos e status de produção.
- `ItemPedido`: Representa itens adicionais do pedido (como bebidas e sobremesas).

4 Possíveis Melhorias

Embora o sistema atenda às especificações propostas, algumas melhorias poderiam ser implementadas para aumentar sua robustez e flexibilidade:

- **Persistência de dados**: Implementar a funcionalidade de salvar e carregar pedidos a partir de um arquivo (JSON ou banco de dados), para permitir que os pedidos sejam reabertos ou arquivados.
- **Descontos e promoções**: Adicionar a funcionalidade de aplicar descontos ou promoções ao pedido.
- **Interface Gráfica**: Desenvolver uma interface gráfica (GUI) para facilitar a interação do cliente com o sistema, deixando de ser apenas uma aplicação de console.

5 Conclusão

O sistema de pedidos para a lanchonete foi desenvolvido com sucesso, utilizando os padrões de projeto *Decorator* e *Observer* conforme especificado. A aplicação permite que o cliente personalize seu hambúrguer dinamicamente e, ao mesmo tempo, mantém a equipe da lanchonete informada sobre o progresso dos pedidos em tempo real. O código foi estruturado de maneira modular, facilitando futuras expansões e melhorias.