Relatório sobre a Aplicação de Padrões de Design

Introdução

Este relatório analisa a aplicação de padrões de design em um sistema de pedidos de uma lanchonete, desenvolvido em C#. O sistema permite a criação, visualização e processamento de pedidos, além de incorporar acompanhamentos e notificações sobre o estado dos pedidos. Os principais padrões de design aplicados são o **Padrão Observer** e o **Padrão Decorator**.

Estrutura do Código

O sistema é estruturado em várias classes, cada uma responsável por uma parte específica da lógica de negócios:

1. **Classe `Pedido`**: Esta classe representa um pedido realizado pelo cliente. Contém informações sobre o estado do pedido, o hambúrguer e uma lista de acompanhamentos. A classe implementa o padrão Observer, permitindo que múltiplos monitores (observadores) se inscrevam para receber atualizações sobre o estado do pedido. Os métodos `AddObserver`, `RemoveObserver` e `AvisarMonitores` gerenciam a lista de observadores e notifica-os quando o estado do pedido muda.
2. **Classe `IHamburguer` e Decoradores**: O padrão Decorator é utilizado para permitir a adição dinâmica de acompanhamentos ao hambúrguer. A interface `IHamburguer` define a estrutura básica de um hambúrguer, enquanto classes como `BaconDecorator`, `QueijoDecorator` e outros decoradores estendem a funcionalidade, acrescentando itens ao pedido de forma flexível.
3. **Classes de Monitores**: As classes `MonitorProducao`, `MonitorMontagem` e `MonitorBalcao` implementam a interface `IObserver`, permitindo que elas reagem a mudanças no estado do pedido. Cada monitor possui lógica específica para executar ações quando um pedido atinge determinados estados, como "Pronto para Montar" ou "Pronto para Entrega".
4. **Classe `Program`**: Esta classe contém o método `Main`, que é o ponto de entrada do aplicativo. O loop de menu permite aos usuários criar novos pedidos, visualizar pedidos existentes e processar pedidos. A interface é simples e intuitiva, facilitando a interação do usuário com o sistema.

Aplicação de Padrões de Design

Padrão Observer

O padrão Observer é implementado na classe `Pedido`, onde a lista de observadores é gerenciada. O método `AvisarMonitores` itera sobre os observadores e os notifica com o estado atual do pedido. Esse padrão permite que diferentes partes do sistema reagem a mudanças de estado sem necessidade de acoplamento rígido, promovendo uma arquitetura mais flexível.

Padrão Decorator

O padrão Decorator é utilizado para permitir a personalização dos hambúrgueres com diferentes acompanhamentos. A interface `IHamburguer` define um contrato que as classes de hambúrguer devem seguir, enquanto as classes decoradoras estendem essa funcionalidade. Isso permite que os usuários adicionem acompanhamentos de forma dinâmica e modular, sem modificar a estrutura existente do código.

Possíveis Melhorias

1. **Validação de Entradas**: Implementar verificações para garantir que as entradas do usuário sejam válidas, reduzindo o risco de erros e melhorando a robustez do sistema.
2. **Gerenciamento de Estado**: Centralizar a lógica de alteração de estado dentro da classe `Pedido` poderia melhorar a consistência na atualização de estados e a notificação dos observadores.
3. **Feedback ao Usuário**: Adicionar mais feedback e confirmações durante a execução do programa pode enriquecer a experiência do usuário.
4. **Persistência de Dados**: Considerar a implementação de uma solução de persistência, como um banco de dados, para armazenar os pedidos e suas informações.
5. **Uso de Enum para Estados**: O uso de `enums` para representar os diferentes estados do pedido pode melhorar a legibilidade e manutenibilidade do código.

Conclusão

A aplicação dos padrões de design Observer e Decorator no sistema de pedidos demonstrou uma arquitetura eficaz e modular, permitindo a adição de funcionalidades e a flexibilidade na notificação de estados. Apesar das boas práticas adotadas, existem oportunidades de melhorias que poderiam fortalecer a robustez e a usabilidade do sistema. A adoção de um design mais centrado no usuário e a melhoria no gerenciamento de estados e entradas elevarão a qualidade geral do software.