Instituto Federal de Santa Catarina - Câmpus Gaspar
Análise e Desenvolvimento de Sistemas
Padrões de Projeto de Software
Prof. Dr. Renato Simões Moreira
Aluno: Túlio Gabriel Bitencourt Jorge

Avaliação Prática

Introdução

Neste projeto, foram implementadas demonstrações de quatro padrões de design: Builder, Singleton, State e Facade. Cada um desses padrões resolve problemas específicos de design, promovendo um código mais limpo, modular e fácil de manter.

1. Builder

O Builder é um padrão de design criacional que permite a criação de objetos complexos passo a passo. Este padrão separa a construção de um objeto de sua representação, permitindo diferentes representações ou construções parciais.

O padrão Builder é aplicável quando:

- A criação de um objeto envolve muitas etapas ou parâmetros.
- O processo de criação precisa ser flexível para acomodar diferentes representações.
- Deseja-se melhorar a legibilidade do código na criação de objetos complexos.

Neste projeto, o padrão Builder foi utilizado para a construção da classe *Quarto*. A classe *Quarto* possui diversos atributos, como número do quarto, capacidade máxima, vista para o mar e sacada. Usando o Builder, conseguimos criar objetos de *Quarto* de forma clara e flexível, sem a necessidade de múltiplos construtores sobrecarregados ou métodos estáticos complexos. Este padrão também permite adicionar novas características de forma transparente sem afetar o código existente.

2. Singleton

O Singleton é um padrão de design criacional que garante que uma classe tenha apenas uma instância e fornece um ponto de acesso global a essa instância.

O padrão Singleton é aplicável quando:

- Deve existir apenas uma instância de uma classe e ela deve ser acessível a partir de um ponto global.
- É necessário controlar o acesso a recursos compartilhados.

O padrão Singleton foi implementado na classe *GerenciadorDeReservaSingleton* para garantir que apenas uma instância do gerenciador de reservas exista em todo o aplicativo. Isso é

crucial, pois permite o gerenciamento centralizado de quartos e hóspedes, evitando conflitos de estado ou duplicação de dados. O Singleton também facilita o controle do acesso aos recursos compartilhados entre diferentes partes do aplicativo.

3. State

State é um padrão de design comportamental que permite que um objeto altere seu comportamento quando seu estado interno muda.

O padrão State é aplicável quando:

- O comportamento de um objeto muda dependendo do seu estado.
- Há muitas condições condicionais baseadas no estado do objeto.

No projeto, o padrão State foi usado para gerenciar o estado dos quartos. Os quartos podem estar em diferentes estados, como Disponível, Reservado, Ocupado e Manutenção. Cada estado possui seu próprio comportamento, por exemplo, apenas quartos disponíveis podem ser reservados, e quartos em manutenção não podem ser reservados ou ocupados. Implementando esses comportamentos em classes de estado separadas, podemos adicionar ou modificar estados facilmente sem alterar a lógica principal da classe *Quarto*.

4. Facade

Facade é um padrão de design estrutural que fornece uma interface simplificada para um conjunto de interfaces de um subsistema, tornando-o mais fácil de usar.

O padrão Facade é aplicável quando:

- Um sistema tem uma estrutura complexa, e é necessário simplificar o acesso a ele.
- Deseja-se reduzir as dependências de um sistema por meio de uma interface simples.
- Há necessidade de organizar melhor o código, encapsulando interações complexas em uma única classe.

O padrão foi implementado na classe *ReservaFacade* para simplificar o processo de reserva de quartos e gerenciamento de hóspedes. Através desta fachada, é possível verificar a disponibilidade de quartos, criar reservas e gerenciar outras operações comuns sem precisar interagir diretamente com as classes subjacentes. Isso reduz a complexidade para o usuário final e desacopla o código de negócios da lógica de aplicação, facilitando a manutenção e evolução do sistema.