Interface gráfica do usuário, Texto

Descrição gerada automaticamente com confiança médiaMINISTÉRIO DA EDUCAÇÃO

SECRETARIA DE EDUCAÇÃO PROFISSIONAL, TÉCNICA E TECNOLÓGICA

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA CÂMPUS GASPAR

**Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas**

**Disciplina:** Padrões de Projeto de Software;

**Discentes:** Jéssica Kraus **Matricula:** 202110808905;

Nadia Naely Leitão Castro **Matricula:** 202110806719.

**Professor:** Renato Simões Moreira.

**Avaliação 1**

**Singleton**

Este padrão de projetos é utilizado para garantir que uma classe possua apenas uma instância e assim permitindo o acesso global. No domínio ‘Pousada’, o padrão Singleton pode ser aplicado à classe "Pousada" para assegurar que exista apenas uma instância da pousada no sistema.

Na implementação desse padrão, a classe "Pousada" possui um atributo estático chamado "instancia" que armazena uma única instância da pousada. Para evitar a criação direta de instâncias da classe, o construtor é definido como privado.

O método estático "getInstance()" é responsável por retornar a instância única da pousada. Na primeira chamada desse método, é verificado se a instância já foi criada. Caso não tenha sido, uma nova instância é criada e atribuída ao atributo "instancia". Nas chamadas subsequentes, a mesma instância é retornada, garantindo a consistência e o acesso global à instância da pousada.

Essa abordagem permite controlar e gerenciar efetivamente a criação da instância da pousada em todo o sistema, evitando múltiplas instâncias e fornecendo um ponto centralizado de acesso à mesma.

**Builder**

Este padrão de projetos é utilizado para facilitar a criação de objetos complexos, permitindo a construção passo a passo com configurações opcionais. No domínio da ‘Pousada’, podemos aplicar o padrão Builder para a criação de objetos de reserva de quartos.

Nesse exemplo, a classe "Reserva" possui um construtor privado que recebe um objeto do tipo "Builder". O Builder é uma classe interna estática que contém os mesmos atributos da classe "Reserva" e métodos para definir esses atributos de forma incremental.

A classe "ReservaBuilder" fornece métodos como "quarto()", "dataInicio()" e "dataFim()" para configurar cada atributo da reserva. Esses métodos retornam a própria instância do Builder, permitindo que as chamadas sejam encadeadas. O método "build()" é responsável por criar e retornar uma instância de "Reserva", utilizando os valores configurados no Builder.

Dessa forma, o padrão Builder facilita a construção de objetos complexos, como a reserva de quartos em uma pousada, permitindo que os atributos sejam configurados de forma opcional e em qualquer ordem, melhorando a legibilidade e a flexibilidade do código.

**Factory**

O padrão Factory é útil para criar objetos de diferentes tipos de quartos, permitindo que você os instancie de forma mais flexível e encapsulada.

Nesse exemplo, temos uma classe abstrata "Quarto" que define os atributos comuns a todos os tipos de quartos e declara um método abstrato "exibirDetalhes()" para ser implementado nas classes concretas.

As classes " QuartoDeluxe", “QuartoMaster” e " QuartoStandard" são subclasses de "Quarto" que implementam o método "exibirDetalhes()" de acordo com as características específicas de cada tipo de quarto.

A classe "QuartoFactory" é a fábrica responsável pela criação dos objetos de quartos. Ela possui um método estático "criarQuarto()" que recebe o tipo e o número do quarto como parâmetros. Com base no tipo fornecido, a fábrica decide qual classe concreta deve ser instanciada e retorna o objeto correspondente.

Dessa forma, o padrão Factory permite a criação dinâmica de instâncias de diferentes tipos de quartos, encapsulando a lógica de criação.