Documentação dos Padrões de Projeto no Sistema de Gestão Hospitalar Introdução

Brena monike Laurindo - 202220304920 * Edson Luiz Zuchi - 202220302295 * Mateus da Silva Reinert - 202010005320

Este documento tem como objetivo apresentar a documentação dos três padrões de projeto utilizados no sistema de gestão hospitalar. Os padrões utilizados são: **Singleton**, **Builder** e **Prototype**. Através da análise dos métodos e códigos presentes nas classes Hospital, Doctor e Nurse, vamos descrever a aplicabilidade de cada padrão e sua justificativa de uso.

1. **Padrão Singleton**
O padrão Singleton é utilizado na classe Hospital para garantir que apenas uma instância da classe seja criada e fornecer um ponto global de acesso a essa instância. O objetivo é evitar a criação de múltiplas instâncias do hospital e garantir que todas as partes do sistema acessem a mesma instância.

Justificativa de uso:

Em um sistema de gestão hospitalar, é desejável ter apenas uma instância do hospital em execução, pois todos os dados e recursos relacionados ao hospital devem ser compartilhados e consistentes.
O uso do padrão Singleton garante que as informações dos médicos, enfermeiros e demais entidades relacionadas sejam acessadas de forma coerente em todo o sistema.

Através do método estático *getInstance()*, é possível obter a instância única do hospital e realizar operações sobre ela, como adicionar médicos e enfermeiros, obter informações do hospital, entre outras.

2. Padrão Builder
O padrão Builder é utilizado nas classes Doctor e Nurse para facilitar a construção desses objetos complexos, fornecendo uma interface fluente para definir os atributos dessas classes e criar as instâncias correspondentes.

Justificativa de uso:

As classes Doctor e Nurse possuem atributos que podem variar e exigem a criação de objetos com diferentes combinações desses atributos.
O uso do padrão Builder simplifica a criação desses objetos, pois separa a construção do objeto da sua representação final.

Através dos métodos *name()* e *specialization()* no caso de Doctor, e *name()* e *experienceYears()* no caso de Nurse, é possível definir os valores dos atributos individualmente.
O método *build()* é responsável por criar a instância final da classe, utilizando os valores definidos no processo de construção.

3. Padrão Prototype
O padrão de projeto Prototype é utilizado para criar novos objetos através da clonagem de um objeto existente, evitando a necessidade de criar novas instâncias a partir do zero. No contexto do sistema de gestão hospitalar, o padrão Prototype pode ser aplicado para criar cópias de objetos que representam profissionais da área da saúde, como médicos e enfermeiros.

Justificativa de uso:

A classe **HealthcareProfessional**, apresentada no código fornecido, é uma classe abstrata que serve como base para as classes Doctor e Nurse. Essa classe implementa a interface Cloneable, que indica que objetos dessa classe podem ser clonados.

O método *clone()* na classe HealthcareProfessional é responsável por realizar a clonagem do objeto. Ele chama o método *clone()* da superclasse Object e faz um cast para o tipo HealthcareProfessional. Esse método retorna uma cópia do objeto, preservando o estado atual dos atributos.

Conclusão:

No desenvolvimento de sistemas complexos, como o sistema de gestão hospitalar apresentado, a escolha adequada dos padrões de projeto desempenha um papel fundamental na criação de um código estruturado, flexível e de fácil manutenção. Neste documento, exploramos três padrões de projeto amplamente utilizados: Singleton, Builder e Prototype.

O padrão Singleton foi aplicado no sistema de gestão hospitalar para garantir que haja apenas uma única instância da classe Hospital. Isso é vantajoso quando se deseja centralizar o acesso a uma instância compartilhada, evitando inconsistências e conflitos de estado. O Singleton garante que o Hospital seja único em todo o sistema, permitindo o acesso global e controlado aos seus métodos e atributos.

O padrão Builder foi empregado na construção dos objetos Doctor e Nurse, fornecendo uma maneira elegante de criar instâncias dessas classes com uma sintaxe fluente e legível. O uso do Builder simplifica a criação de objetos complexos, permitindo que sejam especificados os atributos desejados e fornecendo um ponto centralizado para a construção desses objetos. Isso melhora a clareza do código e facilita a manutenção, especialmente quando há muitos atributos opcionais ou mutáveis.

Por fim, o padrão Prototype foi utilizado para permitir a clonagem dos objetos HealthcareProfessional, representados pelas classes Doctor e Nurse. Através do método clone(), é possível criar cópias dos profissionais de saúde existentes, preservando o estado atual de seus atributos. Esse padrão oferece uma maneira eficiente de criar novos objetos a partir de modelos pré-existentes, evitando a necessidade de recriar instâncias do zero. Isso traz flexibilidade e melhora o desempenho do sistema.