PLANO DE PROJETO DE SOFTWARE

Cliente: Hospital

Responsáveis pelo projeto no IFSC: Bárbara Weege e Bernardo Niehues

Projeto: Padrões de Projeto Utilizados no Sistema Hospitalar

Unidade Curricular: Padrões de Projeto de Software

Professor: Renato Simões Moreira

GASPAR

2024

Sumário

1. Introdução

Este documento visa descrever a implementação de dois padrões de projeto — Singleton e Builder — em um sistema hospitalar, com foco na gestão de médicos e pacientes. Abaixo, discutiremos a aplicabilidade de cada padrão e justificamos sua escolha.

2. Padrão Singleton

2.1 Definição

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

2.2 Aplicabilidade no Sistema Hospitalar

No contexto do sistema hospitalar, o Singleton foi utilizado para a classe Hospital. Essa classe precisa gerenciar os médicos e pacientes de maneira centralizada, e é fundamental que exista apenas um "hospital" para garantir que todos os dados sejam consistentes ao longo do sistema.

A classe Hospital contém listas de médicos e pacientes e é responsável por adicionar novos médicos e pacientes, além de fornecer acesso a essas listas. Através do Singleton, garantimos que todas as partes do sistema acessem e modifiquem a mesma instância de Hospital, evitando conflitos ou inconsistências de dados.

2.3 Código de Exemplo

```
public class Hospital {
    private static Hospital instance;
    private List<Paciente> pacientes;
    private List<Medico> medicos;

    private Hospital() {
        pacientes = new ArrayList<>();
        medicos = new ArrayList<>();
    }

    public static Hospital getInstance() {
        if (instance == null) {
            instance = new Hospital();
        }
        return instance;
    }
```

```
    // Métodos para adicionar e listar pacientes e médicos
}
```

2.4 Justificativa

A escolha do Singleton para a classe Hospital se justifica pela necessidade de centralizar o gerenciamento de informações de médicos e pacientes. Como a instância é única, podemos garantir a integridade dos dados em todo o sistema, evitando múltiplas instâncias que possam ocasionar inconsistências.

2.5 Vantagens do Singleton:

- Controle centralizado de dados;
- Facilidade de acesso a uma única instância em todo o sistema.

2.6 Desvantagens:

- Pode dificultar testes unitários se não houver um mecanismo adequado de redefinição da instância.

3. Padrão Builder

3.1 Definição

O Builder é um padrão de criação que separa a construção de um objeto complexo da sua representação, permitindo que o mesmo processo de construção crie diferentes representações.

3.2 Aplicabilidade no Sistema Hospitalar

No sistema hospitalar, o padrão Builder foi aplicado para a criação das classes Paciente e Médico. Essas classes possuem atributos que nem sempre são conhecidos no momento da instanciação, como o endereço de um paciente ou a especialidade de um médico.

Com o Builder, podemos construir objetos de forma flexível, sem sobrecarregar o construtor com múltiplos parâmetros opcionais. Isso facilita a adição de novos atributos ou parâmetros no futuro, sem precisar alterar a lógica principal da construção dos objetos.

3.3 Código de Exemplo

3.3.1 Builder de Médico:

```
public class MedicoBuilder {
    private String nome;
    private String especialidade;

    public MedicoBuilder setNome(String nome) {
        this.nome = nome;
        return this;
    }

    public MedicoBuilder setEspecialidade(String especialidade) {
        this.especialidade = especialidade;
        return this;
    }

    public Medico build() {
        return new Medico(nome, especialidade);
    }
}
```

3.3.2 Builder de Paciente:

```
public class PacienteBuilder {
    private String nome;
    private int idade;
    private String endereco;

    public PacienteBuilder setNome(String nome) {
        this.nome = nome;
        return this;
    }

    public PacienteBuilder setIdade(int idade) {
        this.idade = idade;
        return this;
    }

    public PacienteBuilder setEndereco(String endereco) {
        this.endereco = endereco;
        return this;
    }

    public Paciente build() {
        return new Paciente(nome, idade, endereco);
    }
}
```

3.4 Justificativa

A escolha do padrão Builder para as classes Medico e Paciente se justifica pela flexibilidade que ele oferece na criação de objetos complexos com múltiplos parâmetros. Como as informações dos médicos e pacientes podem variar, o Builder permite criar objetos com apenas os dados disponíveis, mantendo a clareza e simplicidade do código.

3.5 Vantagens do Builder:

- Flexibilidade na construção de objetos com múltiplos parâmetros;
- Facilita a legibilidade do código;
- Evita a criação de construtores com muitos parâmetros.

3.6 Desvantagens:

- Pode adicionar complexidade extra quando os objetos são simples.

4. Conclusão

Os padrões Singleton e Builder foram aplicados de maneira a otimizar a criação e o gerenciamento de objetos dentro do sistema hospitalar. O Singleton garante uma instância única e centralizada do Hospital, enquanto o Builder proporciona flexibilidade na construção dos objetos Paciente e Médico.