Aplicação de Padrões de Projeto em um Sistema Hospitalar

1. Introdução

Neste documento, vamos falar sobre como usar quatro padrões de projeto em um sistema hospitalar. Esses padrões são: Singleton, Builder, Adapter e Strategy. Cada um desses padrões ajuda a resolver problemas diferentes, como gerenciar dados, criar relatórios, integrar sistemas e escolher tratamentos.

2. Padrões Criacionais

2.1 Singleton

O Singleton é usado para garantir que só exista uma única instância de uma classe em todo o sistema. No hospital, usamos o Singleton para o Gerenciador de Registros Médicos. Isso garante que todas as partes do sistema usem o mesmo conjunto de informações dos pacientes.

meu_projeto_hospital > src > singleton.py > GerenciadorRegistrosMedicos

```python
class GerenciadorRegistrosMedicos:
    _instancia = None

    def __new__(cls):
        if cls._instancia is None:
            cls._instancia = super(GerenciadorRegistrosMedicos, cls).__new__(cls)
            cls._instancia._registros = {}
        return cls._instancia

    def adicionar_registro(self, id_paciente, dados_paciente):
        self._instancia._registros[id_paciente] = dados_paciente

    def obter_registro(self, id_paciente):
        return self._instancia._registros.get(id_paciente)

```

2.2 Builder

O Builder ajuda a criar objetos complexos de uma maneira organizada e passo a passo. No hospital, usamos o Builder para criar Relatórios Médicos. Assim, podemos adicionar informações diferentes, como nome do paciente, diagnóstico e prescrição, de maneira flexível.

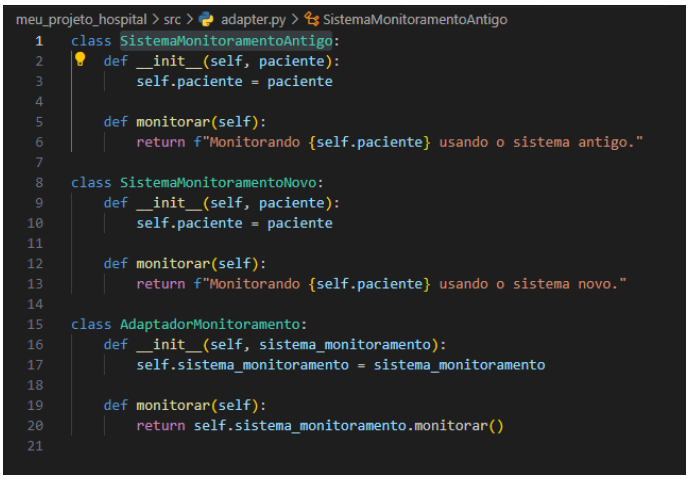
meu_projeto_hospital > src > builder.py > ConstrutorRelatorioMedico

```python
class ConstrutorRelatorioMedico:
    def __init__(self):
        self.relatorio = {}

    def adicionar_informacoes_paciente(self, nome, idade):
        self.relatorio['nome'] = nome
        self.relatorio['idade'] = idade
        return self

    def adicionar_diagnostico(self, diagnostico):
        self.relatorio['diagnostico'] = diagnostico
        return self

    def adicionar_prescricao(self, prescricao):
        self.relatorio['prescricao'] = prescricao
        return self

    def construir(self):
        return self.relatorio

```

3. Padrão Estrutural

3.1 Adapter

O Adapter ajuda a conectar sistemas que não são compatíveis. No hospital, usamos o Adapter para que o Sistema de Monitoramento Novo e o Antigo possam trabalhar juntos. Com o Adapter, ambos os sistemas podem se comunicar sem precisar mudar os sistemas antigos.



4. Padrão Comportamental

4.1 Strategy

O padrão Strategy permite definir diferentes formas de fazer algo e escolher a melhor quando necessário. No hospital, usamos o Strategy para Gerenciar Estratégias de Tratamento. Dependendo da condição do paciente, podemos escolher entre medicação, cirurgia ou fisioterapia sem mudar a lógica principal do sistema.