Relatório Programação Orientada a Objetos

Docente: Luís Ferreira

Ricardo Amaro (26004)
João Ferreira (a25986)

15 de novembro de 2023

Resumo

Este trabalho da Unidade Curricular (UC) de Programação Orientada a Objetos (POO) foca a análise de problemas reais simples e a aplicação do Paradigma Orientado a Objetos na implementação de possíveis soluções.

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

O presente relatório resulta do trabalho desenvolvido na Unidade Curricular de Programação Orientada a Objetos (POO), onde explorámos a aplicação do Paradigma Orientado a Objetos para resolver problemas reais utilizando a linguagem de programação C. O foco principal centra-se na criação de soluções eficazes para desafios específicos, incluindo a identificação de classes, definição de estruturas de dados e implementação dos principais processos.

Este projeto vai além da programação, envolvendo o desenvolvimento da capacidade analítica para abordar e resolver problemas do mundo real. Destaca-se também a importância da competência em redação técnica, com ênfase na clareza na exposição de ideias e na elaboração de documentação adequada, elementos fundamentais no desenvolvimento de sistemas orientados a objetos.

Capítulo 2

Diagrama de Classes: MedSync RM

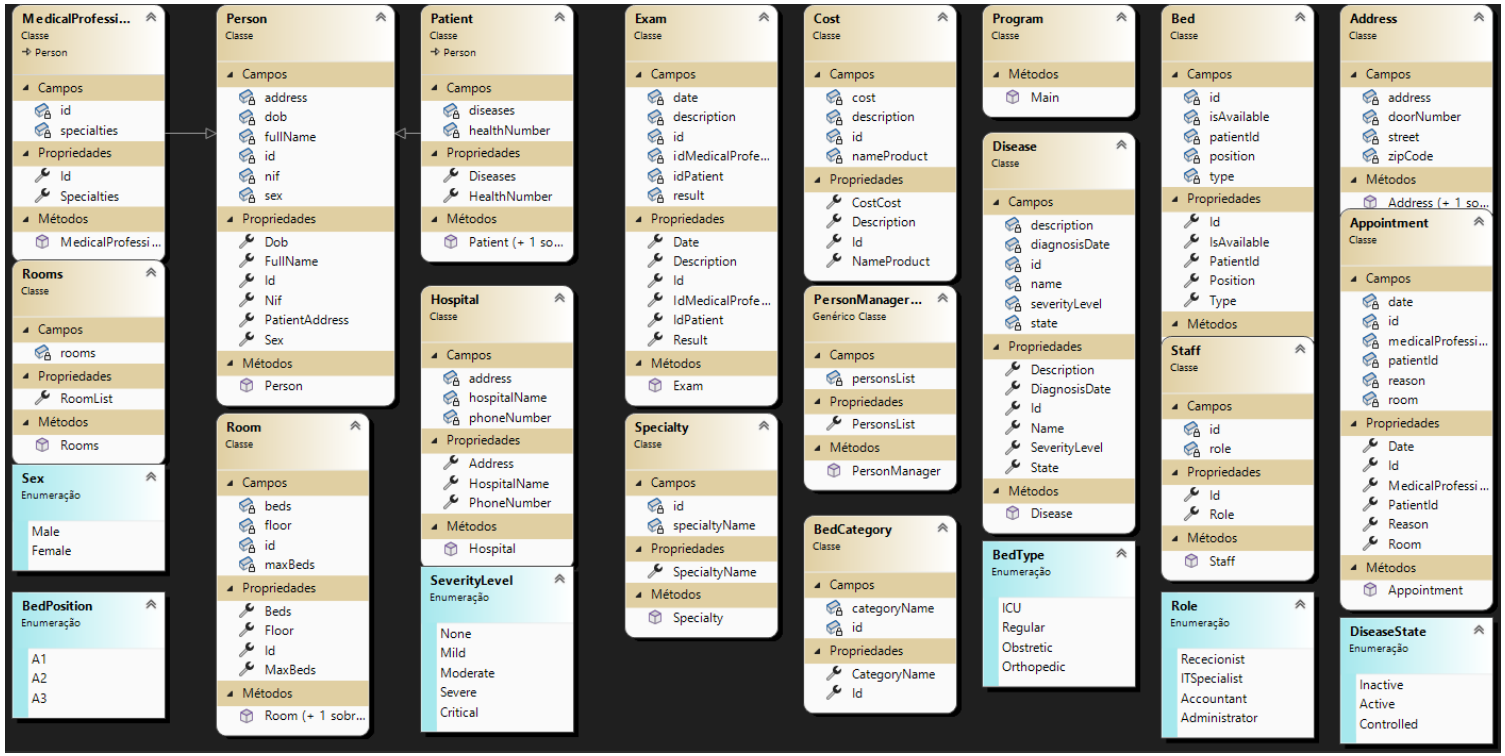

Figura 2.1: Diagramas de Classe

Address: Classe útil para armazenar e organizar informações essenciais de localização, como rua, código postal e número da porta, facilitando a gestão e referência de endereços em um sistema.

Appointment: Classe destinada a representar consultas com médicos. Possui atributos como ID, ID do paciente, ID do profissional médico, motivo, sala e data da consulta.

Bed: Classe com o propósito de representar camas hospitalares. Inclui atributos como ID, tipo de cama (ICU, Regular, Obstétrico, Ortopédico), disponibilidade, ID do paciente e posição da cama.

BedCategory: Classe desenvolvida para representar categorias de camas hospitalares. Possui atributos como ID e nome da categoria. Esta classe é útil para categorizar camas hospitalares.

Cost: Classe destinada a representar custos com atributos como ID, nome do produto, descrição e valor.

Disease: Classe desenvolvida para representar informações relacionadas a doenças. Possui atributos como ID, nome, estado da doença, data de diagnóstico, descrição e nível de gravidade.

Exam: Classe criada para representar exames médicos. Inclui atributos como ID, ID do paciente, ID do profissional médico, data do exame, descrição e resultado.

Hospital: Classe destinada a representar informações sobre hospitais. Possui atributos como o nome do hospital, endereço (utilizando a classe Address) e número de telefone.

MedicalProfessional: Classe derivada da classe Person, desenvolvida para representar profissionais médicos. Possui atributos como ID e uma lista de especialidades (do tipo Specialty).

Patient: Classe derivada da classe Person, criada para representar informações relacionadas a pacientes. Possui atributos como uma lista de doenças (do tipo Disease) e um número de saúde.

Person: Classe destinada a representar informações pessoais. Possui atributos como ID, nome completo, data de nascimento, número de identificação fiscal (NIF), endereço (utilizando a classe Address) e sexo.

PersonManager<T>: Classe criada para gerir objetos do tipo Person ou suas subclasses. Possui um atributo, personsList, que é uma lista genérica do tipo T, restrita a subclasses de Person.

Program: Classe interna que contem o método Main como ponto de entrada do programa.

Room: Classe destinada a representar informações sobre salas hospitalares, incluindo atributos como ID, lista de leitos (do tipo Bed), número máximo de leitos e andar.

Rooms: Classe destinada a representar um conjunto de salas hospitalares. Possui um atributo, rooms, que é uma lista de objetos do tipo Room.

Specialty: Classe criada para representar especialidades médicas. Inclui atributos como ID e nome da especialidade.

Staff: Classe criada para representar informações sobre o staff do hospital. Inclui

atributos como ID e função (role) na instituição, representada pelo enum Role.

2.1 Herança

O diagrama de classes apresenta um exemplo claro do conceito de herança, que descreve a relação hierárquica entre as classes *Person*, *Patient* e *MedicalProfessional*. A herança nesse contexto permite uma representação eficiente e lógica das relações entre diferentes entidades. A utilização dessa técnica na construção do diagrama de classes proporciona uma estrutura organizada, onde a classe base *Person* fornece uma base compartilhada para as subclasses *Patient* e *MedicalProfessional*, refletindo a natureza hierárquica e especializada dessas entidades no contexto do sistema modelado.

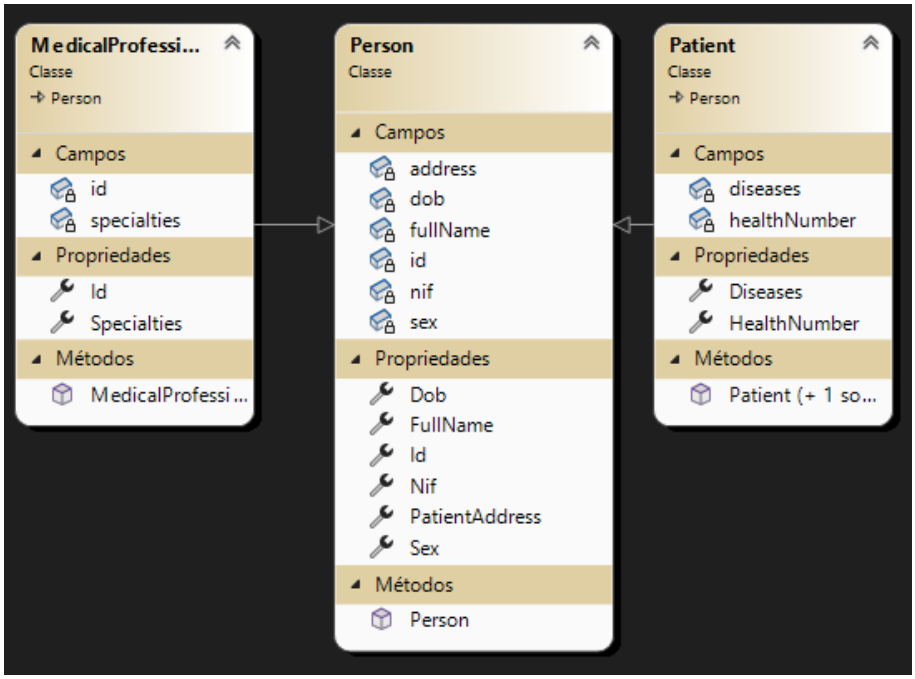

Figura 2.2: Diagrama de classes que demostram uma herança de uma superclasse.