UNIVERSIDADE DE BRASÍLIA - UnB
FACULDADE DE CIÊNCIAS E TECNOLOGIA EM ENGENHARIA - FCTE
ORIENTAÇÃO À OBJETOS

JOÃO LUCAS REOLON CABRAL
LUCAS DE SOUZA
RAFAELA ANDREA RADAMES GUERRA

RELATÓRIO DO SISTEMA DE GERENCIAMENTO DE CLÍNICA MÉDICA

Brasília/DF
2025

JOÃO LUCAS REOLON CABRAL - 232014692

LUCAS DE SOUZA - 241039654

RAFAELA ANDREA RADAMES GUERRA - 231031723

RELATÓRIO DO SISTEMA DE GERENCIAMENTO DE CLÍNICA MÉDICA

Relatório apresentado como requisito de aprovação na disciplina Orientação à Objetos do curso de Engenharias – FCTE.

Prof.: Andre Luiz Peron Martins Lanna

Brasília/DF

2025

RESUMO

O presente trabalho tem como objetivo relatar o desenvolvimento um sistema de gerenciamento para uma clínica médica, utilizando os conceitos de Orientação a Objetos (OO) em Java. O sistema permite o cadastro de pacientes e médicos, agendamento de consultas, prescrição de exames e medicamentos, gestão de pagamentos e tratamento de exceções personalizadas. Foram aplicados princípios como encapsulamento, herança, polimorfismo e modularidade, além da criação de um diagrama de classes UML para representar as entidades e suas relações. O sistema foi estruturado em pacotes como *entidades*, *servicos* e *excecoes*, garantindo organização e clareza no código. Como resultado, obteve-se uma aplicação funcional que atende às necessidades de uma clínica médica, com validações de regras de negócio e tratamento de erros específicos.

ABSTRACT

This work aims to relate a development a management system for a medical clinic, using Object-Oriented (OO) concepts in Java. The system allows the registration of patients and doctors, scheduling of appointments, prescription of exams and medications, payment management, and handling of custom exceptions. Principles such as encapsulation, inheritance, polymorphism, and modularity were applied, along with the creation of a UML class diagram to represent the entities and their relationships. The system was structured into packages such as *entidades*, *servicos* and *excecoes*s, ensuring code organization and clarity. As a result, a functional application was obtained that meets the needs of a medical clinic, with validations of business rules and specific error handling.

1. INTRODUÇÃO

A gestão de clínicas médicas envolve uma série de desafios, como o controle de consultas, o registro de históricos médicos, a prescrição de exames e medicamentos, e a administração de pagamentos. A automação desses processos por meio de um sistema computacional pode trazer eficiência, redução de erros e melhor experiência para pacientes e profissionais da saúde. Neste contexto, o presente trabalho propõe o desenvolvimento de um sistema de gerenciamento de clínica médica, utilizando os conceitos de Orientação a Objetos (OO) em Java.

O sistema foi projetado para atender aos requisitos funcionais descritos, como o cadastro de pacientes e médicos, agendamento de consultas com validações de horários e especialidades, prescrição de exames e medicamentos, e gestão de pagamentos. Além disso, foram implementadas exceções personalizadas para tratar situações específicas, como horários indisponíveis, pagamentos pendentes e especialidades inválidas.

A aplicação foi estruturada em pacotes que separam as entidades, serviços e exceções, garantindo modularidade e organização. Um diagrama de classes UML foi criado para representar as principais entidades e suas relações, como Paciente, Médico, Consulta, Exame e Pagamento. O uso de herança, polimorfismo e encapsulamento foi explorado para garantir a reutilização de código e a manutenção do sistema.

Este relatório apresenta a estrutura do sistema, o diagrama de classes UML, as justificativas para as exceções customizadas e as decisões de projeto. Ao final, espera-se que o sistema desenvolvido atenda às necessidades de uma clínica médica, proporcionando uma gestão eficiente e segura dos processos envolvidos.

2. REQUISITOS FUNCIONAIS

O sistema de gerenciamento de clínica médica foi desenvolvido com base nos seguintes requisitos funcionais:

2.1 Cadastro de Pacientes e Médicos

1. Pacientes: Atributos obrigatórios: nome, CPF, data de nascimento e histórico médico (lista de consultas e exames).
2. Funcionalidades: CRUD completo, adicionar consulta ao histórico e bloquear cadastro se CPF já estiver registrado.
3. Médicos: Atributos obrigatórios: nome, CPF, data de nascimento, CRM e especialidade.
4. Funcionalidades: CRUD completo, adicionar consulta ao histórico e bloquear cadastro se CPF já estiver registrado.

2.2 Agendamento de Consultas

O sistema permite agendar consultas apenas se:

1. O médico estiver disponível no horário.

2. O paciente não tiver outra consulta no mesmo dia.
3. O médico tiver a especialidade requerida.

Para cada consulta, são informados: data, horário de início, duração, status, paciente, médico, lista de exames prescritos, lista de medicamentos e valor.

2.3 Prescrição de Exames e Medicamentos

1. Médicos podem prescrever exames (ex: sangue, raio-X) ou medicamentos, associando-os à consulta.
2. O sistema implementa CRUD completo para exames e medicamentos.

2.4 Gestão de Pagamentos

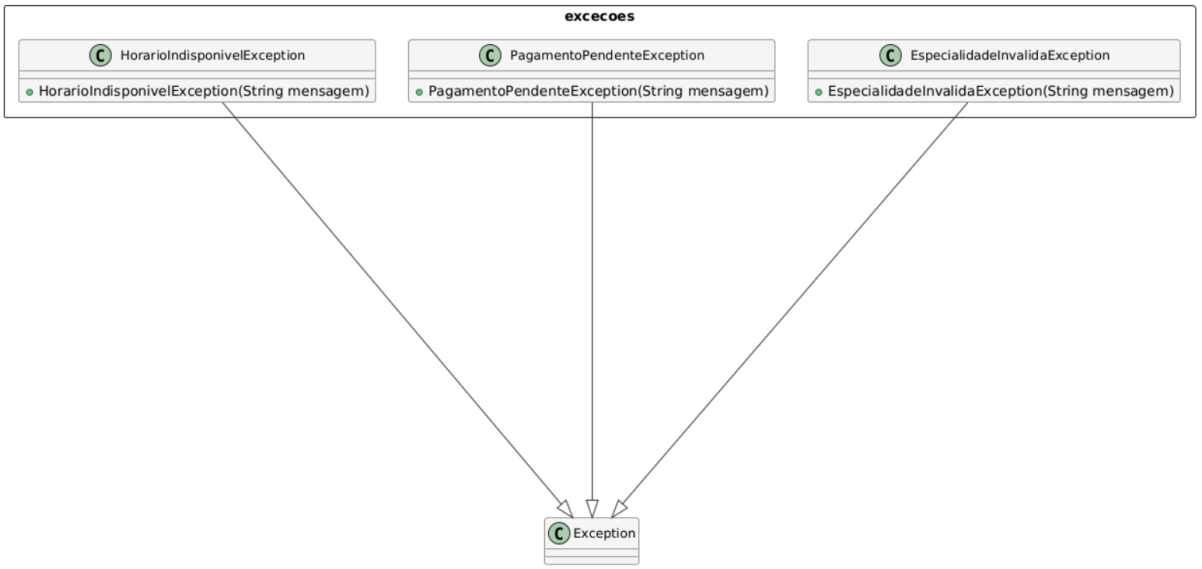
1. Cada consulta/exame tem um valor associado.
2. Pacientes com pagamentos pendentes não podem agendar novas consultas.

2.5 Tratamento de Exceções

1. Exceções personalizadas são utilizadas para lidar com:
2. Agendamento em horário indisponível.
3. Paciente com pagamento pendente.
4. Médico não encontrado para uma especialidade.

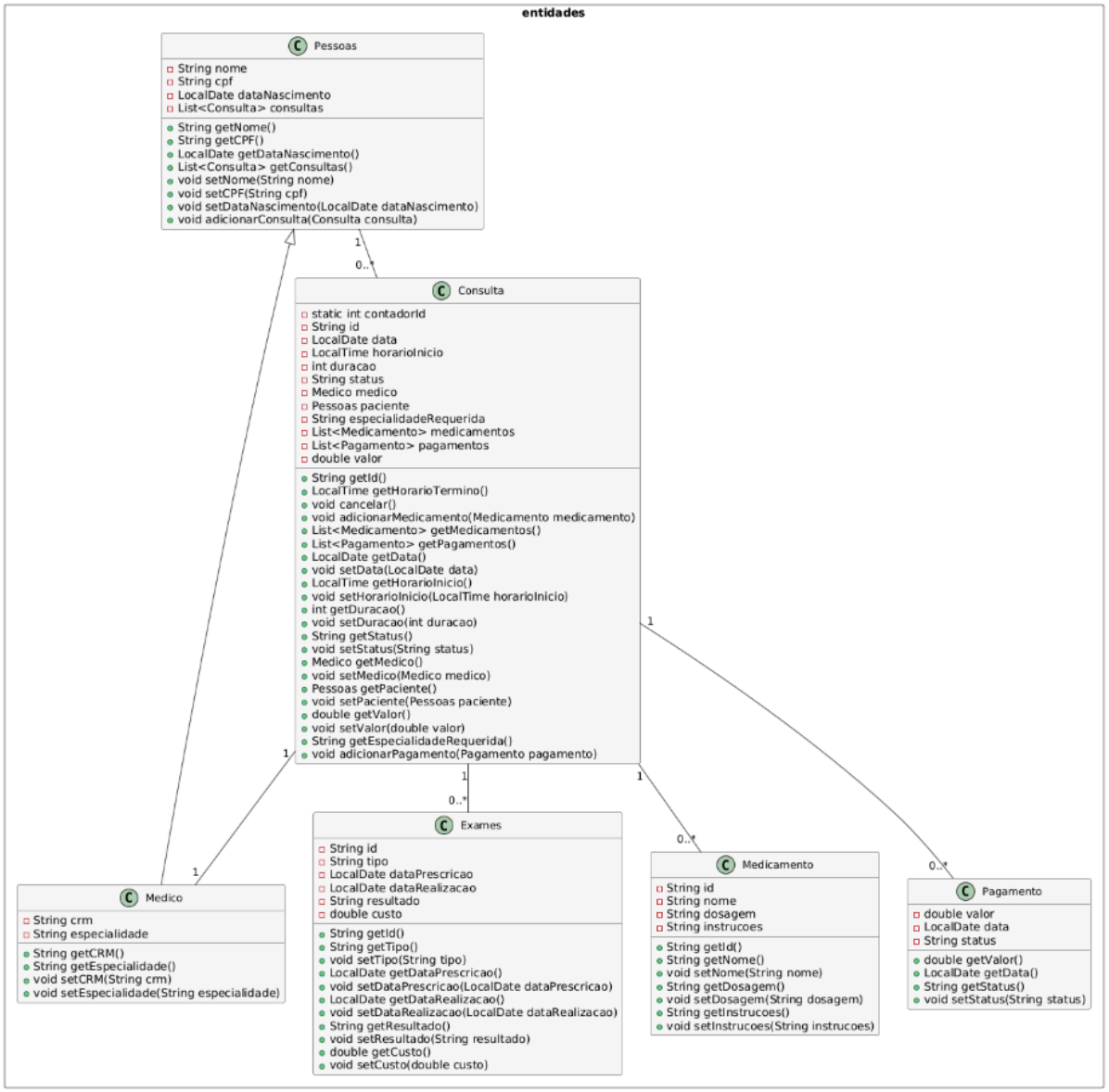
3. DIAGRAMA DE CLASSES UML

O diagrama de classes UML representa a estrutura do sistema, organizado em três pacotes principais:exceções, entidade e serviço .



Fonte: Os autores (2025).

A seguir o diagrama do pacote de entidades:

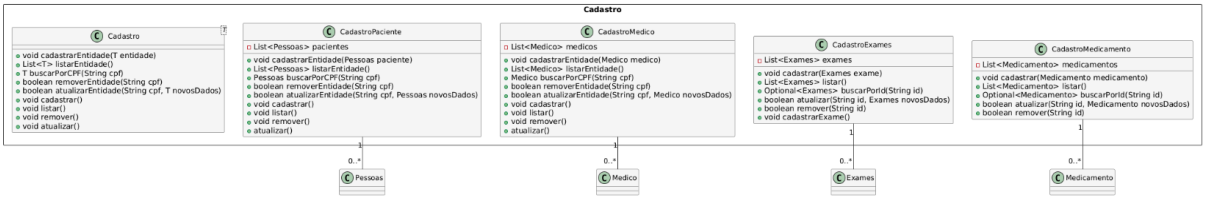


Fonte: Os autores (2025)

Por último.o diagrama do pacote de serviços:

Fonte: Os autores (2025)

Incluindo seus cadastros:



Fonte: Os autores (2025)

3.1 Principais Classes, Atributos e Métodos

Pacote entidades:

1. Pessoas: Classe base que representa pacientes e médicos. Contém atributos como nome, cpf, dataNascimento e uma lista de consultas.
2. Medico: Herda de Pessoas e adiciona atributos específicos como crm e especialidade.
3. Consulta: Representa uma consulta médica, com atributos como data, horarioInicio, status, e relacionamentos com Medico, Paciente, Medicamento, Pagamento e Exames.
4. Exames: Representa um exame médico, com atributos como tipo, dataPrescricao, resultado e custo.
5. Medicamento: Representa um medicamento prescrito, com atributos como nome, dosagem e instrucoes.
6. Pagamento: Representa um pagamento associado a uma consulta, com atributos como valor, data e status.

Pacote servicos:

1. Agendamento: Gerencia o agendamento e desagendamento de consultas.
2. ServicoPagamento: Gerencia os pagamentos das consultas.
3. Cadastro: Classe genérica para cadastro de entidades (pacientes, médicos, exames, medicamentos).
4. Pacote excecoes:

5. HorarioIndisponivelException: Lançada quando um médico não está disponível no horário solicitado.
6. PagamentoPendenteException: Lançada quando um paciente tem pagamentos pendentes.
7. EspecialidadeInvalidaException: Lançada quando um médico não possui a especialidade requerida.

3.2 Relacionamentos e Multiplicidades

1. Pessoas → Consulta (1 para muitos): Um paciente ou médico pode ter várias consultas.
2. Consulta → Medico (1 para 1): Cada consulta está associada a um médico.
3. Consulta → Medicamento (1 para muitos): Uma consulta pode prescrever vários medicamentos.
4. Consulta → Pagamento (1 para muitos): Uma consulta pode ter vários pagamentos.
5. Consulta → Exames (1 para muitos): Uma consulta pode prescrever vários exames.

4. EXPLICAÇÃO DAS ASSOCIAÇÕES, HERANÇAS E POLIMORFISMO

4.1 Associações

1. Paciente e Consulta: Um paciente pode ter várias consultas, representado pela multiplicidade 1..*.
2. Médico e Consulta: Um médico pode ter várias consultas, representado pela multiplicidade 1..*.
3. Consulta e Medicamento/Exames: Uma consulta pode prescrever vários medicamentos e exames, representado pela multiplicidade 1..*.

4.2 Heranças

1. Pessoas e Medico: A classe Medico herda de Pessoas, aproveitando atributos como nome, cpf e dataNascimento.
2. Exceções Personalizadas: As exceções personalizadas herdam da classe Exception do Java.

4.3 Polimorfismo

1. Sobrescrita de Métodos: A classe CadastroPaciente sobrescreve métodos da classe Cadastro<T>.
2. Sobrecarga de Métodos: A classe Agendamento possui métodos sobrecarregados, como agendarConsulta().
3. Polimorfismo Paramétrico: A classe Cadastro<T> utiliza generics para ser reutilizada com diferentes tipos de entidades.

5. JUSTIFICATIVA PARA EXCEÇÕES PERSONALIZADAS

1. HorarioIndisponivelException: Evita conflitos de horários ao agendar consultas.
2. PagamentoPendenteException: Garante que pacientes com pendências financeiras não agendem novas consultas.
3. EspecialidadeInvalidaException: Garante que consultas sejam agendadas apenas com médicos da especialidade correta.

6. CONCLUSÃO

O sistema de gerenciamento de clínica médica desenvolvido atende aos requisitos funcionais propostos, aplicando os conceitos de Orientação a Objetos de forma eficiente. A modularidade, o uso de exceções personalizadas e a validação de regras de negócio garantem a robustez e a segurança da aplicação. Dificuldades foram encontradas durante o desenvolvimento, principalmente na validação de horários e especialidades, mas foram superadas com a implementação de exceções personalizadas. Como melhorias futuras, sugere-se a integração com um banco de dados e a implementação de uma interface gráfica.

REFERÊNCIAS BIBLIOGRÁFICAS

DEITEL, P. J.; DEITEL, H. M. **Java: Como Programar**. 10ª ed. Pearson, 2017.

FOWLER, M. **UML Essencial: Um Breve Guia para a Linguagem Padrão de Modelagem de Objetos**. 3ª ed. Bookman, 2005.

Documentação Oficial do Java: https://docs.oracle.com/javase/8/docs/