**Relatório do Trabalho Prático**

**Orientação a Objetos**

**FGA – 0158**

# Gestão de Pacientes em Clínica Médica

**Professor(a):** Dr. André Luiz Peron Martins Lanna

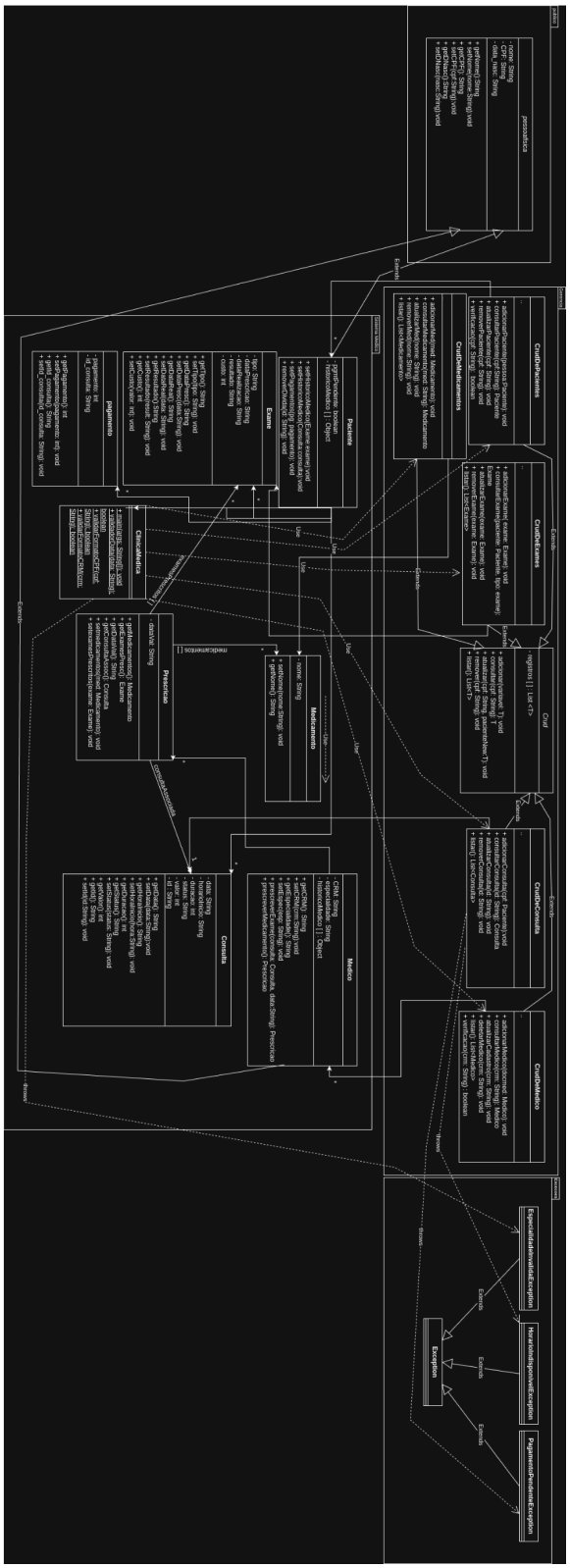
**Aluno(a):** Enzo Fernandes Borges – 202017361

**Aluno(a):** Milla Reis Pereira - 232014530

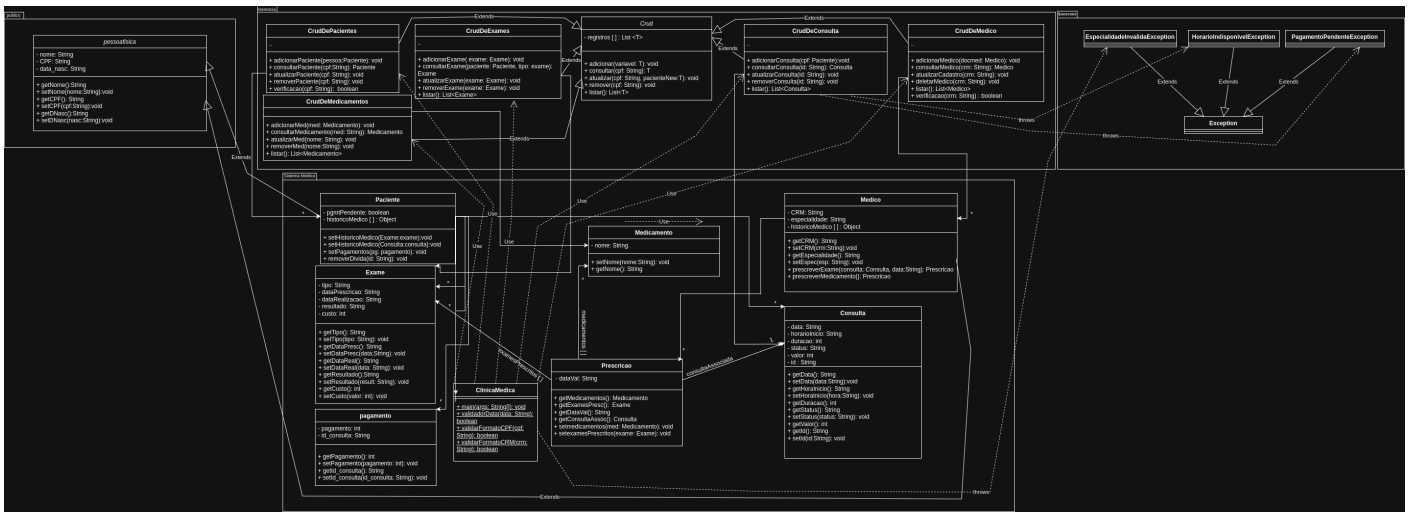
**14 de Fevereiro de 2025**

**Universidade de Brasília**

# UML NA VERTICAL



# UML NA HORIZONTAL




**Link do UML** (para melhor visualização): **LinkUML**

# Sumário

# Relatório de Análise do Diagrama UML

**1. Introdução** Este relatório tem como objetivo analisar a modelagem orientada a objetos do sistema de gerenciamento de uma clínica médica, conforme representado no diagrama UML fornecido. Serão exploradas as relações entre as classes, heranças aplicadas e aspectos de polimorfismo.

---

**2. Associações entre Classes** As associações representam as relações entre os objetos do sistema, indicando como eles interagem entre si. Algumas das principais associações observadas no diagrama incluem:

- **Paciente, Consulta e Exame** : Cada paciente pode ter um histórico médico, estabelecendo uma relação de composição entre a classe `Paciente` com as classes `Consulta e Exame`.
- **Médico e Consulta**: A classe `Médico` também está associada a `Consulta`, indicando que um médico pode realizar várias consultas.
- **Consulta e Exame**: Existe uma associação entre `Consulta` e `Exame`, indicando que exames podem ser prescritos durante uma consulta.
- **Paciente e Pagamento**: Um paciente possui uma relação com `Pagamento`, pois cada consulta ou exame realizado pode gerar um pagamento pendente.
- **Prescrição**: A classe `Prescrição` está associada tanto a `Exame` quanto a `Medicamento`, indicando que prescrições podem incluir exames e medicamentos.

---

**3. Herança Aplicada** A herança é utilizada para reaproveitamento de atributos e métodos comuns entre classes relacionadas. No diagrama, observa-se o seguinte uso de herança:

- **PessoaFisica (Classe Base)**:
    - `Paciente` e `Médico` herdam de `PessoaFísica`, garantindo que ambos possuam atributos comuns como nome, CPF e data de nascimento.
- **Exceções Personalizadas**:
    - `EspecialidadeInvalidaException`, `HorarioIndisponivelException` e `PagamentoPendenteException` são subclasses de `Exception`, criando um modelo estruturado para tratamento de erros específicos no sistema.
- **Sistema CRUD**:
    - As classes de gerenciamento (`CrudDePacientes`, `CrudDeMedico`, `CrudDeConsulta`, `CrudDeExames`, `CrudDeMedicamentos`) derivam de uma classe base `Crud<T>`, que fornece operações genéricas de CRUD. CRUD é um acrônimo para Create, Read, Update e Delete.

---

**4. Polimorfismo** O conceito de polimorfismo é aplicado no sistema através de:

- **Polimorfismo Paramétrico (Uso de Generics no CRUD)**: A classe `Crud<T>` permite operações genéricas para diferentes tipos de dados, garantindo reutilização de código e flexibilidade.
- **Tratamento de Exceções**: Como as exceções personalizadas herdam de `Exception.`
- **Métodos sobrescritos (Sobrescrita de Métodos)**: As classes CRUD implementam seus próprios métodos para operações específicas, sobrescrevendo funções da classe base.
- **Sobrecarga de Métodos**: O processo de definir múltiplos métodos com o **mesmo nome**, mas com **assinaturas diferentes** (diferentes números ou tipos de parâmetros). Utilizado na classe `Paciente`, no método **`setPgmtPendente()`**.
- **Polimorfismo por coerção:** A classe `Paciente` e `Medico` permitiram manipulação de objetos de tipos distintos (`Exame` e `Consulta`) no mesmo Array.

---

**5. Exceções Customizadas**

Foram essenciais para melhorar a robustez do sistema e fornecer mensagens mais especificas e objetivas sobre erros.

- `EspecialidadeInvalidaException`
  - Exceção usada quando há uma tentativa de cadastrar uma consulta de uma determinada especialidade com médico de outra.
- `HorarioIndisponivelException`
  - Ocorre quando um paciente tenta agendar uma consulta, mas o horário escolhido já se encontra ocupado.
- `PagamentoPendenteException`
  - Indica que um paciente está tentando realizar uma ação(como agendar uma nova consulta ou exame) sem quitar pagamentos anteriores.

**6. Conclusão** O diagrama UML do sistema de gerenciamento da clínica médica apresenta uma estrutura bem definida, utilizando princípios da orientação a objetos, como encapsulamento, herança e polimorfismo. As associações estão bem representadas, garantindo uma modelagem eficiente para o funcionamento do sistema. O uso de exceções personalizadas, bem como a implementação de um CRUD genérico, destaca boas práticas de programação orientada a objetos.