Relatório Consolidado: Associações, Heranças, Polimorfismos e Exceções

Resumo

Para construir o sistema de gerenciamento de consultas foi decidido utilizar uma estrutura bem definida com separação de responsabilidades, integrando conceitos sólidos de orientação a objetos como associações, heranças e polimorfismo. Essa abordagem não só facilita a manutenção e a escalabilidade do sistema, mas também permite que cada componente se comunique de forma clara e consistente. O uso de exceções customizadas complementa essa estrutura, assegurando que regras de negócio importantes sejam rigorosamente aplicadas e que a integridade dos dados seja preservada em todas as operações.

I. Relações de Herança, Associações e Polimorfismo

Associações

1. **Entities:**
    - **Consulta ↔ Paciente e Médico:** Uma consulta está associada a um paciente e a um médico.
    - **Consulta ↔ Prescrição:** Uma consulta pode gerar uma prescrição.
    - **Prescrição ↔ Medicamento e Exame:** Uma prescrição contém uma lista de medicamentos e exames.
2. **Controllers ↔ Repositories:**
    - Cada controller (ex: MedicoController, ConsultaController) utiliza um repositório correspondente para operações de persistência (ex: MedicoRepository, ConsultaRepository).
3. **Views ↔ Controllers:**

- As views (ex: MedicoView, ConsultaView) interagem com controllers para executar ações (ex: criar, atualizar, listar).
- Views específicas (ex: PrescricaoView) dependem de múltiplos controllers (ex: PrescricaoController, MedicamentoController).

4. **Views ↔ Entities:**

    - As views formatam e exibem dados de entidades (ex: exibir detalhes de um paciente ou histórico de consultas).

Heranças

1. **PessoaFisica (Superclasse):**

    - **Médico e Paciente (Subclasses):** Compartilham atributos como nome, CPF, data de nascimento e histórico de consultas.
2. **BaseView (Superclasse):**

    - **Todas as Views (Subclasses):** Herdam métodos comuns para interação com o usuário, como leitura de dados e exibição de mensagens.

Polimorfismos

1. **Sobrescrita (Override):**

    - **Método adicionarConsultaAoHistorico:** Implementado de forma específica em Médico e Paciente para adicionar consultas ao histórico.
    - **Método showMenu:** Cada view redefine este método para exibir um menu específico de sua entidade.
2. **Sobrecarga (Overloading):**

- **Métodos CRUD:** Controllers e repositories possuem múltiplas versões de métodos como delete (ex: deletar por objeto ou por ID).

3. **Polimorfismo de Subtipagem:**

   - **Tratamento de Médico e Paciente como PessoaFisica:** Permite operações genéricas em listas ou métodos que aceitam a superclasse.
   - **Uso de BaseView:** Todas as views podem ser referenciadas pela superclasse, aproveitando métodos compartilhados.

II. Análise das Exceções

1. **PacientePossuiConsultaNoMesmoDIa**

   - **Descrição:** Exceção lançada quando um paciente já possui uma consulta agendada no mesmo dia.
   - **Finalidade:** Impedir que sejam realizadas múltiplas consultas para o mesmo paciente em um único dia, evitando conflitos de agendamento.

2. **PagamentoPendenteException**

   - **Descrição:** Exceção lançada quando há um pagamento pendente relacionado a uma consulta ou serviço.
   - **Finalidade:** Garantir que todos os pagamentos pendentes sejam resolvidos antes de prosseguir com operações críticas, protegendo a integridade financeira do sistema.

3. **CpfJaCadastradoException**

   - **Descrição:** Exceção lançada quando o CPF informado já está cadastrado no sistema.
   - **Finalidade:** Prevenir a duplicação de cadastros, assegurando que cada CPF seja único para pacientes e médicos.

4. **EspecialidadeInvalidaException**

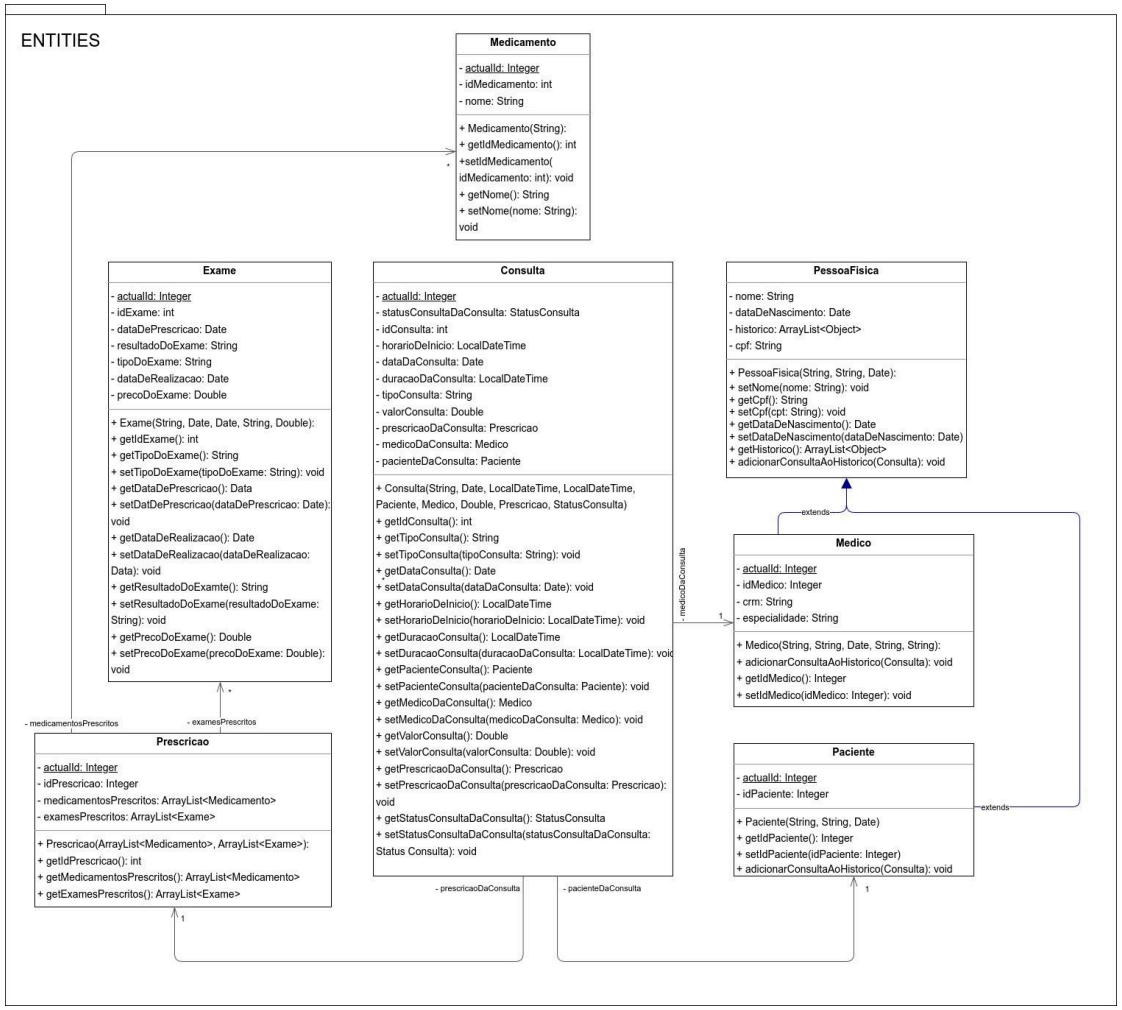
- **Descrição:** Exceção lançada quando uma especialidade inválida é atribuída a um médico ou consulta.
- **Finalidade:** Assegurar que os médicos realizem consultas apenas dentro de suas especialidades, evitando erros na alocação de recursos e atendimentos inadequados.

5. **HorarioIndisponivelException**

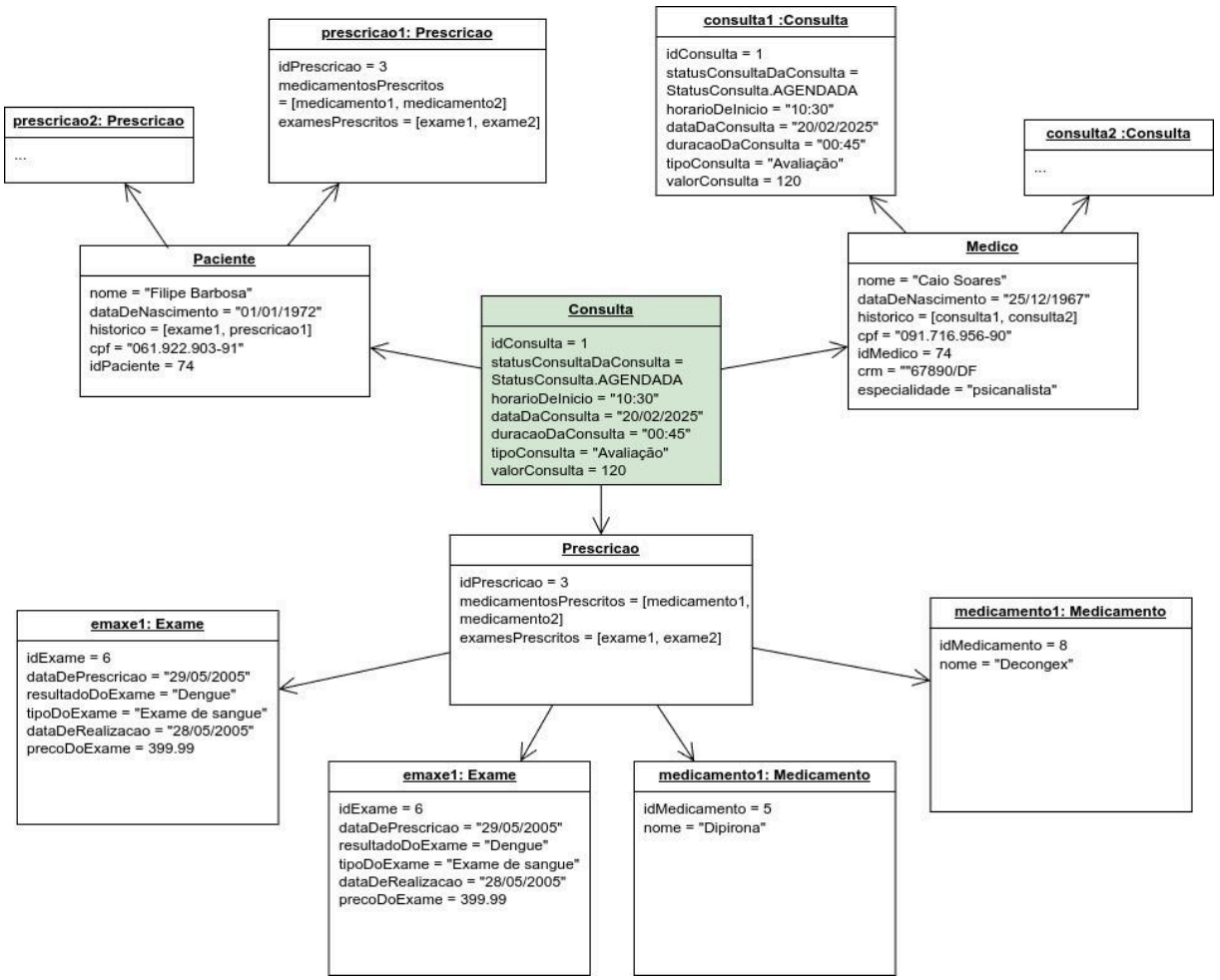
- **Descrição:** Exceção lançada quando o horário desejado para agendamento de consulta não está disponível.
- **Finalidade:** Evitar a sobreposição de consultas e conflitos de horário, garantindo a disponibilidade e organização dos agendamentos.

III. UMLs

1. Diagrama de Classes do Modelo (Entities)



2. Diagrama de Objetos em um momento ao criarmos um objeto do tipo “Consulta” do pacote Entities.



IV. Conclusão

Ao implementar o sistema de gerenciamento de consultas, optamos por uma arquitetura orientada a objetos que enfatiza a clareza na divisão de responsabilidades. A utilização de associações, heranças e polimorfismo foi essencial para modelar as interações entre as entidades, controllers e views de forma coesa.

A integração de exceções customizadas fortalece a confiabilidade do sistema, garantindo que as regras de negócio sejam rigorosamente aplicadas. Como resultado, construímos um sistema robusto, organizado e seguro, refletindo boas práticas de desenvolvimento de software.