# Sistema Hospitalar em Java

Raissa Silva de Oliveira – 232014763

Davi Monteiro de Negreiros – 232013971

Guilherme Negreiros Pereira – 232014001

Pedro Barreto Cavalcante do Amaral – 232038433

# Sistema de Gerenciamento de uma Clínica Médica

## Introdução e divisão do projeto

Este relatório apresenta o desenvolvimento de um Sistema Hospitalar em Java, com o intuito de agilizar e tornar mais eficiente o gerenciamento de médicos, pacientes e consultas. O sistema permite o cadastro de usuários, o agendamento de consultas, a prescrição de exames e medicamentos, além do controle de pagamentos.

O sistema foi desenvolvido utilizando os conceitos de Orientação a objetos, como herança, polimorfismo, encapsulamento e tratamento de exceções. A estrutura do código está organizada em pacotes para facilitar a manutenção.

O projeto está dividido entre duas pastas: out, que contém todos os arquivos compilados (.class) e src, que contém todos os arquivos executáveis (.java). A pasta src contém quatro pacotes (cadastro, classes, exceptions e view) e o arquivo Main. Os pacotes foram criados com o intuito de separar e organizar os arquivos de acordo com suas funções. O pacote cadastro possui os arquivos com as classes CadastroConsulta, CadastroExame, CadastroMédico e CadastroPaciente; classes possui as classes Consulta, Exame, Medicamento, Médico, Paciente, PessoaFisica, Validações e Prescrição; exceptions possui as classes CampoEmBrancoException, CpfDuplicadoException, CrmDuplicadoException, EspecialidadeInvalidaException, HorarioIndisponivelException e PagamentoPendenteException e view possui as classes MenuInicial, MenuMedico, MenuPaciente, MenuConsulta e MenuExame.

## Conceitos aplicados

**Herança:** As classes Médico e Paciente herdam atributos de PessoaFisica, que atua como sua classe base (superclasse). Ao invés de repetir esses atributos e métodos nas classes, usa-se o conceito de Herança e as subclasses herdam de sua classe base utilizando o comando *extends*.

**Associação**: As classes Consulta e Paciente possuem uma relação de associação, pois um Paciente possui uma lista de Consultas associadas a ele. Da mesma forma, as classes CadastroPaciente e Paciente também representam uma associação, já que CadastroPaciente mantém uma lista de objetos do tipo Paciente. Essa mesma lógica se aplica às relações entre CadastroMedico e Meduico, CadastroConsulta e Consulta, e CadastroExame e Exame, onde cada classe de cadastro gerencia uma lista de objetos do tipo correspondente**.**

**Associação do tipo *uso:*** A associação do tipo uso esta presente em quase todo o projeto pois: Main *usa* MenuInicial; MenuInicial *usa* MenuPaciente e MenuMedico; Menus *usam* Paciente, Medico, Consulta etc.

**Polimorfismo por Sobrescrita:** Na superclasse PessoaFisica existe um método toString () de uma forma e em Paciente e Medico existe o mesmo método, mas chamando novos atributos.

**Polimorfismo por Sobrecarga:**

## Exceptions personalizadas

**CampoEmBrancoException:** A exceção de campo em branco foi aplicada para solucionar o erro que pode ocorrer se o usuário esquecer ou pular uma etapa do cadastro. Foi implementado um método nas classes MenuPaciente e MenuMedico em todos os campos que devem ser preenchidos

**CpfDuplicadoException:** A exceção de Cpf duplicado foi aplicada para evitar que um paciente já cadastrado se cadastre novamente, foi implementada na classe MenuPaciente.

**CrmDuplicadoException:** Seguindo a mesma lógica de Cpf duplicado, Crm duplicado foi aplicada para impedir que um médico já cadastrado se cadastre novamente e foi implementada na classe MenuMedico.

**EspecialidadeInvalidaException:** A exceção de Especialidade Invalida foi implementada nas classes Medico e Main para avisar ao usuário quando não houver a especialidade que o mesmo procura, na classe CadastroMedico, o medico precisa informar sua especialidade e essas especialidades são armazenadas, se acaso não existir, a mensagem de erro da exception aparece na tela.

**HorarioIndisponivelException:** A exceção aplica-se para avisar o paciente quando o horário solicitado não estiver disponível, que pode ocorrer devido ao fato de ele já possuir consulta no mesmo horário ou o médico estiver ocupado. Foi implementada na classe Main

**PagamentoPendenteException:** A exceção aplica-se quando o paciente tenta realizar uma consulta, mas possui um pagamento pendente. Foi implementada nas classes Consulta e Paciente.