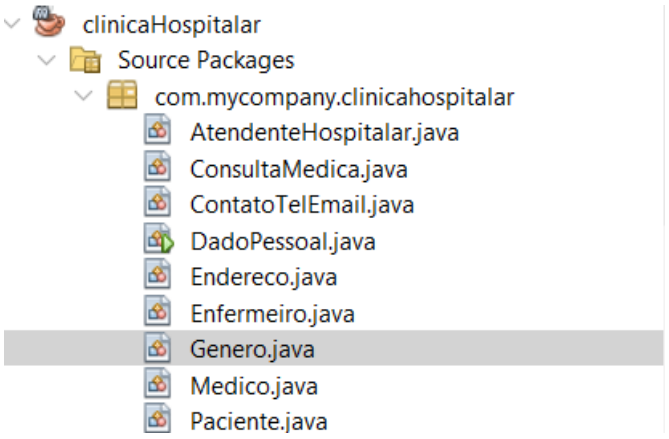
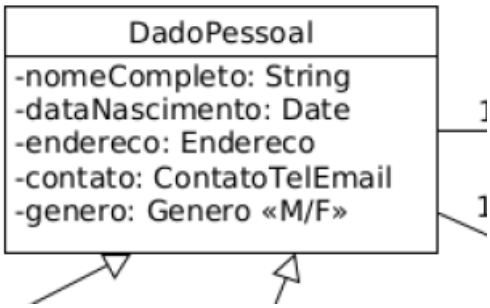
Programação de Computadores III
Aluna: Tuany Daumas Braga Barbosa

Desenvolvimento de um sistema Java com interface gráfica Desktop para o cadastro de Pacientes, Médicos, Enfermeiros e Consultas Médicas.

Estrutura do Projeto:

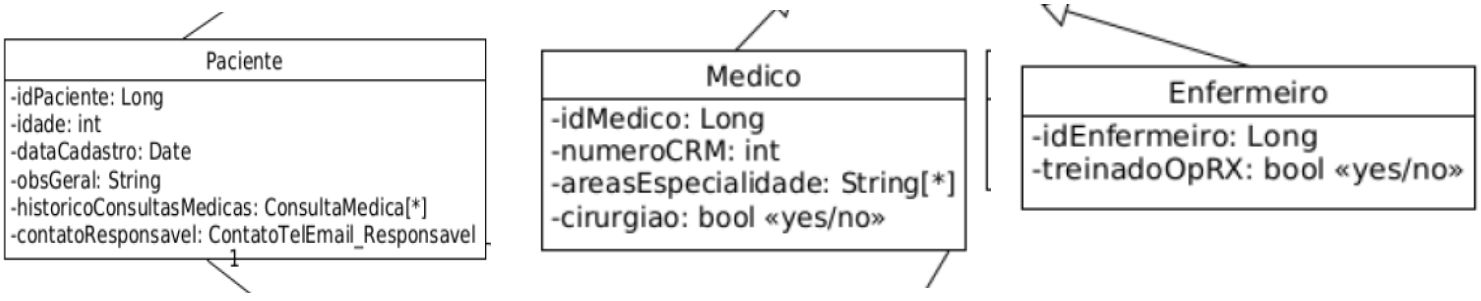
backend/





- **Construtores Especializados:** Foram criados construtores especializados para cada classe **(Paciente, Médico, Enfermeiro)** que inicializam os atributos específicos de cada uma, além dos atributos herdados da classe **DadoPessoal**. Todos os construtores garantem o **cadastro obrigatório de Endereço e Contato.**

```
// Construtor
public DadoPessoal(String nomeCompleto, Date dataNascimento, Endereco endereco, ContatoTelEmail contato, Genero.Tipo genero) {
    this.nomeCompleto = nomeCompleto;
    this.dataNascimento = dataNascimento;
    this.endereco = endereco;
    this.contato = contato;
    this.genero = genero;
}
```



```java
public class Paciente extends DadoPessoal {

    // Atributos
    private final Long idPaciente;
    private final int idade;
    private Date dataCadastro;
    private final String obsGeral;
    private List<ConsultaMedica> historicoConsultasMedicas;
    private ContatoTelEmail contatoResponsavel;


    // Construtor
    public Paciente(String nomeCompleto, Date dataNascimento, Endereco endereco, ContatoTelEmail contato, Genero.Tipo genero,
                    Long idPaciente, int idade, Date dataCadastro, String obsGeral, List<ConsultaMedica> historicoConsultasMedicas,
                    ContatoTelEmail contatoResponsavel) {
        super(nomeCompleto, dataNascimento, endereco, contato, genero);
        this.idPaciente = idPaciente;
        this.idade = idade;
        this.dataCadastro = dataCadastro;
        this.obsGeral = obsGeral;
        this.historicoConsultasMedicas = historicoConsultasMedicas;
        this.contatoResponsavel = contatoResponsavel;
    }
```

```java
public class Medico extends DadoPessoal {
    // Atributos
    private Long idMedico;
    private int numeroCRM;
    private List<String> areasEspecialidade;
    private boolean cirurgiao;

    // Construtor
    public Medico(String nomeCompleto, Date dataNascimento, Endereco endereco, ContatoTelEmail contato, Genero.Tipo genero,
                  Long idMedico, int numeroCRM, List<String> areasEspecialidade, boolean cirurgiao) {
        super(nomeCompleto, dataNascimento, endereco, contato, genero);
        this.idMedico = idMedico;
        this.numeroCRM = numeroCRM;
        this.areasEspecialidade = areasEspecialidade;
        this.cirurgiao = cirurgiao;
    }
```

```java
public class Enfermeiro extends DadoPessoal {
    // Atributos
    private Long idEnfermeiro; // ID do enfermeiro
    private boolean treinadoOpRX; // Indica se o enfermeiro está treinado em operações de radiologia

    // Construtor
    public Enfermeiro(String nomeCompleto, Date dataNascimento, Endereco endereco, ContatoTelEmail contato, Genero.Tipo genero,
                      Long idEnfermeiro, boolean treinadoOpRX) {
        super(nomeCompleto, dataNascimento, endereco, contato, genero);
        this.idEnfermeiro = idEnfermeiro;
        this.treinadoOpRX = treinadoOpRX;
    }
```

- **Relação de Herança e Encapsulamento:** As classes foram organizadas de forma a aproveitar a herança onde apropriado, como a classe **AtendenteHospitalar** que herda de **DadoPessoal.**

```java
public class AtendenteHospitalar extends DadoPessoal {
    // Atributos
    private String setor; // Setor de trabalho do atendente hospitalar
    private int chSemanal; // Carga horária semanal do atendente hospitalar

    // Construtor
    public AtendenteHospitalar(String nomeCompleto, Date dataNascimento, Endereco endereco, ContatoTelEmail contato,
                               Genero.Tipo genero, String setor, int chSemanal) {
        super(nomeCompleto, dataNascimento, endereco, contato, genero);
        this.setor = setor;
        this.chSemanal = chSemanal;
    }
```

- **Cadastro de Consulta Médica:** A classe ConsultaMedica contém referências aos IDs de Paciente e Médico, estabelecendo a relação paciente-médico.

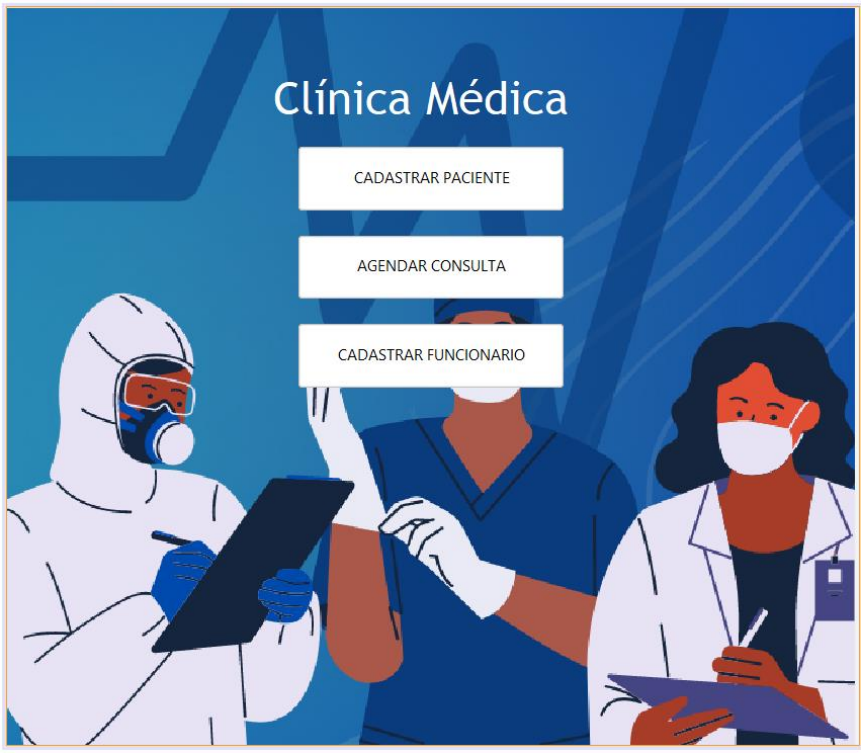
```java
public class ConsultaMedica {
    // Atributos
    private final Long idConsulta;
    private final Paciente paciente;
    private final Medico medico;
    private final String exameQueixa;
    private final String diagnostico;
    private final String prescricao;
    private boolean indicacaoCirurgica;

    // Construtor
    public ConsultaMedica(Long idConsulta, Paciente paciente, Medico medico, String exameQueixa,
            String diagnostico, String prescricao, boolean indicacaoCirurgica) {
        this.idConsulta = idConsulta;
        this.paciente = paciente;
        this.medico = medico;
        this.exameQueixa = exameQueixa;
        this.diagnostico = diagnostico;
        this.prescricao = prescricao;
        this.indicacaoCirurgica = indicacaoCirurgica;
    }

    // Getters e setters
```

- **Classes Acessórias e Enumeração:** classes Endereco, ContatoTelEmail e Genero.

```java
public class Endereco {
    private final String rua;
    private final int numero;
    private final String bairro;
    private final String cidade;
    private final String estado;
    private final int cep;

    public Endereco(String rua, int numero, String bairro, String cidade, String estado, int cep) {
        this.rua = rua;
        this.numero = numero;
        this.bairro = bairro;
        this.cidade = cidade;
        this.estado = estado;
        this.cep = cep;
    }

    // Getters e setters

}
```

```java
public class ContatoTelEmail {
    private final String telefone;
    private final String celular;
    private final String email;

    public ContatoTelEmail(String telefone, String celular, String email) {
        this.telefone = telefone;
        this.celular = celular;
        this.email = email;
    }

    // Getters e setters

}
```

```java
package com.mycompany.clinicahospitalar;

/**
 *
 * @author Tuany
 */
public class Genero {
    enum Tipo {
        M, // Masculino
        F // Feminino
    }
}
```



métodos são responsáveis por controlar o comportamento dos botões na interface do usuário. Quando um botão específico é clicado, o método associado é chamado e executa a lógica necessária para criar e exibir uma nova janela correspondente à funcionalidade desejada (cadastrar paciente, cadastrar funcionário ou visualizar a agenda).

```
    private void btnCadastrarPacienteActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        new DadosPaciente ().setVisible(b: true);
    }

    private void btnCadastrarFuncionarioActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        new DadosFuncionario ().setVisible(b: true);
    }

    private void btnAgendaActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        new TelaAgenda ().setVisible(b: true);
    }
```