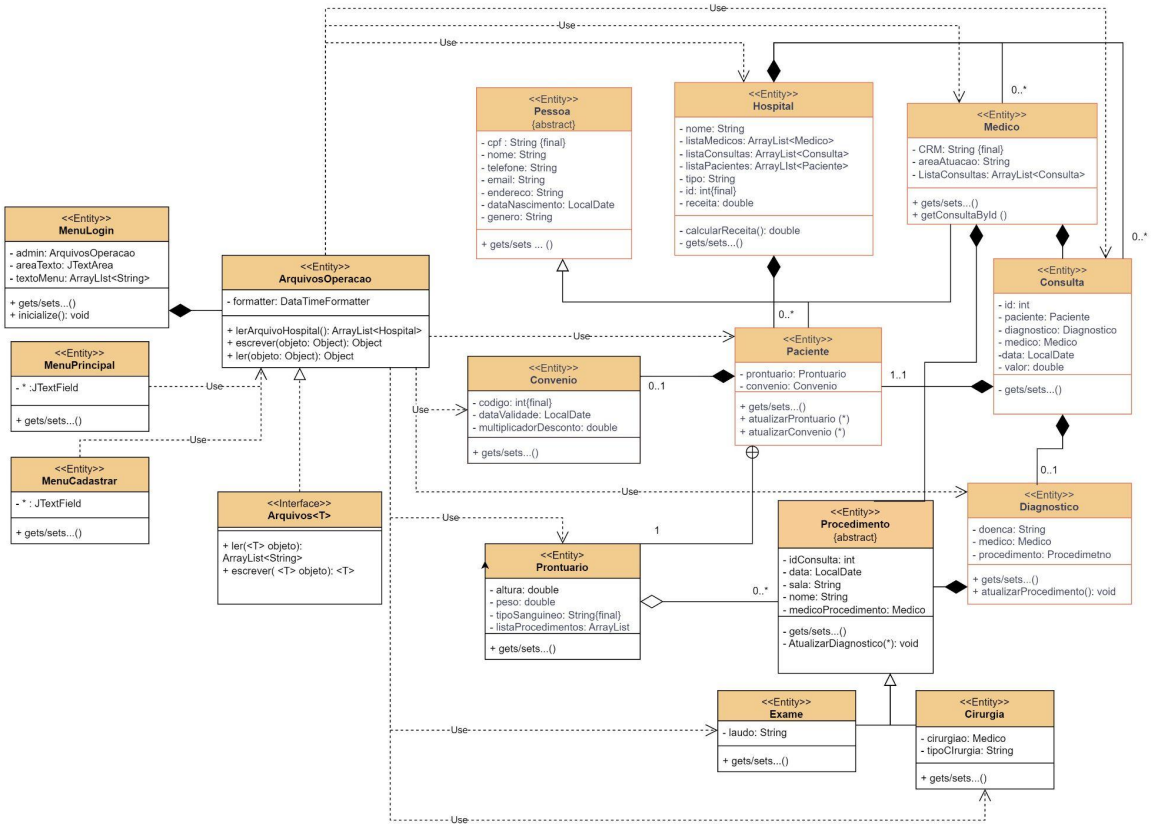
Resumo Projeto Final MC322

Matheus, Pietro e Lucas 29 de Junho de 2023

Estrutura Geral de Classes:

Decidimos por fazer no projeto um sistema hospitalar. Com isso em mente, pensamos em quais classes seriam necessárias, quais métodos, como iam se relacionar para então implementarmos, levando, principalmente, em consideração os conceitos lecionados em aula.



Classe Hospital:

Entre os atributos vistos acima, no diagrama UML, existem listas de pacientes, consultas e médicos, que por sua vez são outras classes desenvolvidas no projeto, formando relações de composição com o Hospital. Ele também possui um atributo referente a receita, que é calculado percorrendo a lista de consultas e somando seus valores, que são calculados utilizando um valor base e o multiplicador de desconto do convênio do paciente.

Pessoa

Essa é a uma outra base do nosso projeto, como classe abstrata tem paciente e médico como suas subclasses e contém as informações genéricas de ambos.

Paciente, Convênio e Prontuário

A classe paciente, subclasse de Pessoa, forma uma relação de composição com a classe convênio, importante, como dito acima, para o valor da consulta. Já o prontuário é uma classe aninhada no paciente com informações médicas e uma lista de objetos procedimentos, que serão citados mais tarde.

Médico

Essa classe, também subclasse da classe abstrata Pessoa, Forma relações de composição com consulta, tendo cada objeto médico uma lista própria de consultas.

Consulta

A classe consulta é o centro do projeto, por ser a premissa marcar consultas e concentra, direta ou indiretamente, relações com boa parte das classes do projeto.

Diagnóstico, Procedimento, Exame e Cirurgia

A classe diagnóstico composta da consulta, pode, ao ser criada, instanciar um procedimento, porém, o procedimento é uma classe abstrata. Então na verdade as classes Exame e Cirurgia que são instanciadas.

Leitura e Escrita de Arquivos

Nós decidimos criar uma interface que vai receber um objeto genérico, que chamamos de T. Nessa interface definimos dois métodos a serem implementados: Ler escrever.

‘ler’ vai pegar o caminho para o arquivo desejado e vai ler linha a linha o csv relativo ao que se busca no momento de chamada do método. Para chegar ao arquivo desejado, utilizamos de um switch case para dividir em um caso para cada classe de Leitura, visto que os arquivos estão divididos em classes, contendo seus atributos.

```java
public class ArquivosOperacao implements Arquivos<Object> {
    protected static DateTimeFormatter formatter = DateTimeFormatter.ofPattern(pattern:"dd/MM/yyyy");

    @Override
    public ArrayList<String> ler(Object objeto) {
        System.out.println(x:"Lendo arquivo: ");
        ArrayList<String> dadosRetorno = new ArrayList<String>();
        String caminho = "arquivosCSV/" + objeto.getClass().getSimpleName() + ".csv";
        try {
            // Instancia um objeto File com o caminho do arquivo CSV
            File file = new File(new File(pathname:".").getCanonicalFile(), caminho);
            try (BufferedReader br = new BufferedReader(new FileReader(file))) {
                String linha = br.readLine();
                String[] cabecalho = linha.split(regex:",");

                String linhaRetorno = "";
                switch (objeto.getClass().getSimpleName()) {
                    case "Paciente":
                        // Imprime o conteúdo de Paciente.csv no menu
                        while ((linha = br.readLine()) != null) {
                            String[] dados = linha.split(regex:",");
                            for (int i = 0; i < dados.length; i++) {
                                linhaRetorno += cabecalho[i] + ": " + dados[i] + "  ";
                            }
                            linhaRetorno += "\n";
                            dadosRetorno.add(linhaRetorno);
                        }
                        break;

                    case "Medico":
                        // Imprime o conteúdo de Medico.csv no menu
```

Já o método escrever é um pouco diferente, pois ele recebe um objeto como parâmetro. Checamos qual classe é o objeto, acessamos os dados relativo a essa classe com um switch e escrevemos.

```java
@Override
public Object escrever(Object objeto) {

    System.out.println(x:"Escrevendo arquivo: ");
    String caminho = "arquivosCSV/" + objeto.getClass().getSimpleName() + ".csv";

    try (BufferedWriter writer = new BufferedWriter(new FileWriter(caminho,
            append:true))) {
        StringBuilder sb = new StringBuilder();

        switch (objeto.getClass().getSimpleName()) {
            case "Paciente":
                // CPF_PESSOA,NOME_PESSOA,TELEFONE,ENDERECO,EMAIL,DATA_NASCIMENTO,GENERO,ALTURA,PESO,TIPO_SANGUINEO,CODIGO_CONVENIO
                Paciente paciente = (Paciente) objeto;

                sb.append(paciente.getCpf() + "," + paciente.getNome() + "," + paciente.getTelefone() + ","
                        + paciente.getEndereco() + "," + paciente.getEmail() + "," + paciente.getDataNascimento()
                        + "," + paciente.getGenero() + "," + paciente.getProntuario().getAltura() + ","
                        + paciente.getProntuario().getPeso() + "," + paciente.getProntuario().getTipoSanguinio());

                writer.write(sb.toString());
                writer.newLine();
                break;

            case "Medico":
```

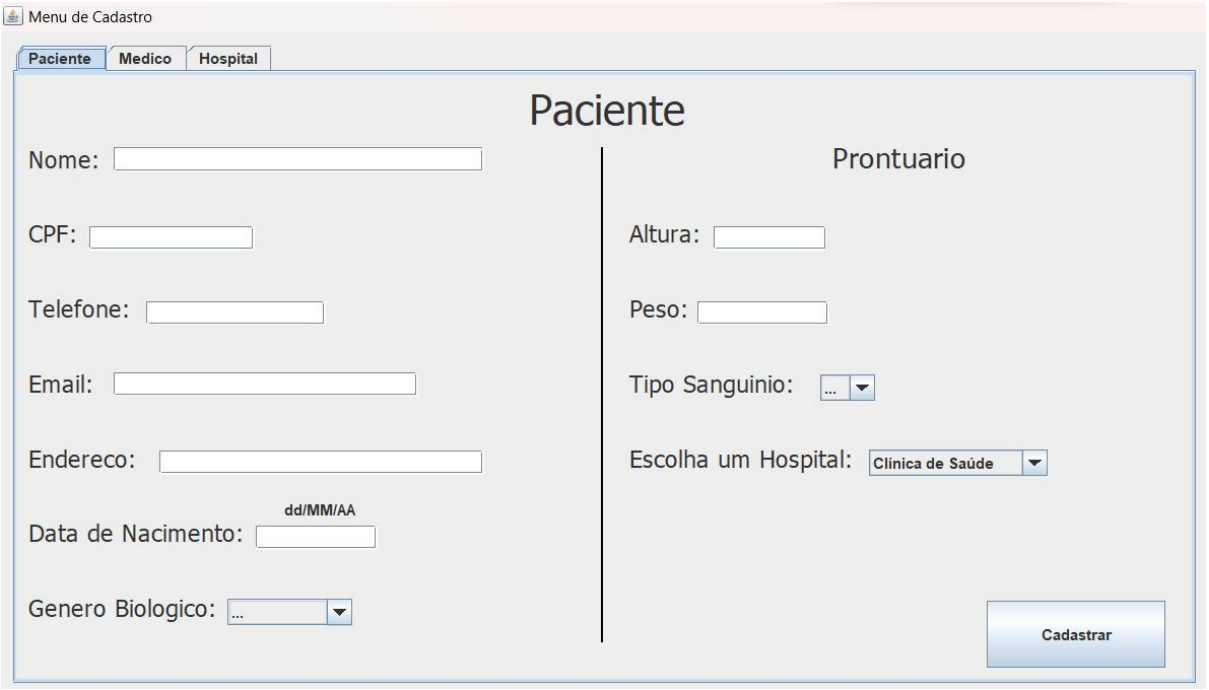
Na classe Hospital tem um array de médico, pacientes e consultas, que vão ser utilizados somente no menu. Porém foi criado também um método para leitura de arquivos chamada “LerArquivosHospitais”, nela serão lidos todos os arquivos e serão instanciados no objeto hospital, para isso, em todas as nossa classes principais são gerados ID’s aleatórios, que são verificados a cada novo ID criado, dessa forma a relacionar corretamente os objetos.

```java
public class Hospital {
    ArrayList<Medico> listaMedicos;
    ArrayList<Paciente> listaPacientes;
    ArrayList<Consulta> listaConsultas;
    String tipo;
    final String cnpj;
    final int id;
    String nome;
    double receita;

    // Hospital sem id, gera id aleatório e único
    public Hospital(String tipo, String nome, String cnpj) {
        ArquivosOperacao admin = new ArquivosOperacao();
        Random random = new Random();
        int id = random.nextInt(bound:1000);

        // Gera id aleatorio, checa se já existe no CSV
        ArrayList<String> listaIdHospital = admin.ler(new Hospital());
        while (listaIdHospital.contains(Integer.toString(id))) {
            id = random.nextInt(bound:1000);
        }
        this.id = id;

        this.listaMedicos = new ArrayList<Medico>();
        this.listaPacientes = new ArrayList<Paciente>();
        this.listaConsultas = new ArrayList<Consulta>();
        this.tipo = tipo;
        this.nome = nome;
        this.cnpj = cnpj;
        this.receita = 0.0;
    }
```

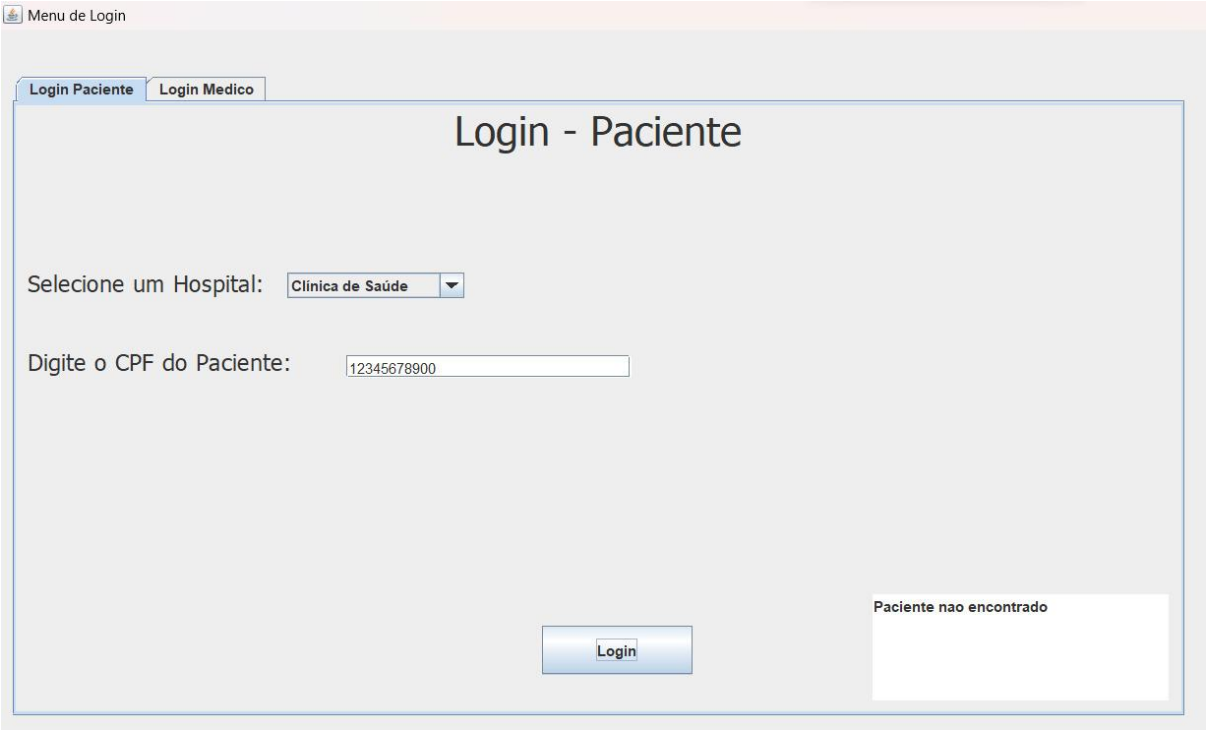
Interface Gráfica

De início o usuário se depara com o menu principal em que tem as opções de listar informações, cadastrar e a tela de login.

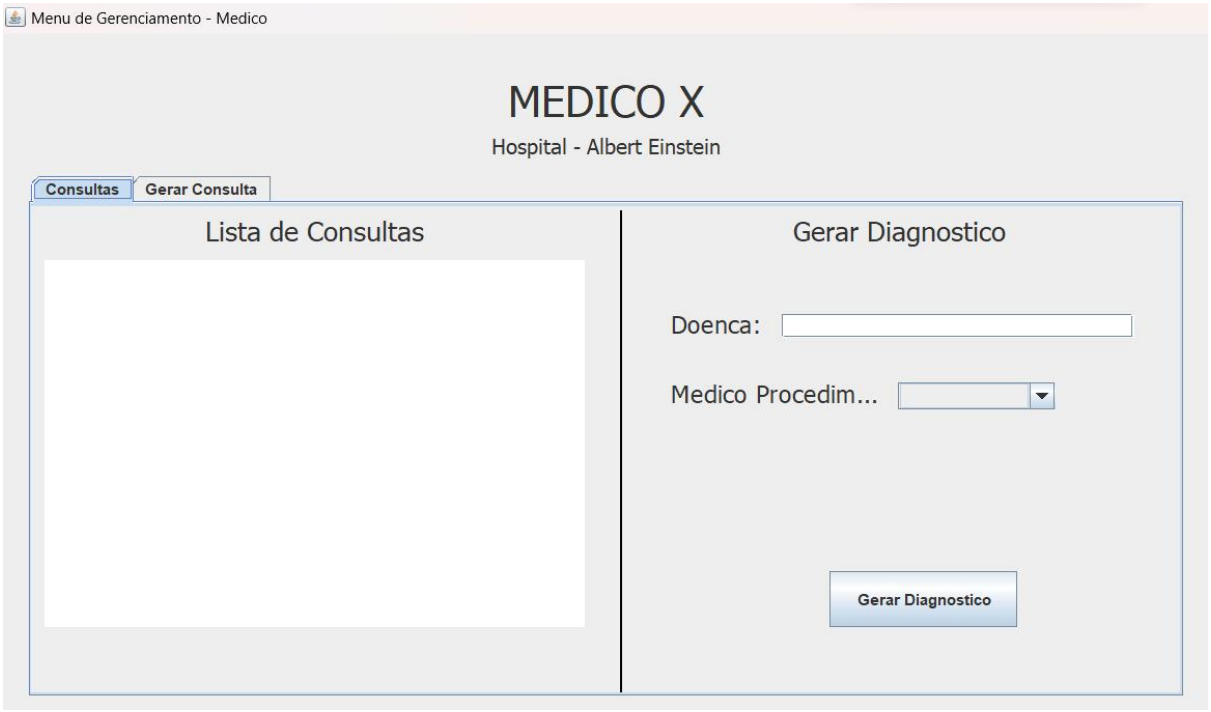
No menu de cadastro permite o usuário instanciar três tipos diferentes de objetos, que serão registrados nos arquivos csv.



Na tela de login o usuário pode escolher entrar como paciente ou como médico:



Caso o usuário seja um médico, ele entrará em menu de cadastro para acessar e gerar consultas, além de poder criar procedimentos e diagnósticos:



Do contrário, ele será um paciente e terá acesso ao seu prontuário: