Programação Orientada por Objetos

Projeto 2 – "Centro Médico"

Docentes:

- Mónica Cameirão
- Sergi Bermúdez

Alunos:

- Joel Camacho, nº 2082516
- Juan Silva, nº 2083516
- Ricardo Lucas, nº 2040416

25 de maio de 2018

Índice

Resumo

Com este relatório pretende-se demonstrar como foi elaborado este segundo projeto da disciplina, dando relevo às dificuldades e soluções que encontramos no seu desenvolvimento. Com o intuito de não tornar este relatório muito extenso, vamos falar de maneira geral, de cada uma das classes apresentando com devida atenção e detalhe as principais classes e os principais métodos.

Em anexo a este relatório está também o código em java do projeto, devidamente organizado e comentado para que o código se torne de fácil compreensão e modificação, no futuro.

Introdução

Neste projeto desenvolveu-se, em linguagem java, um centro médico que permite gerir diferentes atos médicos realizados pelos utentes do centro médico. Este centro médico permite:

- registar novos médicos e utentes, atribuindo aleatoriamente um número único;
- listar todos os utentes registados, estando ordenados, ou não, de forma ascendente ou descendente pelo valor que gastaram;
- listar todos os médicos registados, estando ordenados pela classificação;
- listar todas as consultas do centro médico;
- listar os utentes que determinado médico consultará num determinado dia;
- avançar um dia de cada vez do calendário;
- realizar novas consultas;
- atribuir uma avaliação a um médico;
- fazer pagamentos por parte dos utentes;
- imprimir o valor total que o centro médico recebeu;
- imprimir o número total de consultas realizadas no centro médico;
- sair da aplicação;
- inserir informações do centro médico através do teclado ou através de ficheiros de texto, de forma automática;

Neste relatório é apresentado como a aplicação funciona de uma maneira geral, dando mais relevo ao código desenvolvido.

Desenvolvimento

Neste projeto foram desenvolvidas quatro classes, destinadas aos médicos, aos utentes, às consultas e ao menu.

Classe Utentes

Na classe "Utentes" está presente um construtor que recebe como argumentos um nome, uma idade e um número único, o valor pago é inicializado a zero. Está presente os métodos "*getters*" e "*setters*" para o nome, número único, idade, seguro e para o valor pago. Há um *override* do método "*equals*" que verifica se o número de utente gerado aleatoriamente já está em uso, há um outro *override* do método "*toString*" que mostra os dados do utente instanciado.

Classe Médicos

O construtor da classe "Medicos" recebe como argumentos um número único e uma especialidade, a avaliação media é inicializado a zero e é criado uma lista para o médico instanciado que guarda todas as classificações atribuídas. Existem métodos "*getters*" e "*setters*" para o número único, especialidade, avaliação e para a avaliação média. Está definido um método "*media*" que é responsável por calcular e atualizar a média das classificações atribuídas. Tal como na classe "Utentes" há um *override* do método "*equals*" que verifica se o número único gerado aleatoriamente já está em uso e há outro *override* do método "*toString*" que mostra os dados do médico instanciado.

Classe Consultas

O construtor da classe "Consultas" recebe como argumentos um médico, um utente, o dia pretendido para a consulta. Está presente nesta classe métodos "*getters*" e "*setters*" para o médico, o utente, a data da consulta, a data do exame, a data do pagamento, a data da consulta de resultados e para o booleano que decide se o médico pediu exame ou não. Nesta classe está também definido o método "Exames", que decide de forma aleatória se é necessário fazer exame ou não e o método "pagamento", que decide qual o valor a pagar e, no caso de ter seguro ou idade superior a 65 anos, efetua o desconto correspondente. Tal como na classe "Medicos" e "Utentes há um *override* do método "*toString*" que mostra os dados da consulta instanciada.

Classe Menu

A classe “Menu” começa por verificar se existem os ficheiros que contêm os médicos e os pacientes, caso estes ficheiros existam é lido e colocado na lista adequada cada um dos dados presentes no ficheiro, logo de seguida é apresentado ao utilizador da aplicação uma mensagem de boas vindas, a data atual e o menu principal do centro médico.

No menu principal estão disponíveis três opções sendo uma para sair da aplicação, uma para entrar no menu de utente e outra para entrar no menu de administrador.

Ao selecionar o menu de utente é apresentada uma lista dos utentes registados em que o utilizador deverá selecionar o utente pretendido, após essa escolha é apresentada uma lista dos médicos registados, e a sua especialidade, em que, novamente, o utilizador deverá selecionar o médico pretendido. De seguida é apresentado um menu onde o utilizador deverá selecionar se pretende marcar uma consulta, ver as consultas marcadas ou avaliar um médico.

Ao selecionar o menu de administrador é necessária a introdução de uma *password*, de forma a que só quem a sabe tenha acesso a esse menu, estando está definida com o valor “12345”. Após ser introduzida a *password* correta é apresentado o menu de administrador que permite registar e consultar médicos e utentes, listar as consultas e avançar a data do calendário.

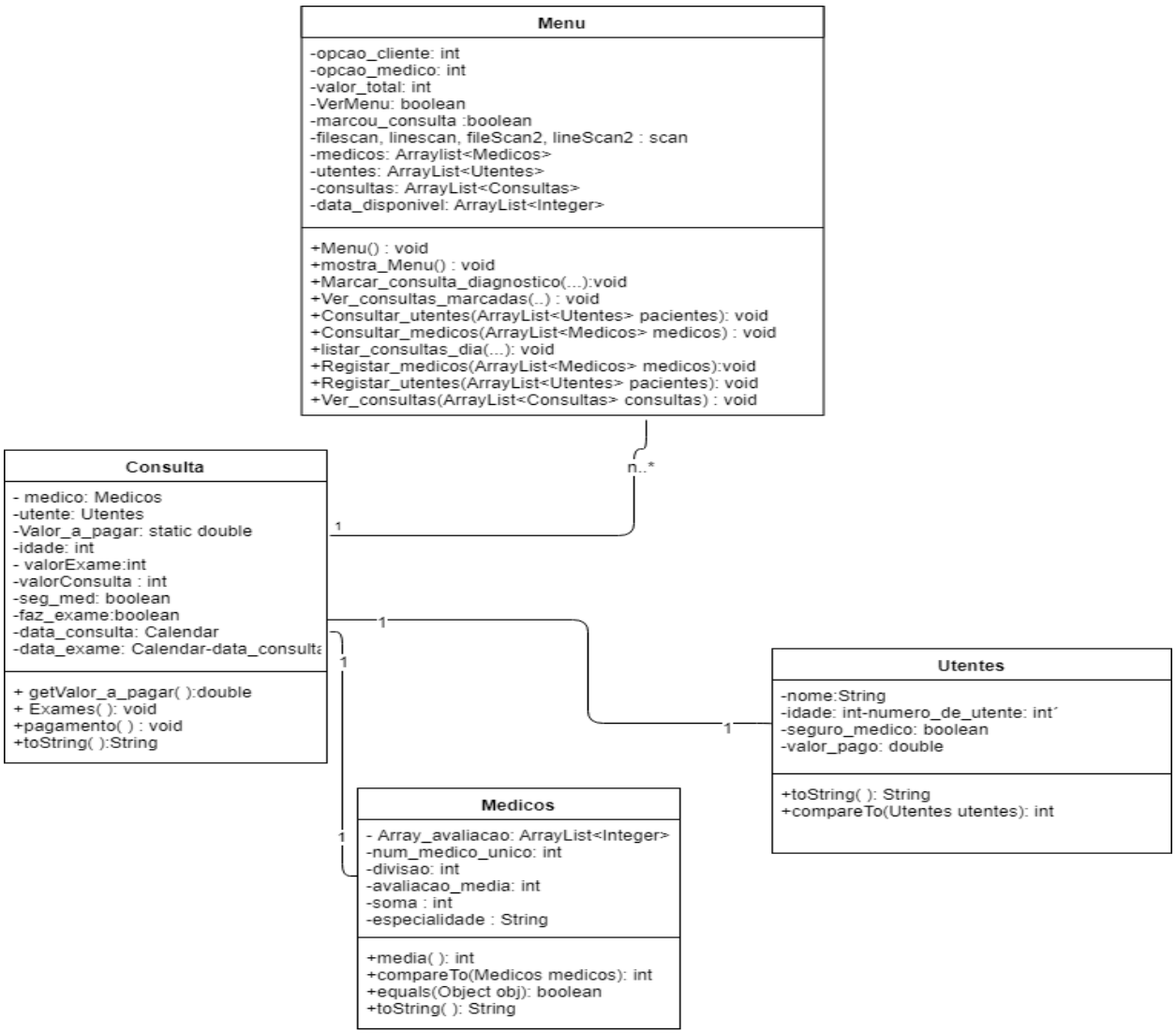
Ainda, nesta classe, estão definidos os seguintes métodos:

- **“Marcar_consulta_diagnostico”**, que marca uma consulta no dia pretendido caso o médico selecionado não faça mais que 5 consultas diárias e o utente não tenha mais que 4 consultas por semana.
- **“Ver_consultas_marcadas”**, que mostra ao utilizador se tem, ou não, consultas marcadas e o respetivo dia.
- **“Consultar_utentes”**, que lista, de forma ordenada ou não, todos os utentes registados no centro médico.
- **“Consultar_medicos”**, que lista todos os médicos registados no centro médico.
- **“listar_consultas_dia”**, que lista todas as consultas que determinado médico efetua num determinado dia.
- **“Registar_medicos”** e **“Registar_utentes”**, que, tal como o nome indica, adiciona ao centro médico um novo médico e um novo utente, respetivamente.
- **“Ver_consultas”**, que lista todas as consultas do centro médico.

Conclusão

Durante o desenvolvimento deste projeto surgiram algumas dificuldades, nomeadamente nas datas, nas avaliações dos médicos, entre outras, que foram superadas. Durante o desenvolvimento do projeto foi testada cada uma das classes e cada um dos métodos de forma a verificar se funcionavam como o pretendido.

Concluiu-se que este projeto foi muito gratificante visto que aprofundamos os nossos conhecimentos sobre a linguagem java e sobre programação orientada por objetos, e, permitiu-nos, de uma forma mais espontânea, colocar em prática muitos dos conceitos transmitidos no decorrer das aulas, quer práticas quer teóricas.



Anexos

```
package com.projeto_2;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.*;

public class Menu {

    private int avaliacao;
    private int opcao_paciente;
    private int opcao_medico;

    private double valor_total = 0;

    private boolean VerMenu = true;
    private boolean marcou_Consulta;

    private Scanner scan = new Scanner( System.in );
    private Scanner fileScan, lineScan, fileScan2, lineScan2;

    private Calendar data = Calendar.getInstance();
    private Calendar data_marcacao = Calendar.getInstance();

    private ArrayList<Medicos> medicos = new ArrayList<>();
    private ArrayList<Utentes> utentes = new ArrayList<>();
    private ArrayList<Consultas> consultas = new ArrayList<>();
    private ArrayList<Integer> data_disponivel = new ArrayList<>();

    public Menu( ) throws IOException {
```

```
/*
   VERIFICA SE O FICHEIRO MEDICOS.TXT EXISTE, E CASO EXISTA COPIA O CONTEÚDO DO FICHEIRO PARA A LISTA medicos
   LINHA A LINHA, CASO NÃO EXISTA MOSTRA MENSAGEM QUE NÃO FORAM IMPORTADOS MEDICOS
 */

FileInputStream file_medicos = null;
try {
   file_medicos = new FileInputStream( "Medicos.txt" );

   fileScan2 = new Scanner( file_medicos );

   while (fileScan2.hasNext()) {
      String linha2 = fileScan2.nextLine();
      lineScan2 = new Scanner( linha2 );

      int num_medico_unico = Integer.parseInt( lineScan2.next() );
      String especialidade = lineScan2.next();

      medicos.add( new Medicos( num_medico_unico, especialidade ) );
   }
} catch (FileNotFoundException ex) {
   System.out.println( "Nao foi possivel importar nenhuma lista de medicos" );
}

/*
   VERIFICA SE O FICHEIRO UTENTES.TXT EXISTE, E CASO EXISTA COPIA O CONTEÚDO DO FICHEIRO PARA A LISTA utentes
```

```
        LINHA A LINHA, CASO NÃO EXISTA MOSTRA MENSAGEM QUE NÃO FORAM
IMPORTADOS UTENTES
        */

      FileInputStream file_pacientes = null;
      try {
         file_pacientes = new FileInputStream( "Utentes.txt" );

         fileScan = new Scanner( file_pacientes );

         while (fileScan.hasNext()) {
            String linha = fileScan.nextLine();
            lineScan = new Scanner( linha );

            String nome = lineScan.next();
            int idade = Integer.parseInt( lineScan.next() );
            int numero_de_utente = Integer.parseInt( lineScan.next() );
            boolean seguro_medico = lineScan.nextBoolean();
            utentes.add( new Utentes( nome, idade, numero_de_utente, seguro_medico ) );
         }
      } catch (FileNotFoundException ex) {
         System.out.println( "Nao foi possivel importar nenhuma lista de utentes\n" );
      }

      //APÓS AS IMPORTAÇÕES É APRESENTADA UMA MENSAGEM DE BOAS VINDAS E A DATA

      System.out.println( "Bem vindo ao centro medico !!" );
      System.out.println( "Data:  " + data.get( Calendar.DAY_OF_MONTH ) + "/" + data.get(
Calendar.MONTH ) + "/" + data.get( Calendar.YEAR ) );

      //INVOCA O MÉTODO QUE MOSTRA O MENU PRINCIPAL
      mostra_Menu();
```

```
    }

    public void mostra_Menu( ) throws IOException {
        //EQUANTO vermenu FOR VERDADEIRO MOSTRA O MENU PRINCIPAL
        while (VerMenu) {

            System.out.println( "\n Menu" + "\n 1-- Administrador " + "\n 2-- Utente " + "\n 3-- Sair"
);

            switch (scan.nextInt()) {
                case 1: //SELECIONADO O MENU ADMINISTRADOR E É PEDIDA UMA PASSWORD
                    System.out.println( "Escreva a password do administrador" );
                    int password = scan.nextInt();

                    if (password == 12345) {
                        Menu_Admin(); //MOSTRA MENU DE ADMINISTRADOR CASO A PASSWORD
INTRODUZIDA ESTEJA CORRETA
                    } else System.out.println( "Password incorreta" );
                    break;
                case 2: //MOSTRA O MENU DE UTENTE
                    Menu_Utente();
                    break;
                case 3:
                    VerMenu = false; //SAI DA APLICAÇÃO, NÃO MOSTRA MENU
                    break;
                default:
                    System.out.println( "Opcao errada !" );
            }
        }
    }

    public void Menu_Admin( ) throws IOException {
```

```
boolean verMenu2 = true;
//EQUANTO vermenu2 FOR VERDADEIRO MOSTRA O MENU DE ADMINISTRADOR
while (verMenu2) {

   System.out.println( "Data:  " + data.get( Calendar.DAY_OF_MONTH ) + "/" + data.get( Calendar.MONTH ) + "/" + data.get( Calendar.YEAR ) );
   System.out.println( "\n Administrador" +
      "\n 1 -- Registar medicos" +
      "\n 2 -- Registar utentes" +
      "\n 3 -- Consultar medicos" +
      "\n 4 -- Consultar utentes" +
      "\n 5 -- Ver consultas" +
      "\n 6 -- Listar os utentes que um determinado médico consultará num determinado dia" +
      "\n 7 -- Avançar no calendario" +
      "\n 8 -- Voltar ao Inicio" +
      "\n Valor total ganho: " + valor_total +
      "\n Numero total de consultas: " + consultas.size() );

   switch (scan.nextInt()) {
      case 1:
         Registar_medicos( medicos );
         break;
      case 2:
         Registar_utentes( utentes );
         break;
      case 3:
         Consultar_medicos( medicos );
         break;
      case 4:
         Consultar_utentes( utentes );
```

```
            break;
        case 5:
            Ver_consultas( consultas );
            break;
        case 6:
            listar_consultas_dia( medicos, utentes, consultas );
            break;
        case 7:
            //AVANÇA UM DIA
            data.set( Calendar.DAY_OF_MONTH, (data.get( Calendar.DAY_OF_MONTH ) + 1) );
            double valor_cobrado_neste_dia = 0;

            for (int i = 0; i < consultas.size(); i++) {
                /*VERIFICA SE O DIA ATUAL (DEPOIS DE TER AVANÇADO EM CIMA)
CORRESPONDE A DATA DE PAGAMENTO DE ALGUMA DAS CONSULTAS
                SE CORRESPONDER:
                 */
                if (data.get( Calendar.DAY_OF_MONTH ) == consultas.get( i
).getData_pagamento().get( Calendar.DAY_OF_MONTH )
                        && data.get( Calendar.MONTH ) == consultas.get( i
).getData_pagamento().get( Calendar.MONTH ))
                {
                    //INCREMENTA O VALOR TOTAL DO CENTRO MEDICO E INCREMENTA O VALOR
A PAGAR NO DIA
                    valor_total = valor_total + consultas.get( i ).getValor_a_pagar();
                    valor_cobrado_neste_dia = valor_cobrado_neste_dia + consultas.get( i
).getValor_a_pagar();

                    //PERCORRE A LISTA DE MEDICOS
                    for (int j = 0; j < medicos.size(); j++) {
                        //QUANDO ENCONTRA O MEDICO DA CONSULTA COBRADA EM CIMA
                        if (medicos.get( j ).getNum_medico_unico() == consultas.get( i
).getMedico().getNum_medico_unico()) {
```

```
                    if (data.get( Calendar.DAY_OF_MONTH ) == consultas.get( i
).getData_consulta().get( Calendar.DAY_OF_MONTH )

                        && data.get( Calendar.MONTH ) == consultas.get( i
).getData_consulta().get( Calendar.MONTH )) {


                        //ATRIBUI UMA AVALIAÇÃO ALEATÓRIA AO MEDICO E INFORMA A
MÉDIA ATUALIZADA

                        medicos.get( j ).setAvaliacao( new Random().nextInt( 6 ) );

                        System.out.println( "A avaliação do medico foi " + medicos.get( j
).getAvaliacao_media() );


                    }

                    //VERIFICA SE CORRESPONDE A UMA CONSULTA DE RESULTADOS

                    if (data.get( Calendar.DAY_OF_MONTH ) == consultas.get( i
).getData_consulta_resultados().get( Calendar.DAY_OF_MONTH )

                        && data.get( Calendar.MONTH ) == consultas.get( i
).getData_consulta_resultados().get( Calendar.MONTH )) {


                        //ATRIBUI UMA AVALIAÇÃO ALEATÓRIA AO MEDICO E INFORMA A
MÉDIA ATUALIZADA

                        medicos.get( j ).setAvaliacao( new Random().nextInt( 6 ) );

                        System.out.println( "A avaliação do medico foi " + medicos.get( j
).getAvaliacao_media() );


                    }
                }
            }
        }
    }


    //INFORMA AO UTILIZADOR O VALOR COBRADO

    System.out.println( "Foi cobrado um valor de " + valor_cobrado_neste_dia + " €
hoje !!" );
```

```
                break;
            case 8:
                //VOLTA AO MENU PRINCIPAL E MOSTRA A DATA E A MENSAGEM DE BOAS VINDAS
                System.out.println( "Bem vindo ao centro medico !!" );
                System.out.println( "Data:  " + data.get( Calendar.DAY_OF_MONTH ) + "/" +
data.get( Calendar.MONTH ) + "/" + data.get( Calendar.YEAR ) );

                mostra_Menu();
                break;
            default:
                System.out.println( "Opcao errada !" );
        }
    }
  }

  public void Menu_Utente( ) throws IOException {

    int apontador = 0;

    data_disponivel.clear();
    System.out.println( "Selecione o seu numero de utente:\n" );

    //LISTA TODOS OS UTENTES
    for (int n = 0; n < utentes.size(); n++)
       System.out.println( n + " --> " + utentes.get( n ).getNumero_de_utente() );

    opcao_paciente = scan.nextInt();

    System.out.println( "Qual o medico que pretende:\n" );

    //LISTA TODOS OS MEDICOS
```

```
for (int n = 0; n < medicos.size(); n++)

   System.out.println( n + " --> " + medicos.get( n ).getNum_medico_unico() + " | Especialidade --> " + medicos.get( n ).getEspecialidade() );


opcao_medico = scan.nextInt();


//INFORMA O NOME DO UTENTE SELECIONADO

System.out.println( "\n Utente: " + utentes.get( opcao_paciente ).getNome() + "\n" );


int numeroVezes = 0;


boolean verMenu3 = true;


while (verMenu3) { //EQUANTO VERMENU3 FOR VERDADEIRO MOSTRA O MENU DE UTENTE


   System.out.println( " Menu Utente" +

         "\n 1-- Marcar consulta_diagnostico " +

         "\n 2-- Ver consultas marcadas" +

         "\n 3-- Avaliar medico" +

         "\n 4-- Voltar atras" );


   switch (scan.nextInt()) {

      case 1:

         Marcar_consulta_diagnostico( data, medicos, utentes, consultas, data_marcacao, data_disponivel, opcao_medico, opcao_paciente, marcou_Consulta, data );


         //CASO A CONSULTA A MARCAR CORRESPONDA AO DIA ATUAL:

         if (consultas.get( consultas.size() - 1 ).getData_pagamento().get( Calendar.DAY_OF_MONTH ) == data.get( Calendar.DAY_OF_MONTH )

               && consultas.get( consultas.size() - 1 ).getData_pagamento().get( Calendar.MONTH ) == data.get( Calendar.MONTH ))

         {
```

```
                //INCREMENTA VALOR TOTAL A PAGAR (PAGA CONSULTA NA HORA)
                valor_total += consultas.get( consultas.size() - 1 ).getValor_a_pagar();



                for (int j = 0; j < utentes.size(); j++) {
                  if (utentes.get( j ).getNumero_de_utente() == consultas.get( consultas.size() - 1
).getUtente().getNumero_de_utente()) {
                    utentes.get( j ).setValor_pago( consultas.get( consultas.size() - 1
).getValor_a_pagar() );


                  }
                }
              }


              //CASO A CONSULTA A MARCAR CORRESPONDA AO DIA ATUAL:
              if (data.get( Calendar.DAY_OF_MONTH ) == consultas.get( consultas.size()-1
).getData_consulta().get( Calendar.DAY_OF_MONTH )
              && data.get( Calendar.MONTH ) == consultas.get( consultas.size()-1
).getData_consulta().get( Calendar.MONTH ))
              {
                //ATRIBUI UMA CLASSIFICAÇÃO ALEATÓRIA AO MEDICO DE UMA CONSULTA
                int rand = new Random().nextInt( 6 );


                consultas.get( consultas.size() - 1 ).getMedico().setAvaliacao( rand );


                System.out.println( rand );


                System.out.println( "A avaliação do medico foi " + medicos.get( opcao_medico
).getArray_avaliacao() );


              }
              //CASO A CONSULTA A MARCAR CORRESPONDA AO DIA ATUAL:
```

```
            if (data.get( Calendar.DAY_OF_MONTH ) == consultas.get( consultas.size()-1
).getData_consulta_resultados().get( Calendar.DAY_OF_MONTH )

                && data.get( Calendar.MONTH ) == consultas.get( consultas.size()-1
).getData_consulta_resultados().get( Calendar.MONTH )

                && consultas.get( consultas.size()-1 ).isFaz_exame())

            {

              //ATRIBUI UMA CLASSIFICAÇÃO ALEATÓRIA AO MEDICO DE UMA CONSULTA DE
RESULTADOS

              int rand = new Random().nextInt( 6 );


              consultas.get( consultas.size() - 1 ).getMedico().setAvaliacao( rand );


              System.out.println( "A avaliação do medico foi " + medicos.get( opcao_medico
).getArray_avaliacao() );


            }

            marcou_Consulta = true;


            break;

          case 2:

            try {

              Ver_consultas_marcadas(marcou_Consulta, utentes, consultas, opcao_paciente,
consultas.get(consultas.size() - 1).isFaz_exame());

            }

            catch ( ArrayIndexOutOfBoundsException ioe){

              System.out.println("Nao fez consultas com este medico\n ");

            }

            break;

          case 3:

            try {

              //CASO A CONSULTA A MARCAR CORRESPONDA AO DIA ATUAL:

              if (data.get(Calendar.DAY_OF_MONTH) == consultas.get(consultas.size() -
1).getData_consulta().get(Calendar.DAY_OF_MONTH)
```

```
                    && data.get(Calendar.MONTH) == consultas.get(consultas.size() -
1).getData_consulta().get(Calendar.MONTH)) {


                    //É PEDIDO AO UTENTE PARA AVALIAR O MEDICO EM QUESTÃO
                    System.out.println("Avalie o medico de 0 a 5 ");
                    int next = scan.nextInt();
                    if( next <= 5 && next >=0 )
                       consultas.get(consultas.size() - 1).getMedico().setAvaliacao(next);
                    else
                       System.out.println("Valor incorrecto");



                  } else
                    System.out.println("Nao fez consultas com este medico ");
                }
                catch(ArrayIndexOutOfBoundsException ioe) {
                  System.out.println("Nao fez consultas com este medico\n ");
                }
                break;
             case 4:
                //VOLTA A MOSTRAR O MENU PRINCIPAL, A DATA ATUAL E A MENSAGEM DE BOAS
VINDAS
                System.out.println( "Bem vindo ao centro medico !!" );
                System.out.println( "Data:  " + data.get( Calendar.DAY_OF_MONTH ) + "/" +
data.get( Calendar.MONTH ) + "/" + data.get( Calendar.YEAR ) );
                verMenu3 = false;
                mostra_Menu();
                break;
             default:
                System.out.println( "Opcao errada !" );
          }
```

```
    }
  }

  public void Marcar_consulta_diagnostico(Calendar calendar, ArrayList<Medicos> medicos,
ArrayList<Utentes> pacientes, ArrayList<Consultas> consultas, Calendar data_marcacao,
ArrayList<Integer> data_disponivel, int opcao_medico, int opcao_paciente, boolean
marcou_Consulta, Calendar data) {

    int data_repetida = 0;

    System.out.println( "Selecione o dia que pretede:" );

    for (int j = 0; j < 8; j++) { //VERIFICA AS CONSULTAS DOS PROXIMOS 7 DIAS
      for (int n = 0; n < consultas.size(); n++) {
        //CONTA O NUMERO DE CONSULTAS DE DETERMINADO MEDICO NO DETERMINADO
DIA
        if (consultas.get( n ).getData_consulta().get( Calendar.DAY_OF_MONTH ) == (data.get(
Calendar.DAY_OF_MONTH ) + j)
            || consultas.get( n ).getData_consulta_resultados().get(
Calendar.DAY_OF_MONTH ) == (data.get( Calendar.DAY_OF_MONTH ) + j)
            && consultas.get( n ).getMedico().getNum_medico_unico() == medicos.get(
opcao_medico ).getNum_medico_unico()) {

          data_repetida = data_repetida + 1;
        }
      }
      //SE O NUMERO DE CONSULTAS FOR IGUAL OU INFERIOR A 5, TUDO BEM
      if (data_repetida < 6) {
        data_marcacao.set( Calendar.DAY_OF_MONTH, (data.get( Calendar.DAY_OF_MONTH
) + j) );
        System.out.println( data_marcacao.get( Calendar.DAY_OF_MONTH ) + "/" +
data_marcacao.get( Calendar.MONTH ) + "/" + data_marcacao.get( Calendar.YEAR ) );
        data_disponivel.add( (data.get( Calendar.DAY_OF_MONTH ) + j) );
```

```
            }
            data_repetida = 0;
        }

        int opcao_data = scan.nextInt();

        //ENQUANTO NAO INTRODUZIR UM DIA VÁLIDO, FICA A PEDIR UM NOVO DIA
        while (opcao_data > data_disponivel.get( data_disponivel.size() - 1 ) || opcao_data < data_disponivel.get( 0 )) {
            System.out.println( "OPCAO ERRADA" );
            opcao_data = scan.nextInt();
        }

        int dia_pretendidio = opcao_data;

        data_marcacao.set( Calendar.DAY_OF_MONTH, dia_pretendidio );

        consultas.add( new Consultas( medicos.get( opcao_medico ), pacientes.get( opcao_paciente ), dia_pretendidio ) );

        System.out.print( consultas.get( consultas.size() - 1 ).toString() );


    }

    public void Ver_consultas_marcadas(boolean marcou_Consulta, ArrayList<Utentes> utente, ArrayList<Consultas> consultas, int opcao_paciente, boolean FezExame) {
        try {
            //VERIFICA SE DETERMINADO UTENTE TEM CONSULTA MARCADA E EXAME MARCADO
            if (marcou_Consulta == true && FezExame == true) {

                for (int i = 0; i < consultas.size(); i++) {
```

```
            if (consultas.get(i).getUtente().getNumero_de_utente() ==
utente.get(opcao_paciente).getNumero_de_utente()) {


                System.out.println("\n Tem consulta Diagnostico dia: " +
consultas.get(i).getData_consulta().get(Calendar.DAY_OF_MONTH) + "/" +
consultas.get(i).getData_consulta().get(Calendar.MONTH) + "/" +
consultas.get(i).getData_consulta().get(Calendar.YEAR));

                System.out.println(" Tem exame dia: " +
consultas.get(i).getData_exame().get(Calendar.DAY_OF_MONTH) + "/" +
consultas.get(i).getData_exame().get(Calendar.MONTH) + "/" +
consultas.get(i).getData_exame().get(Calendar.YEAR));

                System.out.println(" Tem consulta resultados dia: " +
consultas.get(i).getData_consulta_resultados().get(Calendar.DAY_OF_MONTH) + "/" +
consultas.get(i).getData_consulta_resultados().get(Calendar.MONTH) + "/" +
consultas.get(i).getData_consulta_resultados().get(Calendar.YEAR) + "\n");

                System.out.println("Com o medico: " +
consultas.get(i).getMedico().getNum_medico_unico());

                System.out.println("Especialidade: "  +
consultas.get(i).getMedico().getEspecialidade() +"\n");


            } else if (i == consultas.size() - 1)

                System.out.println("Nao tem consultas marcadas \n");

            }

        } else if (marcou_Consulta == true && FezExame == false) {

            //VERIFICA SE DETERMINADO UTENTE TEM CONSULTA MARCADA MAS NAO PRECISA
DE EXAME

            for (int i = 0; i < consultas.size(); i++) {


                if (consultas.get(i).getUtente().getNumero_de_utente() ==
utente.get(opcao_paciente).getNumero_de_utente()) {


                System.out.println("\n Tem consulta Diagnostico dia: " +
consultas.get(i).getData_consulta().get(Calendar.DAY_OF_MONTH) + "/" +
consultas.get(i).getData_consulta().get(Calendar.MONTH) + "/" +
consultas.get(i).getData_consulta().get(Calendar.YEAR));
```

```
                System.out.println("Com o medico: " +
consultas.get(i).getMedico().getNum_medico_unico());
                System.out.println("Especialidade: "  +
consultas.get(i).getMedico().getEspecialidade() + "\n");
            } else if (i == consultas.size() - 1)
                System.out.println("Nao tem consultas marcadas \n");
            }
        } else
            System.out.println("Nao tem consultas marcadas \n");
    }
    catch (ArrayIndexOutOfBoundsException ioe ) {
        System.out.println("Nao tem consultas marcadas \n");
    }
  }


  public void Consultar_utentes(ArrayList<Utentes> pacientes) {


    System.out.println( "\nDeseja ordenear a lista de utentes pelo valor pago ?\n1 - Sim\n2 -
Não" );


    switch(scan.nextInt()) {
        case 1:
            System.out.println("1 - Ascendente\n 2 - Descendente");
            if (scan.nextInt() == 1) {
                //LISTA DE ORDEM ASCENDENTE
                Collections.sort(pacientes);
                System.out.println(pacientes);


            } else if (scan.nextInt() == 2) {
                //LISTA DE ORDEM DESCENDENTE
                Collections.sort(pacientes, Collections.reverseOrder());
```

```
            System.out.println(pacientes);
        } else
            System.out.println("Opcao errada");

        break;
      case 2:
        if (pacientes.size() != 0) {
          for (int n = 0; n < pacientes.size(); n++) {
            System.out.println(n + 1);
            System.out.println(pacientes.get(n).toString());
          }
        } else
          System.out.println("Nao é possivel");
        break;

      default:
        System.out.println("Opcao errada !");
    }
  }
  public void Consultar_medicos(ArrayList<Medicos> medicos) {

    System.out.println( "\nDeseja ordenear as lista de medicos pela avaliacao media ?\n1 -
Sim\n2 - Não" );

    switch(scan.nextInt()) {
      case 1:
        System.out.println("1 - Ascendente\n 2 - Descendente");
        if (scan.nextInt() == 1) {
          //LISTA DE ORDEM ASCENDENTE
          Collections.sort(medicos);
          if (medicos.size() != 0) {
```

```
            for (int n = 0; n < medicos.size(); n++) {
                System.out.println(n + 1);
                System.out.println(medicos.get(n).toString());
            }
        } else
            System.out.println("Nao é possivel");

    } else if (scan.nextInt() == 2) {
        //LISTA DE ORDEM INVERSA, DESCENDENTE
        Collections.sort(medicos, Collections.reverseOrder());
        if (medicos.size() != 0) {
            for (int n = 0; n < medicos.size(); n++) {
                System.out.println(n + 1);
                System.out.println(medicos.get(n).toString());
            }
        } else
        System.out.println("Nao é possivel");
    } else
        System.out.println("Opcao errada");

    break;
case 2:
    if (medicos.size() != 0) {
        for (int n = 0; n < medicos.size(); n++) {
            System.out.println(n + 1);
            System.out.println(medicos.get(n).toString());
        }
    } else
        System.out.println("Nao é possivel");
    break;
```

```
            default:
                System.out.println("Opcao errada !");
        }
    }


    public void listar_consultas_dia(ArrayList<Medicos> medicos, ArrayList<Utentes> utentes,
ArrayList<Consultas> consultas) {

        int dia_escolhido_erro = 0;

        System.out.println( "Qual o medico que pretende escolher ??" );
        for (int i = 0; i < medicos.size(); i++)
            System.out.println( i + " - " + medicos.get( i ).getNum_medico_unico() + " | " +
medicos.get( i ).getEspecialidade() );

        int opcao = scan.nextInt();

        System.out.println( "Que dia pretende saber ??" );
        //LISTA OS PROXIMOS 7 DIAS
        for (int i = 0; i < 7; i++)
            System.out.println( (data.get( Calendar.DAY_OF_MONTH ) + i) + "/" + data.get(
Calendar.MONTH ) + "/" + data.get( Calendar.YEAR ) );

        int dia_escolhido = scan.nextInt();

        for (int i = 0; i < consultas.size(); i++) {
            //VERIFICA E LISTA A CONSULTA DE DETERMINADO DIA E DE DETERMINADO MEDICO
CORRESPONDEM AO SELECIONADO
            if (consultas.get( i ).getMedico().getNum_medico_unico() == medicos.get( opcao
).getNum_medico_unico()
                    && (consultas.get( i ).getData_consulta().get( Calendar.DAY_OF_MONTH ) ==
dia_escolhido
```

```
            || consultas.get( i ).getData_consulta_resultados().get( Calendar.DAY_OF_MONTH )
== dia_escolhido)) {



            dia_escolhido_erro = dia_escolhido_erro + 1;


            System.out.println( consultas.get( i ).getUtente().getNome() );

            System.out.println( consultas.get( i ).getMedico().getEspecialidade() );

        }

    }

    if (dia_escolhido_erro == 0)

        System.out.println( "Não exitem marcações para o medico " + medicos.get( opcao
).getNum_medico_unico() );

  }


  public void Registar_medicos(ArrayList<Medicos> medicos) {

    System.out.printf( "Existem " + medicos.size() + " medicos registados" + '\n' );


    Random rand = new Random();

    int num_medico_unico = rand.nextInt( 99999999 );



    boolean feito;


    System.out.println( "Insira a especialidade: " );

    String especialidade = scan.next();

    Medicos m;

    m = new Medicos( num_medico_unico, especialidade );

    //VERIFICA SE EXISTE UM MEDICO COM O MESMO NUMERO UNICO

    for (int i = 0; i < medicos.size(); i++) {

      if (m.equals( medicos.get( i ).getNum_medico_unico() )) {

        System.out.println( "Ja existe um medico com este numero" );
```

```
            feito = false;
        } else
            feito = true;

    }
    if (feito = true)
        medicos.add( m );
    else {
        num_medico_unico = rand.nextInt( 99999999 );
        medicos.add( new Medicos( num_medico_unico, especialidade ) );
    }

}

public void Registar_utentes(ArrayList<Utentes> pacientes) {
    //REGITA UM NOVO UTENTE, COM NUMERO DE UTENTE ALEATORIO E PEDE O RESTANTE AO UTILIZADOR
    System.out.printf( "Existem " + pacientes.size() + " utentes registados" + '\n' );

    Random rande = new Random();
    int numero_de_utente = rande.nextInt( 99999999 );

    System.out.println( "Insira o nome: " );
    String nome = scan.next();

    System.out.println( "Insira o idade: " );
    int idade = Integer.parseInt( scan.next() );

    System.out.println( "Seguro medico?" );
    boolean seg_med;
    seg_med = scan.nextBoolean();
```

```
    //INSTANCIADO NOVO UTENTE
    Utentes p;
    p = new Utentes( nome, idade, numero_de_utente, seg_med );

    boolean feito2;
    //VERIFICA SE EXISTE UM UTENTE COM O MESMO NUMERO UNICO
    for (int i = 0; i < pacientes.size(); i++) {
        if (p.equals( pacientes.get( i ).getNumero_de_utente() )) {
            System.out.println( "Ja existe um paciente com este numero" );
            feito2 = false;
        } else
            feito2 = true;

    }
    if (feito2 = true)
        pacientes.add( p );
    else {
        numero_de_utente = rande.nextInt( 99999999 );
        pacientes.add( new Utentes( nome, idade, numero_de_utente, seg_med ) );
    }
}

public void Ver_consultas(ArrayList<Consultas> consultas) { //LISTA AS CONSULTAS
    if (consultas.size() != 0) {
        for (int n = 0; n < consultas.size(); n++) {
            System.out.println( n + 1 + " ----------" );
            System.out.println( consultas.get( n ).toString() );
        }
    } else
        System.out.println( "Nao é possivel" );
```

```
    }
}
package com.projeto_2;

import java.util.ArrayList;
import java.util.Calendar;
import java.util.Scanner;

public class Medicos implements Comparable<Medicos> {


    private ArrayList<Integer> Array_avaliacao;
    private int num_medico_unico;
    private int divisao;
    private int avaliacao_media;
    private int soma = 0;

    private Scanner scan = new Scanner( System.in );
    private String especialidade;

    private ArrayList<Calendar> consultas_dia = new ArrayList<>();

    public Medicos(int num_medico_unico, String especialidade) {
        this.especialidade = especialidade;
        this.num_medico_unico = num_medico_unico;
        avaliacao_media = 0;
        Array_avaliacao = new ArrayList<Integer>();
    }

    public int getAvaliacao_media( ) {
        return avaliacao_media;
```

```
}

public int getNum_medico_unico( ) {
   return num_medico_unico;
}

public String getEspecialidade( ) {
   return especialidade;
}

public void setAvaliacao(Integer avaliacao) {
   Array_avaliacao.add( avaliacao );
}

public ArrayList<Integer> getArray_avaliacao( ) {
   return Array_avaliacao;
}


public int media( ) {
   divisao = (Array_avaliacao.size());
   if (divisao == 0)
      return avaliacao_media = 0;

   else {
      soma = 0;
      for (Integer elemento : Array_avaliacao)
         soma = soma + elemento;
      avaliacao_media = (soma / divisao);

      return avaliacao_media;
```

```
        }
    }
    public int compareTo(Medicos medicos) {
        if (this.avaliacao_media > medicos.getAvaliacao_media( )) {
            return -1;
        }
        if (this.avaliacao_media < medicos.getAvaliacao_media( )) {
            return 1;
        }
        return 0;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj) {
            return true;
        } else if (obj == null || getClass() != obj.getClass()) {
            return false;
        }
        Medicos med = (Medicos) obj;
        return (num_medico_unico == med.getNum_medico_unico());
    }

    @Override
    public String toString( ) {
        return "\n - Numero unico -" + num_medico_unico +
                "\n - Especialidade - " + especialidade +
                "\n - Avaliação - " + media() + '\n' +
                " - Historico de avaliacoes " + getArray_avaliacao();
    }
}
```

```
package com.projeto_2;

import java.io.IOException;

public class Main {

   public static void main(String[] args) throws IOException {
      Menu m = new Menu();
   }
}
package com.projeto_2;

import com.projeto_2.Medicos;
import com.projeto_2.Utentes;

import java.util.Calendar;
import java.util.Random;

public class Consultas {

   private static double Valor_a_pagar; // Static para encontrar a variavel local
   private Medicos medico;
   private Utentes utente;
   private int idade; // variavel para a idade

   private double valorExame = 80; // constante para o valor do exame
   private double valorConsulta = 100; // constante para o valor da consulta
   private boolean seg_med; // variavel boleana para verificar se tem seguro medico
   private boolean faz_exame; // variavel para verificar se fez exame

   private Calendar data_consulta = Calendar.getInstance();
```

```
    private Calendar data_exame = Calendar.getInstance();
    private Calendar data_consulta_resultados = Calendar.getInstance();
    private Calendar data_pagamento = Calendar.getInstance();

    /*
    Construtor para a classe consultas
     */
    public Consultas(Medicos medico, Utentes utente, int dia_pretendidio) {

        Valor_a_pagar = 0;

        data_consulta.set( Calendar.DAY_OF_MONTH, dia_pretendidio );

        this.medico = medico;
        this.utente = utente;
        this.idade = utente.getIdade();
        this.seg_med = utente.getSeguro();

        Exames();
        pagamento();
    }

/*
getter para retornar o valor a pagar
 */

    public static double getValor_a_pagar( ) {
        return Valor_a_pagar;
    }

    /*
```

```
   Metodo que faz que com o medico mande fazer exame ou nao
    */
   private void Exames( ) {
      int rand = new Random().nextInt( 1 );
      faz_exame = (rand == 1 ? true : false);

      if (faz_exame == true) { // LOGICA É AO AVANCAR O DIA ELE VERIFICAR SO A DATA DE
PAGAMENTO E EFETUAR O PAGAMENTO FACIL !
         data_exame.set( Calendar.DAY_OF_MONTH, (data_consulta.get(
Calendar.DAY_OF_MONTH )) );
         data_consulta_resultados.set( Calendar.DAY_OF_MONTH, (data_consulta.get(
Calendar.DAY_OF_MONTH ) + 7 ) );
         data_pagamento.set( Calendar.DAY_OF_MONTH, (data_consulta.get(
Calendar.DAY_OF_MONTH ) + 7 ) );
      } else data_pagamento.set( Calendar.DAY_OF_MONTH, data_consulta.get(
Calendar.DAY_OF_MONTH ) );
   }

   /*
   Metodo pagamento que faz os pagamentos
    */
   private void pagamento( ) {

      Valor_a_pagar = 0; // inicializa a 0
      if (faz_exame == true) { // verifica se fez um exame
         if (seg_med == true) { // verifica se tem seguro medico
            Valor_a_pagar = (valorConsulta * 0.4 + valorExame * 0.2); // aplica o valor a pagar
         } else if (idade >= 65) Valor_a_pagar = (valorConsulta * 0.1 + valorExame * 0.1);
//verifica se tem mais que 65 anos e aplica o valor a pagar
         else Valor_a_pagar = valorConsulta + valorExame; // no caso que nao tenha seguro
medico ou mais que 65 anos paga na totalidade
      } else {
         if (seg_med == true) { // caso nao faz exame nao precisa que o pagar
            Valor_a_pagar = (valorConsulta * 0.4);
```

```
            } else if (idade >= 65) Valor_a_pagar = (valorConsulta * 0.1);
            else Valor_a_pagar = valorConsulta;
        }
    }
/*
Getter para retornar variaveis privadas (Encapsulamento)
 */
    public Medicos getMedico( ) {
        return medico;
    }

    public Utentes getUtente( ) {
        return utente;
    }

    public Calendar getData_consulta( ) {
        return data_consulta;
    }

    public Calendar getData_exame( ) {
        return data_exame;
    }

    public Calendar getData_consulta_resultados( ) {
        return data_consulta_resultados;
    }

    public Calendar getData_pagamento( ) {
        return data_pagamento;
    }
```

```
    public boolean isFaz_exame( ) {
        return faz_exame;
    }
/*
Metodo toString que mostra ao utilizador da aplicaçao se ira fazer exame ou nao, as datas das consultas e o valor a pagar
 */
    @Override
    public String toString( ) {
        String text = "\n\n Medico: " + medico.getNum_medico_unico() +
                "\n Paciente: " + utente.getNome();

        if (faz_exame == false) {
            text = text + "\n\n Não vai fazer exame !" + "\n ";
            text = text + "\n Data da consulta: " + data_consulta.get( Calendar.DAY_OF_MONTH ) + "/" + data_consulta.get( Calendar.MONTH ) + "/" + data_consulta.get( Calendar.YEAR );
            text = text + "\n Data do pagamento: " + getData_pagamento().get( Calendar.DAY_OF_MONTH ) + "/" + getData_pagamento().get( Calendar.MONTH ) + "/" + getData_pagamento().get( Calendar.YEAR );
        } else {
            text = text + "\n\n Vai fazer exame !" + "\n ";
            text = text + "\n Data da consulta: " + data_consulta.get( Calendar.DAY_OF_MONTH ) + "/" + data_consulta.get( Calendar.MONTH ) + "/" + data_consulta.get( Calendar.YEAR );
            text = text + "\n Data do exame: " + data_exame.get( Calendar.DAY_OF_MONTH ) + "/" + data_exame.get( Calendar.MONTH ) + "/" + data_exame.get( Calendar.YEAR );
            text = text + "\n Data da consulta de resultados: " + getData_consulta_resultados().get( Calendar.DAY_OF_MONTH ) + "/" + data_consulta_resultados.get( Calendar.MONTH ) + "/" + getData_consulta_resultados().get( Calendar.YEAR );
            text = text + "\n Data do pagamento: " + getData_pagamento().get( Calendar.DAY_OF_MONTH ) + "/" + getData_pagamento().get( Calendar.MONTH ) + "/" + getData_pagamento().get( Calendar.YEAR );
        }
        text = text + "\n\n Valor a pagar: " + Valor_a_pagar + " € \n\n";

        return text;
```

```
    }
}
package com.projeto_2;

import java.util.Scanner;

public class Utentes implements Comparable<Utentes> {

    private String nome; // variavel para o nome
    private int idade;// variavel para a idade
    private int numero_de_utente;// variavel para o numero de utente

    private boolean seguro_medico;// variavel seguro medico verdadeiro ou false
    private double valor_pago;// variavel pra o valor pago
    private Scanner scan = new Scanner( System.in );

    /*
    Construtor da classe Utentes
     */
    public Utentes(String nome, int idade, int numero_de_utente, boolean seguro_medico) {

        this.nome = nome;
        this.idade = idade;
        this.numero_de_utente = numero_de_utente;
        this.seguro_medico = seguro_medico;
        valor_pago = 0;

    }

/*
Metodos getter para retornar as variaveis privadas (Encapsulamento)
```

```
 */
    public String getNome( ) {
        return nome;
    }

    public int getNumero_de_utente( ) {
        return numero_de_utente;
    }

    public void setNumero_de_utente(int numero_de_utente) {
        this.numero_de_utente = numero_de_utente;
    }

    public int getIdade( ) {
        return idade;
    }

    public boolean isSeguro_medico( ) {
        return seguro_medico;
    }


    public boolean getSeguro( ) {
        return seguro_medico;
    }
/*
Metodo equals para verificar se existe algum numero de utente igual
 */
    @Override
    public boolean equals(Object obj) {
        if (this == obj) {
```

```
            return true;
        } else if (obj == null || getClass() != obj.getClass()) {
            return false;
        }
        Utentes utentes = (Utentes) obj;
        return (numero_de_utente == utentes.getNumero_de_utente());
    }

    public double getValor_pago( ) {
        return valor_pago;
    }
/*
Metodo setValor_pago para encrementar o valor pago pelo utente
 */
    public void setValor_pago(double valor_pago) {
        this.valor_pago = this.valor_pago + valor_pago;
    }
/*
Metodo compareTo que atravez de  uma coleção ordena a lista de utentes
 */

    @Override
    public int compareTo(Utentes utentes) {
        if (this.valor_pago > utentes.getValor_pago()) {
            return -1;
        }
        if (this.valor_pago < utentes.getValor_pago()) {
            return 1;
        }
        return 0;
    }
```

```
/*
Metodo toString que mostra ao Utilizador da aplicação em texto as variaveis de cada utente
 */
    @Override
    public String toString( ) {
        return "\n - Nome - " + getNome() +
                "\n - Idade - " + getIdade() +
                "\n - Numero unico de utente - " + getNumero_de_utente() +
                "\n - Seguro medico - " + getSeguro() +
                "\n - Valor gasto - " + valor_pago;
    }
}
```