

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA DE CIÊNCIAS EXATAS E DA COMPUTAÇÃO.
ANÁLISE E DESENVOLVIMENTO DE SISTEMAS.



AGENDA EM LISTA ENCADEADA E ORDENADA

GOIÂNIA, 12 DE MAIO DE 2020
Bruno Camargo Manso

Agenda em Lista Encadeada e Ordenada

Trabalho para composição das notas de N2

Orientador: José Olímpio Ferreira

GOIÂNIA, GO
2020

Resumo

O presente trabalho tem o objetivo de desenvolver uma agenda cujos dados são guardados em uma lista simplesmente encadeada. A lista deve ser ordenada seguindo a lógica de ordenação de uma lista encadeada. O programa deverá, além de mostrar os dados de forma ordenada (crescente e decrescente) para o usuário, salvar em um arquivo de texto, ou backup, para que este seja devidamente reaproveitado na próxima vez que o programa for executado. O trabalho a seguir contém códigos em Java tanto do programa em modo texto (terminal) quanto em interface gráfica (Jframe).

Palavras-chave: Agenda. Lista simplesmente encadeada. Ordenada. Backup. Códigos em Java. Programa. Modo texto. Interface gráfica.

Sumário

1. Classe Lista Encadeada	5
2. Classe Pessoa	7
3. Classe Gerar Arquivo	8
4. Interface Gráfica	
4.1. Classe FrmContatosLista	9
4.2. Classe FrmContatos	13
5. Menu (modo terminal/texto)	16
6. Exemplo do arquivo gerado	19

1. Classe Lista Encadeada

Essa classe contém o algoritmo necessário para guardar Objetos do tipo Pessoa de maneira ordenada. A ordenação dependerá do método 'Inserir' que sempre irá comparar valores antes da inserção, garantindo que tal pessoa seja inserida de forma alfabeticamente correta.

Abaixo a implementação dos métodos:

```
24 public class Lista {
25     public class No {
26         private Pessoa dados;
27         private No proximo;
28
29         public No(Pessoa aux) {
30             dados = aux;
31             proximo = null;
32         }
33     }
34     private int tamanho;
35     private No inicio;
36
37     public Lista() {
38         tamanho = 0;
39         inicio = null;
40     }
41
42     public boolean vazia() {
43         return (inicio == null);
44     }
45
46     public int getTam() {
47         return tamanho;
48     }
49
50     public Pessoa busca(Pessoa aux) {
51         if (vazia())
52             return null;
53         No i = null;
54         for (i = inicio; i != null && !aux.getNome().equals(i.dados.getNome()); i = i.proximo);
55         if (i == null) {
56             return null;
57         }
58         Pessoa novo = new Pessoa(i.dados.getNome(), i.dados.getTelefone(), i.dados.getEndereco());
59         return novo;
60     }
```

```
61
62     public boolean adicionaOrdenado(Pessoa aux) {
63         Pessoa buscado = busca(aux);
64         if (buscado != null)
65             return false;
66         No novo = new No(aux);
67         if (vazia()) {
68             inicio = novo;
69             tamanho++;
70             return true;
71         }
72         No i = inicio;
73         No ant = inicio;
74         for (; i != null && aux.getNome().compareTo(i.dados.getNome()) > 0; ant = i, i = i.proximo)
75             ;
76         if (i == ant) {
77             novo.proximo = inicio;
78             inicio = novo;
79         } else {
80             ant.proximo = novo;
81             novo.proximo = i;
82         }
83         tamanho++;
84         return true;
85     }
```

```

87● public Pessoa retira(Pessoa aux) {
88     if (vazia())
89         return null;
90     No i = inicio;
91     No ant = inicio;
92     for (; i != null && !aux.getNome().equals(i.dados.getNome()); ant = i, i = i.proximo)
93         ;
94     if (i == null) {
95         return null;
96     }
97     Pessoa novo = i.dados;
98     if (i == ant) {
99         inicio = inicio.proximo;
100     } else {
101         ant.proximo = i.proximo;
102     }
103     tamanho--;
104     return novo;
105 }
106

```

```

107● public String imprima() {
108     String aux = new String("");
109
110     for (No i = inicio; i != null; i = i.proximo) {
111         aux = aux + i.dados.toString();
112     }
113
114     return aux.toString();
115 }
116
117● public String imprimaInv() {
118     String[] aux = imprima().split("\n");
119     String reverso = "";
120
121     for (int i = 0; i < aux.length; i++) {
122         if (i == aux.length - 1)
123             reverso = aux[i] + reverso;
124         else
125             reverso = "\n" + aux[i] + reverso;
126     }
127     return reverso;
128 }
129 }

```

2. Classe Pessoa

Tal classe define os atributos que serão armazenados dentro de um Objeto do tipo Pessoa. Os atributos são do tipo String: nome, telefone e endereço.

```
2
3 public class Pessoa {
4
5     private String nome;
6     private String telefone;
7     private String endereco;
8
9     public Pessoa() {
10    }
11    public Pessoa(String nome, String telefone, String endereco) {
12        this.nome = nome;
13        this.telefone = telefone;
14        this.endereco = endereco;
15    }
16    public String getNome() {
17        return nome;
18    }
19    public void setNome(String nome) {
20        this.nome = nome;
21    }
22    public String getTelefone() {
23        return telefone;
24    }
25    public void setTelefone(String telefone) {
26        this.telefone = telefone;
27    }
28    public String getEndereco() {
29        return endereco;
30    }
31    public void setEndereco(String endereco) {
32        this.endereco = endereco;
33    }
34    @Override
35    public String toString() {
36        return nome + "," + telefone + "," + endereco + "\n";
37    }
38 }
```

3. Classe Gerar Arquivo

Responsável por fazer ações de Ler e Salvar os dados computados no programa em um arquivo de texto. Isso irá garantir que os dados não se percam a após finalização do programa e que sejam lidos assim que o programa reinicia.

```
29 public class GerArquivo {
30
31     static final String NOME_ARQ = "contatos.txt";
32
33
34     public static Lista lerArquivo(Lista list) throws FileNotFoundException, IOException {
35         FileReader fr = new FileReader(NOME_ARQ);
36         BufferedReader br = new BufferedReader(fr);
37         FrmContatosLista lista = new FrmContatosLista();
38         ArrayList<Pessoa> arrayPessoas = new ArrayList<Pessoa>();
39
40         while (br.ready()) {
41             String linha = br.readLine(); // Lê linha
42             String dados[] = linha.split(","); // Divide por ','
43             Pessoa p = new Pessoa(dados[0], dados[1], (dados[2])); // Insere novo Objeto Pessoa com os dados vindos do arquivo
44             list.adicionaOrdenado(p); // Adiciona Objeto Pessoa na Lista ( encadeada )
45             arrayPessoas.add(p); // Adiciona Objeto Pessoa em um ArrayList para ppular a JTable
46         }
47
48         // System.out.println(list.imprima()); // Debug
49         lista.setLista(arrayPessoas); // Popula JTable com ArrayList
50         br.close();
51         fr.close();
52         return list;
53     }
54 }
```

```
56     public static void gravarArquivo(Lista list, boolean manterArq) throws IOException {
57         FileWriter fw = new FileWriter(NOME_ARQ, manterArq); // Pega arquivo por nome e decide se vai sobrepor o arquivo ou adicionar
58         BufferedWriter bw = new BufferedWriter(fw);
59         String aux = list.imprima(); // Pega todos os dados em Lista ligada e transforma em uma String
60         bw.write(aux); // Escreve a String de dados no arquivo
61         bw.close();
62         fw.close();
63     }
64
65     public static void gravarArquivo(Lista list) throws IOException {
66         FileWriter fw = new FileWriter(NOME_ARQ, false);
67         BufferedWriter bw = new BufferedWriter(fw);
68         String str = list.imprima(); // Pega todos os dados em Lista ligada e transforma em uma String
69         bw.write(str); // Escreve a String de dados no arquivo
70         bw.close();
71         fw.close();
72     }
73
74     public static String getNomeArq() {
75         return NOME_ARQ;
76     }
77 }
```


4. Interface Gráfica

Implementada em JFrame podendo também exibir mensagens em JOptionPane, a interface gráfica melhora a experiência do usuário distanciando-o de processos do tipo terminal, facilitando assim o entendimento e a visualização do produto do programa, tornando este bem mais amigável, intuitivo e mais dinâmico.

4.1. Classe FrmContatosLista

Implementa a janela principal do programa. Ao iniciar, deverá carregar em uma tabela (JTable) os dados do arquivo. A tabela, devidamente preenchida, permite a ordenação dos elementos, por atributos e de forma crescente/decrescente. Os botões Adicionar, Editar, Excluir e Sair estão contidos nessa tabela, garantindo o devido CRUD das informações. Então para efetivar uma deleção (ou edição), deve-se clicar sobre a linha desejada e utilizar os botões. Perceba que o aqui existe o uso do ArrayList somente para o preenchimento da tabela JFrame, nada mais. O ArrayList servirá apenas para popular os dados na tabela (JTable). O código abaixo está devidamente comentado ressaltando tanto esse último detalhe, quanto outros métodos implementados no código.

```
43 public class FrmContatosLista extends JFrame {
44
45     private static final long serialVersionUID = 1L;
46     private JPanel contentPane;
47     private JScrollPane scrollPane;
48     public void setScrollPane(JScrollPane scrollPane) {
49         this.scrollPane = scrollPane;
50     }
51
52     private JTable table;
53     private ArrayList<Pessoa> lista = new ArrayList<Pessoa>();
54     private Lista list;
55
56     public static void main(String[] args) {
57         try {
58
59             for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
60                 if ("Nimbus".equals(info.getName())) {
61                     javax.swing.UIManager.setLookAndFeel(info.getClassName());
62                     break;
63                 }
64             }
65         } catch (ClassNotFoundException ex) {
66             java.util.logging.Logger.getLogger(FrmContatosLista.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
67         } catch (InstantiationException ex) {
68             java.util.logging.Logger.getLogger(FrmContatosLista.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
69         } catch (IllegalAccessException ex) {
70             java.util.logging.Logger.getLogger(FrmContatosLista.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
71         } catch (javax.swing.UnsupportedLookAndFeelException ex) {
72             java.util.logging.Logger.getLogger(FrmContatosLista.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
73         }
74     }
75 }
```

iniciando variáveis e configurando o tema do visual da interface gráfica.

```

75     EventQueue.invokeLater(new Runnable() {
76         public FrmContatosLista frame;
77
78         public void run() {
79             try {
80                 frame = new FrmContatosLista();
81                 frame.setVisible(true);
82             } catch (Exception e) {
83                 e.printStackTrace();
84             }
85         }
86     });
87 }
88
89 /* Configurando Janela */
90
91 public FrmContatosLista() {
92     setFont(new Font("Ubuntu Mono", Font.BOLD, 12));
93     setTitle("Lista de Contatos");
94     setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
95     addWindowListener(new WindowAdapter() {
96         @Override
97         public void windowActivated(WindowEvent arg0) {
98             preencherDataTable(true);
99         }
100     });
101     setBounds(100, 100, 700, 300);
102     contentPane = new JPanel();
103     contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
104     setContentPane(contentPane);
105     contentPane.setLayout(new BorderLayout(0, 0));
106     setLocationRelativeTo(null);
107     // Janela no meio da tela

```

iniciando JFrame, gerando janela, configurando módulos da janela.

```

109 JPanel painel = new JPanel();
110 contentPane.add(painel, BorderLayout.CENTER);
111 painel.setLayout(new BorderLayout(0, 0));
112
113 scrollPane = new JScrollPane(table = new JTable());
114 scrollPane.setFont(new Font("Ubuntu Light", Font.PLAIN, 13));
115 scrollPane.setViewportBorder(null);
116 table = new JTable();
117 table.setModel(new DefaultTableModel(new Object[][] {}, new String[] { "Nome", "Telefone", "Endereço" }) {
118     private static final long serialVersionUID = 1L;
119     boolean[] columnEditables = new boolean[] { false, false, false };
120
121     public boolean isCellEditable(int row, int column) {
122         return columnEditables[column];
123     }
124 });
125 table.getColumnModel().getColumn(0).setPreferredWidth(300);
126 table.getColumnModel().getColumn(1).setPreferredWidth(300);
127 table.getColumnModel().getColumn(2).setPreferredWidth(600);
128 scrollPane.setViewportView(table);
129 table.setAutoCreateRowSorter(true);
130
131 painel.add(scrollPane, BorderLayout.CENTER);
132
133 JPanel panel = new JPanel();
134 contentPane.add(panel, BorderLayout.SOUTH);

```

ainda configurando módulos da janela principal.

```

142 JButton btnAdicionar = new JButton("Adicionar");
143 btnAdicionar.setFont(new Font("Ubuntu Light", Font.PLAIN, 14));
144 btnAdicionar.addActionListener(new ActionListener() {
145     public void actionPerformed(ActionEvent arg0) {
146
147         FrmContatos frmCadastro = new FrmContatos(lista);
148         frmCadastro.getBtnGravar().setText("Gravar");
149         frmCadastro.setVisible(true);
150         scrollPane.setVisible(false);
151         dispose();
152
153     }
154 });
155
156 panel.setLayout(new FlowLayout(FlowLayout.CENTER, 5, 5));
157 panel.add(btnAdicionar);
158

```

configurando funções para o botão 'Adicionar'

```

162 JButton btnAlterar = new JButton("Editar");
163 btnAlterar.setMinimumSize(new Dimension(100, 25));
164 btnAlterar.setMaximumSize(new Dimension(100, 25));
165 btnAlterar.setPreferredSize(new Dimension(100, 25));
166 btnAlterar.setFont(new Font("Ubuntu Light", Font.PLAIN, 14));
167 btnAlterar.addActionListener(new ActionListener() {
168     public void actionPerformed(ActionEvent e) {
169         scrollPane.setVisible(false); // Desliga o Jtable
170         dispose(); // Desliga o painel principal
171         FrmContatos frmCadastro = new FrmContatos(lista); // Cria Objeto do Tipo FrmContatos estanciando com um ArrayList já criado
172         int indice = table.getSelectedRow(); // Ao clicar em um nome, pega a posição e guarda na variável indice
173
174         if (indice >= 0) { // Se o indice for maior ou igual a zero, faça:
175
176             frmCadastro.getBtnGravar().setText("Alterar");
177
178             Pessoa p = lista.get(indice); // Cria Objeto do tipo Pessoa instanciado pelo indice escolhido
179             // Preenche os campos com o Objeto Pessoa:
180
181             frmCadastro.getTxtNome().setText(p.getNome());
182             frmCadastro.getTxtMatricula().setText(p.getEndereco());
183             frmCadastro.getTxtFone().setText(p.getTelefone());
184             frmCadastro.setIndice(indice);
185
186             frmCadastro.setVisible(true); // Liga o Painel de Cadastro
187
188         } else {
189             JOptionPane.showMessageDialog(null, "Selecione uma pessoa na tabela");
190             FrmContatosLista janelaCadastro = new FrmContatosLista();
191             janelaCadastro.setVisible(true);
192         }
193     }
194 });
195 panel.add(btnAlterar);

```

configurando funções para o botão ‘Alterar’

```

200 JButton btnExcluir = new JButton("Excluir");
201 btnExcluir.setPreferredSize(new Dimension(100, 25));
202 btnExcluir.setMinimumSize(new Dimension(100, 25));
203 btnExcluir.setMaximumSize(new Dimension(100, 25));
204 btnExcluir.setFont(new Font("Ubuntu Light", Font.PLAIN, 14));
205 btnExcluir.addActionListener(new ActionListener() {
206     public void actionPerformed(ActionEvent arg0) {
207
208         dispose(); // Desliga o Painel Principal
209         int indice = table.getSelectedRow(); // Pega o Indice da Linha escolhida
210         if (indice >= 0) { // Se estiver escolhido, faça:
211             Pessoa aux = new Pessoa(); // Adicionar Objeto auxiliar Pessoa
212             // Adicionar Pessoa do Indice no Objeto auxiliar:
213
214             aux.setNome(lista.get(indice).getNome());
215             aux.setTelefone(lista.get(indice).getTelefone());
216             aux.setEndereco(lista.get(indice).getEndereco());
217
218             list.retira(aux); // Remove Pessoa da Lista ( encadeada )
219             lista.remove(indice); // Remove Pessoa do ArrayList
220
221             try { // tente:
222
223                 GerArquivo.gravarArquivo(list); // Gravar Lista Encadeada no arquivo de Texto
224             } catch (IOException e) { // Tratar Excessões
225                 JOptionPane.showMessageDialog(null, // Avisar ao usuário caso não consiga
226                     "Falha ao gravar os dados da agenda!");
227             }
228             FrmContatosLista janela = new FrmContatosLista(); // Cria um Objeto desta Classe para permitir o acesso a ela
229             janela.setVisible(true); // Religa o Painel Principal
230         }
231     }
232 });
233 panel.add(btnExcluir);

```

configurando funções para botão ‘Excluir’

```

238 JButton btnSair = new JButton("Sair");
239 btnSair.setPreferredSize(new Dimension(100, 25));
240 btnSair.setMinimumSize(new Dimension(100, 25));
241 btnSair.setMaximumSize(new Dimension(100, 25));
242 btnSair.setFont(new Font("Ubuntu Light", Font.PLAIN, 14));
243 btnSair.addActionListener(new ActionListener() {
244     public void actionPerformed(ActionEvent e) {
245         File file = new File(GerArquivo.getNomeArq());
246
247         try {
248             java.awt.Desktop.getDesktop().open(file);           // Abre o arquivo de backup no programa de texto default
249             System.exit(2);                                       // Desliga o programa
250
251         } catch (IOException e2) {
252             JOptionPane.showMessageDialog(null, "Falha ao abrir arquivo no sistema, acesse-o via diretório raiz");
253         }
254     }
255 });
256 panel.add(btnSair);
257 }
258

```

configurando funções para o botão ‘Sair’

```

263 protected void preencherDataTable(boolean lerArquivo) {
264
265     DefaultTableModel modelo = (DefaultTableModel) table.getModel();
266     modelo.setRowCount(0);
267
268     if (lerArquivo == true) {
269         try {
270             list = new Lista();
271             GerArquivo.lerArquivo(list);           // passagem de parametros da Classe Lista para Classe GerArquivo
272         } catch (IOException e) {
273             JOptionPane.showMessageDialog(null, "Falha ao ler os dados da agenda!");
274         }
275     }
276
277     /* inicio do preenchimento da JFrameTable */           /* P: Por quê usar um ArrayList? */
278                                                         /* R: Para o devido preenchimento da tabela JFrame */
279
280     String[] aux = list.imprima().split("\n");           // converte lista encadeada ordenada para array de String e separa por pessoas
281     for (int i = 0; i < aux.length; i++) {               // roda todos os elementos do tamanho da lista encadeada
282         String[] temp = aux[i].split(",");               // divide cada atributo de cada pessoa por virgulas
283         Pessoa p = new Pessoa(temp[0], temp[1], temp[2]); // insere os atributos em um objeto tipo Pessoa
284         lista.add(p);                                     // adiciona as pessoas em um arrayList
285     }
286     for (Pessoa p : lista) {                             // para cada elemento da lista, adiciona uma pessoa
287                                                         // e coloca atributos em uma linha:
288
289         modelo.addRow(new Object[] { p.getNome(), p.getTelefone(), p.getEndereco() });
290     }
291 }
292

```

configurando funções de carregamento e preenchimento da tabela (JTable)

```

295 public Lista getList() {
296     return list;
297 }
298
299 public void setList(Lista list) {
300     this.list = list;
301 }
302
303 public ArrayList<Pessoa> getLista() {
304     return lista;
305 }
306
307 public void setLista(ArrayList<Pessoa> lista) {
308     this.lista = lista;
309 }
310 }

```

getters e setters da classe

A seguir, o resultado do código:



Nome	Telefone	Endereço
Bruno	+55(62)99553-0338	Goiânia
Bzorak	+fd:04(Rzx)32-53	Mars
Campbell	+1(212)78383-3473	Manhattan
Dimitri	+380(4593)573-4573	Chernobyl
Donald Draper	+1(212)43522-3494	Manhattan
Kalashnikov	+7(941)8499-7593	Kurya
Katia	+375(66)345345432	Bellarus
Lilian	+55(62)8374-34787	Goiânia
Linuchka	+375(17)3994-8484	Minsk
Lothar	+fd:04(Rzh)20-40	Mars
Micha	+380(4593)433-4543	Chernobyl
Peggy Olsen	+1(212)38749-3872	Manhattan

4.2. Classe FrmContatos

Quando os botões de adicionar e de excluir são clicados, uma segunda janela responsável pelo cadastro surge. Esta nova janela possuirá campos de preenchimento, e os botões gravar e cancelar. Abaixo, sua implementação:

```
34 public class FrmContatos extends JFrame {
35
36     private static final long serialVersionUID = 1L;
37     private JPanel contentPane;
38     private JTextField txtNome;
39     private JTextField txtFone;
40     private JTextField txtEndereco;
41     private JButton btnGravar;
42     private int index;
43
44     private ArrayList<Pessoa> lista;
45
46     public FrmContatos(ArrayList<Pessoa> lista) {
47         this.lista = lista;
48         addWindowListener(new WindowAdapter() {
49             @Override
50             public void windowActivated(WindowEvent arg0) {
51                 if (btnGravar.getText().equals("Gravar")) {
52                     limparCampos();
53                 }
54             }
55         });
56
57         setTitle("Agenda - Contatos");
58         setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
59
60         setBounds(100, 100, 349, 221);
61         contentPane = new JPanel();
62         contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
63         setContentPane(contentPane);
64         contentPane.setLayout(null);
65         setLocationRelativeTo(null);
```

iniciando nova janela, gerando janela, configurando seus módulos

```

68     JLabel lblNome = new JLabel("Nome:");
69     lblNome.setFont(new Font("Ubuntu Light", Font.PLAIN, 14));
70     lblNome.setBounds(12, 19, 66, 14);
71     contentPane.add(lblNome);
72
73     JLabel lblTelefone = new JLabel("Telefone:");
74     lblTelefone.setFont(new Font("Ubuntu Light", Font.PLAIN, 14));
75     lblTelefone.setBounds(12, 59, 66, 14);
76     contentPane.add(lblTelefone);
77
78     JLabel lblEndereco = new JLabel("Endereço:");
79     lblEndereco.setFont(new Font("Ubuntu Light", Font.PLAIN, 14));
80     lblEndereco.setBounds(12, 99, 66, 14);
81     contentPane.add(lblEndereco);
82
83     txtEndereco = new JTextField();
84     txtEndereco.setFont(new Font("Ubuntu Light", Font.PLAIN, 14));
85     txtEndereco.setBounds(88, 92, 223, 28);
86     contentPane.add(txtEndereco);
87     txtEndereco.setColumns(10);
88
89     txtNome = new JTextField();
90     txtNome.setFont(new Font("Ubuntu Light", Font.PLAIN, 14));
91     txtNome.setBounds(88, 12, 223, 28);
92     contentPane.add(txtNome);
93     txtNome.setColumns(10);
94
95     txtFone = new JTextField();
96     txtFone.setFont(new Font("Ubuntu Light", Font.PLAIN, 14));
97     txtFone.setBounds(88, 52, 223, 28);
98     contentPane.add(txtFone);
99     txtFone.setColumns(10);
100

```

ainda configurando módulos da janela

```

102     JButton btnCancelar = new JButton("Cancelar");
103     btnCancelar.setFont(new Font("Ubuntu Mono", Font.PLAIN, 12));
104     btnCancelar.addActionListener(new ActionListener() {
105         public void actionPerformed(ActionEvent e) {
106             setVisible(false);
107             FrmContatosLista frm = new FrmContatosLista();
108             frm.setVisible(true);
109         }
110     });
111     btnCancelar.setBounds(185, 159, 89, 23);
112     contentPane.add(btnCancelar);
113
114     btnGravar = new JButton("Gravar");
115     btnGravar.setFont(new Font("Ubuntu Mono", Font.PLAIN, 12));
116     btnGravar.addActionListener(new ActionListener() {
117         public void actionPerformed(ActionEvent arg0) {
118             if (btnGravar.getText().equals("Gravar")) {
119                 String mens = inserirPessoa();
120                 JOptionPane.showMessageDialog(null, mens);
121             } else {
122                 String mens = alterarPessoa();
123                 JOptionPane.showMessageDialog(null, mens);
124             }
125             dispose();
126             FrmContatosLista frm = new FrmContatosLista();
127             frm.setVisible(true);
128         }
129     });
130     btnGravar.setBounds(68, 159, 89, 23);
131     contentPane.add(btnGravar);
132 }

```

configurando funções dos botões de cancelar e gravar

```

144● public String inserirPessoa() { // Chamada pelo Método do Botão Gravar
145
146
147     Pessoa p = new Pessoa(); // Cria Objeto da Classe Pessoa
148     Lista list = new Lista();
149
150     try { // Início do tratamento de exceção
151
152         GerArquivo.lerArquivo(list); // Carregando arquivo e adicionando a uma Lista Encadeada de Forma Ordenada
153
154         // Inserindo no Objeto do tipo Pessoa:
155         p.setNome(txtNome.getText());
156         p.setTelefone(txtFone.getText());
157         p.setEndereco(txtEndereco.getText());
158
159         list.adicionaOrdenado(p); // Adicionando Objeto do tipo Pessoa na Lista Ligada Ordenada
160         lista.add(p); // Adicionando Objeto do tipo Pessoa em um ArrayList que será usado pela JTable
161
162         GerArquivo.gravarArquivo(list, false); // Passa a lista Ordenada para a Classe que grava em arquivo de texto
163
164         return "Pessoa inserida com sucesso";
165
166     } catch (Exception e) { // fim do tratamento de exceção
167         return "Erro ao Inserir! Favor digitar apenas números na matrícula!";
168     }
169 }

```

configurando métodos de inserção estendidos pelo botão de gravação

```

174● public String alterarPessoa() {
175     Lista list = new Lista(); // Cria Objeto da Classe Lista ( Encadeada )
176     Pessoa p = lista.get(index); // Cria Objeto da Classe Pessoa
177
178     try {
179
180         GerArquivo.lerArquivo(list); // Carregando arquivo e adicionando a uma Lista Encadeada de Forma Ordenada
181         list.retira(p); // Remove Pessoa a ser editada da Lista Encadeada
182         lista.remove(p); // Remove Pessoa a ser editada do ArrayList
183
184         // Adiciona usando os campos editáveis:
185         p.setNome(txtNome.getText());
186         p.setTelefone(txtFone.getText());
187         p.setEndereco(txtEndereco.getText());
188
189         list.adicionaOrdenado(p); // Adiciona pessoa editada na Lista Encadeada
190         lista.add(p); // Adiciona pessoa editada no ArrayList
191
192         GerArquivo.gravarArquivo(list, false); // Grava Arquivo novamente
193
194         return "Pessoa alterada com sucesso";
195     } catch (Exception e) {
196         return "Erro ao Alterar!";
197     }
198 }
199

```

configurando método com funções que permitem a alteração das pessoas

```

212● public JButton getBtnGravar() {
213     return btnGravar;
214 }
215
216● public int getIndice() {
217     return index;
218 }
219
220● public void setIndice(int index) {
221     this.index = index;
222 }
223
224● public JTextField getTxtNome() {
225     return txtNome;
226 }
227
228● public JTextField getTxtFone() {
229     return txtFone;
230 }
231
232● public JTextField getTxtMatricula() {
233     return txtEndereco;
234 }
235 }

```

getters e setters da classe

Abaixo temos os resultados da implementação:



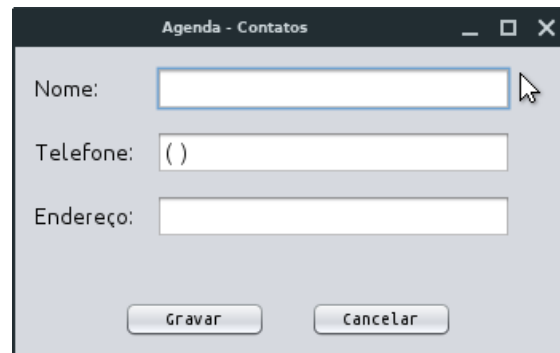
Agenda - Contatos

Nome: Bruno

Telefone: +55(62)99553-0338

Endereço: Goiânia

Alterar Cancelar



Agenda - Contatos

Nome:

Telefone: ()

Endereço:

Gravar Cancelar

5. Menu (modo terminal/texto)

Aqui encontra-se a implementação do menu em modo terminal (ou texto), pouco amigável ao usuário e também menos dinâmico que a interface gráfica supracitada, porém com grande valia, uma vez que as devidas lógicas são testadas antes mesmo de serem implementadas em um ambiente gráfico. Nota-se que um novo método foi implementado na classe Lista para substituir a ação automática de ordenação no JTable do ambiente gráfico. Tal método chamado de imprima invertido (imprimaInv) tem apenas a função de inverter a ordem de impressão do método anterior 'imprima', função que pode ser acessada no menu de opções número 5. Abaixo a implementação do código:

```
23 public class Teste {
24
25     static Scanner leia = new Scanner(System.in);
26
27     public static Pessoa obterItem() {
28         String nome, telefone, endereco;
29
30         leia.skip("\\R");
31         System.out.print("Nome:");
32         nome = leia.nextLine();
33         System.out.print("Telefone:");
34         telefone = (leia.next());
35         leia.skip("\\R"); // para que na hora do espaço entre palavras não de erro
36         System.out.print("Endereço:");
37         endereco = leia.nextLine();
38         Pessoa aux = new Pessoa(nome, telefone, endereco);
39         return aux;
40     }
41
42     public static int menu() {
43         int valor;
44         System.out.println("\nDigite:");
45         System.out.println("1 - para adicionar um item.");
46         System.out.println("2 - para remover um item.");
47         System.out.println("3 - para pesquisar um item.");
48         System.out.println("4 - para imprimir a lista em ordem crescente.");
49         System.out.println("5 - para imprimir a lista em ordem decrescente.");
50         System.out.println("6 - para gravar arquivo");
51         System.out.println("7 - para encerrar o programa.");
52         System.out.print("\nEscolha: ");
53         valor = Integer.parseInt(leia.next());
54         return valor;
55     }
```


implementação dos métodos para obter item e o menu de acesso

```
57● public static void main(String[] args) throws IOException {
58     int n;
59     Lista lista = new Lista();
60     Pessoa novo = null;
61     GerArquivo.lerArquivo(lista);
62
63     do {
64         n = menu();
65         switch (n) {
66
67             case 1:
68                 novo = obterItem();
69                 lista.adicionaOrdenado(novo);
70                 System.out.println("\n --- Agenda com " + lista.getTam() + " itens ---\n");
71                 novo = null;
72                 break;
73             case 2:
74                 Pessoa aux1 = new Pessoa();
75                 leia.skip("\nR");
76                 System.out.print("\nDigite o nome que deseja remover: ");
77                 aux1.setNome(leia.next());
78                 novo = lista.retira(aux1);
79                 if (novo == null)
80                     System.out.println("Erro!");
81                 else
82                     System.out.println("\n" + novo.toString() + "Removido(a) com sucesso");
83                 novo = null;
84                 break;
85             case 3:
86                 Pessoa aux2 = new Pessoa();
87                 System.out.print("\nDigite o nome que deseja pesquisar: ");
88                 aux2.setNome(leia.next());
89                 novo = lista.busca(aux2);
90                 if (novo == null)
91                     System.out.println("Erro!");
92         }
93     }
```

início do switch de menu

```
93         else
94             System.out.println(novo.toString());
95         System.out.println("\n --- Agenda " + lista.getTam() + " itens ---\n");
96         novo = null;
97         break;
98     case 4:
99         System.out.println("\n--- Agenda " + lista.getTam() + " itens ---\n");
100        System.out.println(lista.imprima());
101        break;
102    case 5:
103        System.out.println("\n --- Agenda com " + lista.getTam() + " itens ---\n");
104        System.out.println(lista.imprimaInv());
105        break;
106    case 6:
107        GerArquivo.gravarArquivo(lista);
108        System.out.println("Arquivo gravado com sucesso");
109        break;
110    case 7:
111        for (int i = 0; i < 100; i++) {
112            System.out.println();
113        }
114        System.out.println("Programa encerrando!");
115    }
116    } while (n != 7);
117
118 } while (n != 7);
119
120 }
121
122 }
```

fim do switch de menu

Temos os seguintes resultados:

```
Digite:
1 - para adicionar um item.
2 - para remover um item.
3 - para pesquisar um item.
4 - para imprimir a lista em ordem crescente.
5 - para imprimir a lista em ordem decrescente.
6 - para gravar arquivo
7 - para encerrar o programa.

Escolha: 1
Nome: Bruno
Telefone: +55(62)99553-0338
Endereço: Goiânia
```

menu de acesso opção de adição e cadastro

```
Digite:
1 - para adicionar um item.
2 - para remover um item.
3 - para pesquisar um item.
4 - para imprimir a lista em ordem crescente.
5 - para imprimir a lista em ordem decrescente.
6 - para gravar arquivo
7 - para encerrar o programa.

Escolha: 5

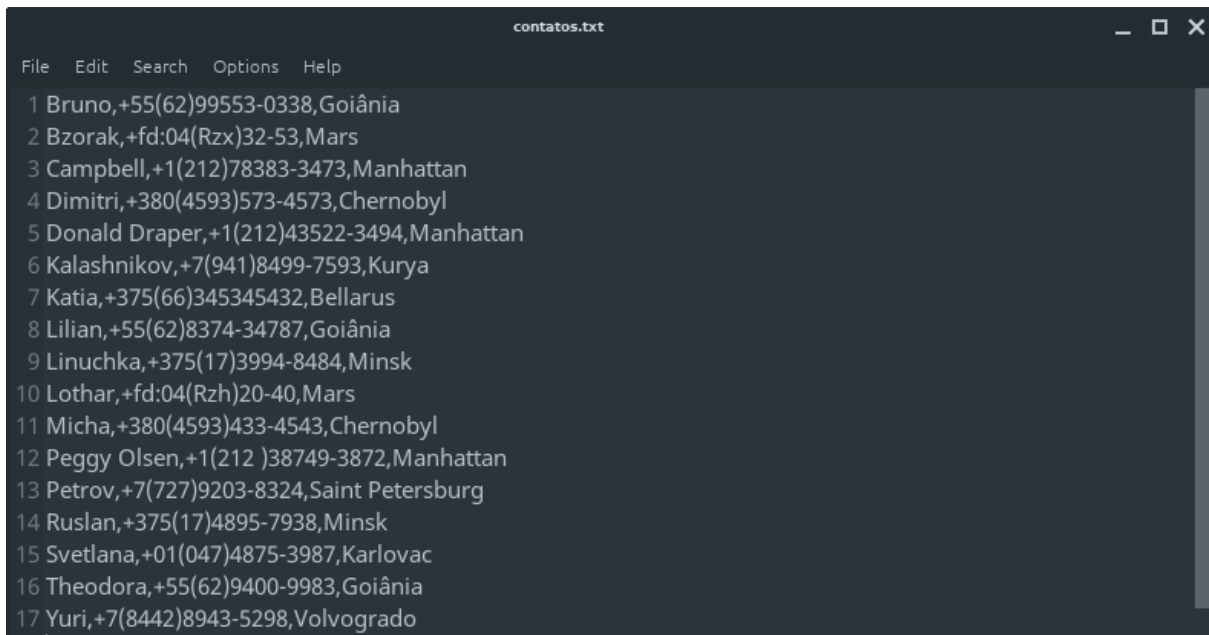
--- Agenda com 17 itens ---

Yuri,+7(8442)8943-5298,Volvogrado
Theodora,+55(62)9400-9983,Goiânia
Svetlana,+01(047)4875-3987,Karlovac
Ruslan,+375(17)4895-7938,Minsk
Petrov,+7(727)9203-8324,Saint Petersburg
Peggy Olsen,+1(212 )38749-3872,Manhattan
Micha,+380(4593)433-4543,Chernobyl
Lothar,+fd:04(Rzh)20-40,Mars
Linuchka,+375(17)3994-8484,Minsk
Lilian,+55(62)8374-34787,Goiânia
Katia,+375(66)345345432,Bellarus
Kalashnikov,+7(941)8499-7593,Kurya
Donald Draper,+1(212)43522-3494,Manhattan
Dimitri,+380(4593)573-4573,Chernobyl
Campbell,+1(212)78383-3473,Manhattan
Bzorak,+fd:04(Rzx)32-53,Mars
Bruno,+55(62)99553-0338,Goiânia
```

menu de acesso opção de imprimir a lista em ordem decrescente

6. Exemplo do arquivo gerado

Enfim, no final deste trabalho, disponibilizo um exemplo de arquivo gerado pelo programa contendo os contatos adicionados pelo mesmo. O arquivo com nome de contatos.txt deve estar na mesma pasta do programa e seu conteúdo descrito em CSV (comma separated values), ou seja, os atributos de cada pessoa separados por vírgulas. A seguir:



```
File Edit Search Options Help
1 Bruno,+55(62)99553-0338,Goiânia
2 Bzorak,+fd:04(Rzx)32-53,Mars
3 Campbell,+1(212)78383-3473,Manhattan
4 Dimitri,+380(4593)573-4573,Chernobyl
5 Donald Draper,+1(212)43522-3494,Manhattan
6 Kalashnikov,+7(941)8499-7593,Kurya
7 Katia,+375(66)345345432,Bellarus
8 Lilian,+55(62)8374-34787,Goiânia
9 Linuchka,+375(17)3994-8484,Minsk
10 Lothar,+fd:04(Rzh)20-40,Mars
11 Micha,+380(4593)433-4543,Chernobyl
12 Peggy Olsen,+1(212 )38749-3872,Manhattan
13 Petrov,+7(727)9203-8324,Saint Petersburg
14 Ruslan,+375(17)4895-7938,Minsk
15 Svetlana,+01(047)4875-3987,Karlovac
16 Theodora,+55(62)9400-9983,Goiânia
17 Yuri,+7(8442)8943-5298,Volvogrado
```