Agenda de Contatos – Interface Gráfica em Swing

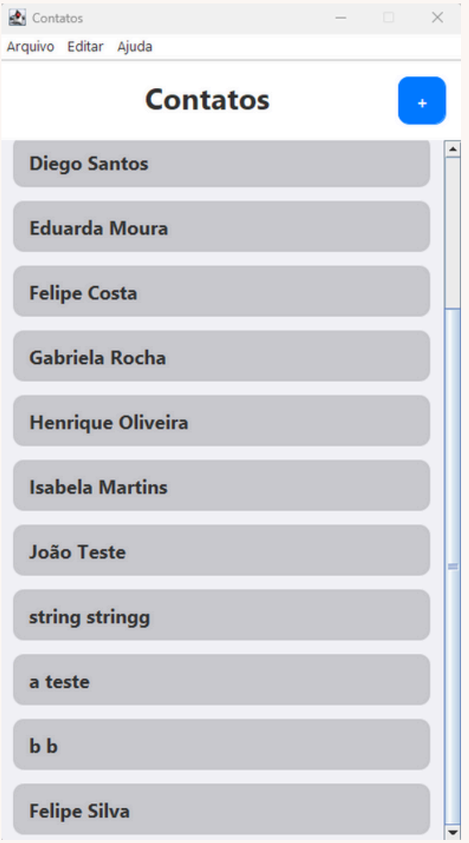
Disciplina: Programação Orientada a Objetos - Marco Mangan

Apresentação por Henrique Nunes e João Scarinci

PUCRS

Visão geral da aplicação

- Trabalho desenvolvido em Java + Swing
- Funcionalidades:
  - Cadastrar contatos
  - Listar contatos
  - Buscar contatos
  - Editar contatos
  - Apagar contatos
- Persistência de dados em arquivo de texto (cada linha, um contato)
- Interface com:
  - Lista de contatos (JList)
  - Menu superior (JMenuBar)
  - Modals (JPanel)
- Principais classes:
  - ContactsApp => janela principal da aplicação
  - Contact e AccessContacts => classes de armazenamento dos contatos
  - ContactAddPanel e ContactDetailPanel => classes das telas de adicionar e editar



Arquitetura geral do código

- app.ContactsApp
  - Extende **JFrame** e monta toda a janela principal
  - Cria menu, lista de contatos e integra com os outros painéis

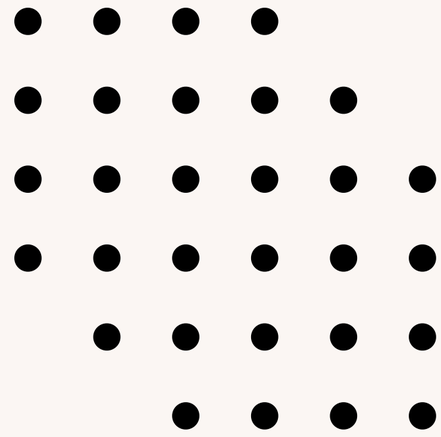
- contacts.Contact
  - Representa um contato (firstName, lastName, phone)
  - Possui um método helper **getFullName()** usado na lista

- contacts.AccessContacts
  - Cuida da persistência de dados:
    - list() => lê todos os contatos do arquivo
    - saveContacts(contact) => grava contato no arquivo
    - searchName(String) => busca contatos pelo nome

- panels.ContactAddPanel
  - Tela/modal para adicionar um novo contato

- panels.ContactDetailPanel
  - Tela/modal para ver, editar, remover um contato

- design.*
  - UITheme => cores e fontes
  - RoundedButton, RoundedPanel, RoundedBorder => componentes visuais customizados



Persistência em arquivo de texto

Como os contatos são armazenados:

- Cada linha do arquivo representa um contato
- Formato:
  - Nome;Sobrenome;Telefone
- Exemplo:
  - Ana;Silva;987654321
  - Bruno;Nunes;912345678
  - Carla;Pereira;99887766

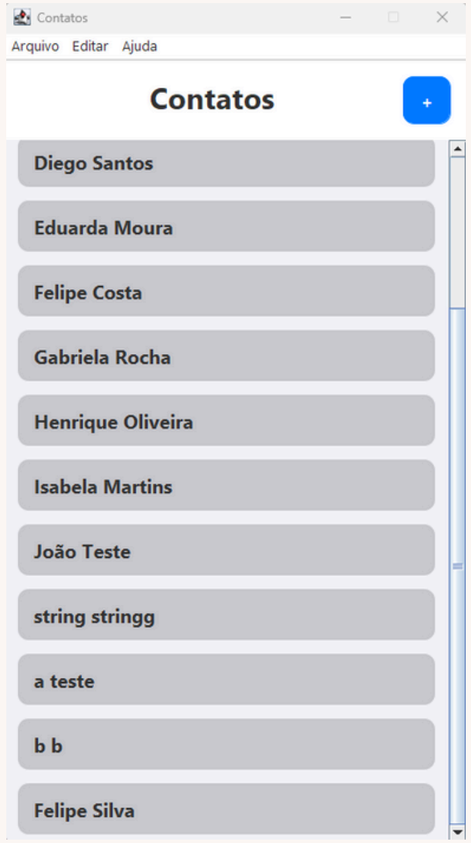
Códigos envolvidos:

- AccessContacts.list()
  - Lê o arquivo linha a linha
  - Separa por ;
  - Cria objetos Contact
  - Retorna List<Contact> para ContactsApp
- AccessContacts.saveContacts(List<Contact> contacts)
  - Recebe a lista de contatos em memória
  - Regrava o arquivo com o conteúdo atualizado

Tarefa 1: Visualizar a lista

- O que o usuário faz (pela interface):
  - Abre a aplicação
  - Vê todos os contatos um uma lista rolável
  - Cada contato aparece como um cartão com o nome completo

- Componentes gráficos envolvidos:
  - DefaultListModel<Contact> listModel
  - JList<Contact> contactsList
  - JScrollPanel scrollPane
  - ContactCellRenderer (classe interna)
    - Usa RoundedPanel + JLabel para deixar cada item como “card”

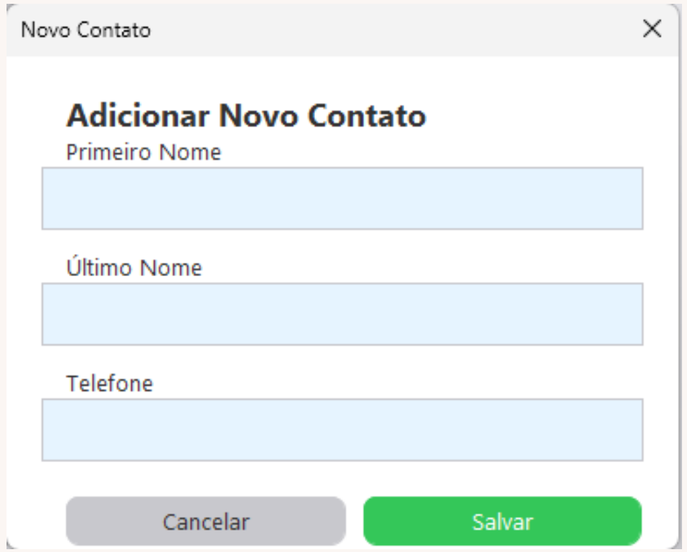


- Trecho de código:

```
listModel = new DefaultListModel<>();
for (Contact c : contacts) {
  listModel.addElement(c);
}

contactsList = new JList<>(listModel);
contactsList.setCellRenderer(new ContactCellRenderer());
contactsList.setBackground(UITheme.DARK_BACKGROUND);
```

Tarefa 2: Adicionar novo contato

- Como o usuário faz (passo a passo):
  - Pode clicar no **botão “+”** no cabeçalho ou em **Arquivo => Novo Contato**
  - Abre uma janela (**ContactAddPanel**) com os campos de formulário
  - Preencher **Nome, Sobrenome e Telefone**
  - Clique em **Salvar**
  - O contato aparece na lista e é gravado no arquivo



Componentes

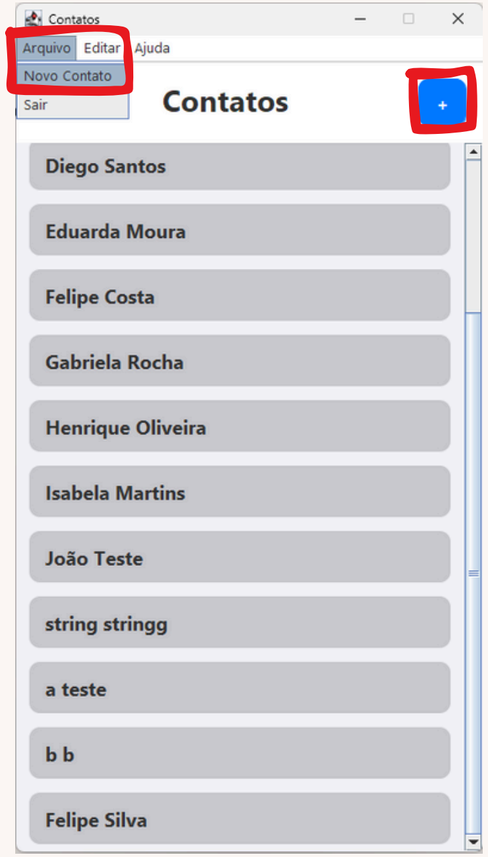
- Interface gráfica:
  - **RoundedButton addButton** (botão “+”)
  - **JMenuBar** => **JMenu** “Arquivo” => **JMenuItem** “Novo Contato”
  - Janela de formulário: **ContactAddPanel**

Códigos:

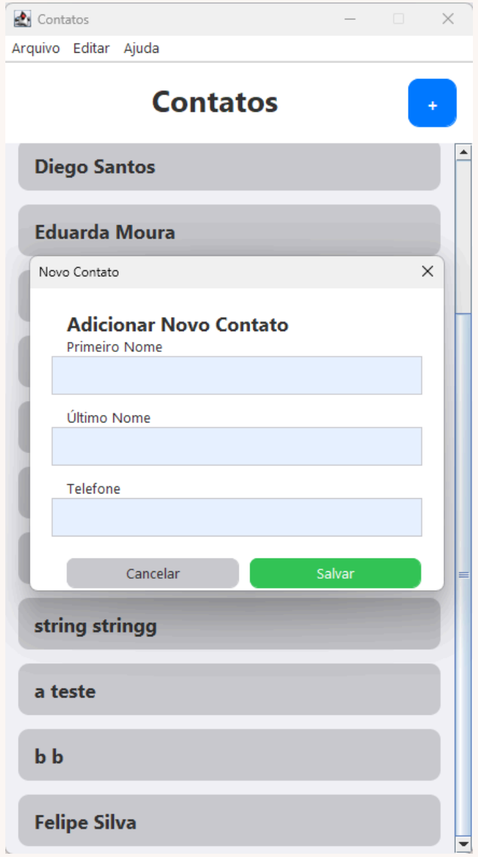
```
JMenuItem newContactItem = new JMenuItem("Novo Contato");
newContactItem.addActionListener(e -> addNewContact());

private void addNewContact() {
  ContactAddPanel dialog = new ContactAddPanel(this, null, this);
  dialog.setVisible(true);
}
```

```
public void addContact(Contact contact) {
  contacts.add(contact);
  listModel.addElement(contact);
  accessContacts.saveContacts(contacts);
  emptyStateLabel.setVisible(false);
  JOptionPane.showMessageDialog(this, "Contato adicionado com sucesso!",
      "Sucesso", JOptionPane.INFORMATION_MESSAGE);
}
```

Como o usuário faz (via aplicação)



=>



Tarefa 3: Ver detalhes e editar um contato

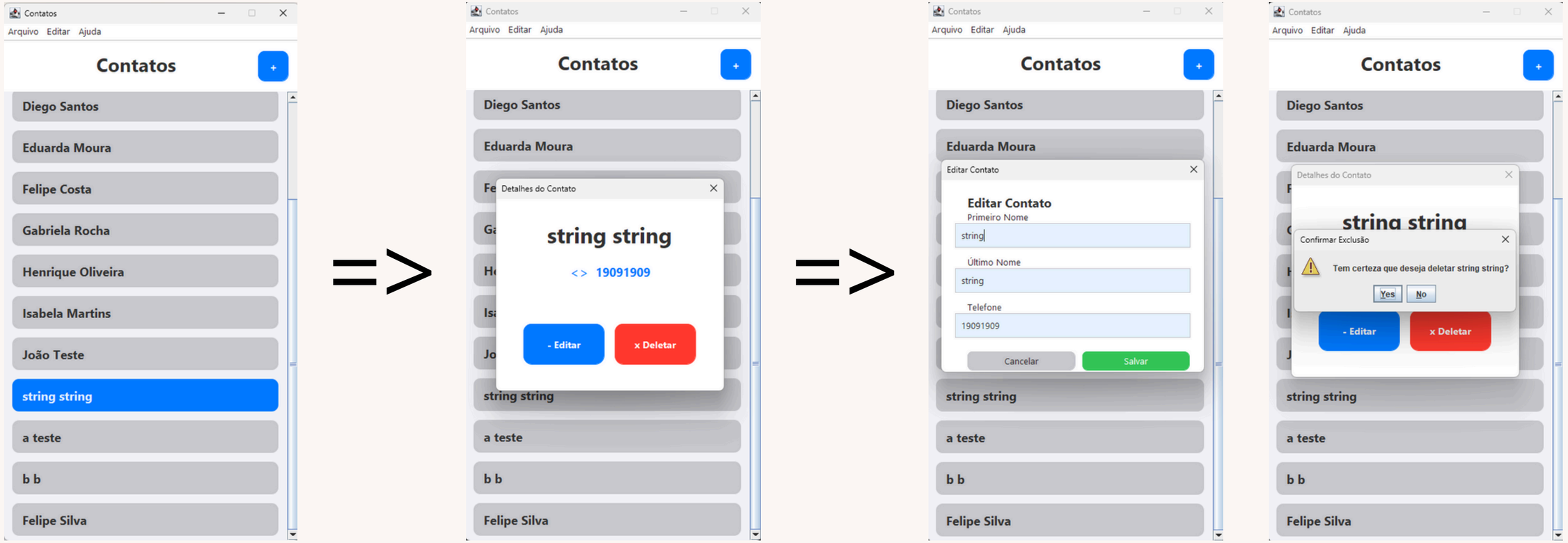
- Como o usuário faz (passo a passo):
  - Clica em um contato da lista
  - O app abre uma janela **(ContactDetailPanel)** com os dados do contato
  - Nessa janela o usuário pode:
    - Ver os dados do contato
    - Editar o contato
    - Excluir o contato

Códigos:

```
private void showContactDetail(Contact contact) {
  ContactDetailPanel dialog = new ContactDetailPanel(this, contact, this);
  dialog.setupLayout();
  dialog.setVisible(true);
}

public void updateContact(Contact contact) {
  accessContacts.saveContacts(contacts);
  contactsList.repaint();
  JOptionPane.showMessageDialog(this, "Contato atualizado com sucesso!",
      "Sucesso", JOptionPane.INFORMATION_MESSAGE);
}
```

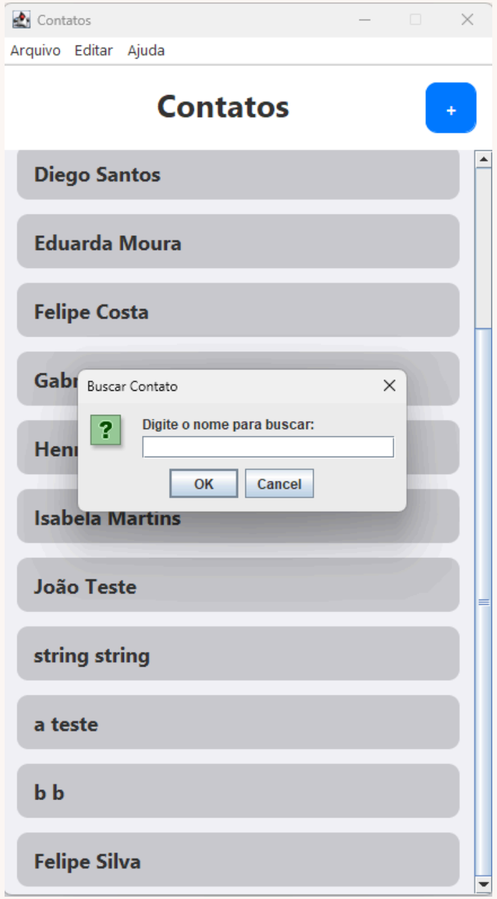
```
private void deleteContact() {
 int option = JOptionPane.showConfirmDialog(this,
   "Tem certeza que deseja deletar " + contact.getFullName() + "?",
   "Confirmar Exclusão",
   JOptionPane.YES_NO_OPTION,
   JOptionPane.WARNING_MESSAGE);

 if (option == JOptionPane.YES_OPTION) {
  parentApp.deleteContact(contact);
  dispose();
  JOptionPane.showMessageDialog(parentApp, "Contato deletado com sucesso!",
    "Sucesso", JOptionPane.INFORMATION_MESSAGE);
 }
}
```

Como o usuário faz (via aplicação)



Tarefa 4: Buscar contato pelo nome

- Como o usuário faz (passo a passo):
  - Vai no menu Editar => Buscar Contato
  - Digita um nome (ou parte do nome) na caixa de diálogo
  - Três cenários:
    - **Nenhum contato encontrado**: mostra mensagem
    - **Um contato encontrado**: abre direto a tela de detalhes
    - **Vários contatos**: aparece uma lista para o usuário escolher



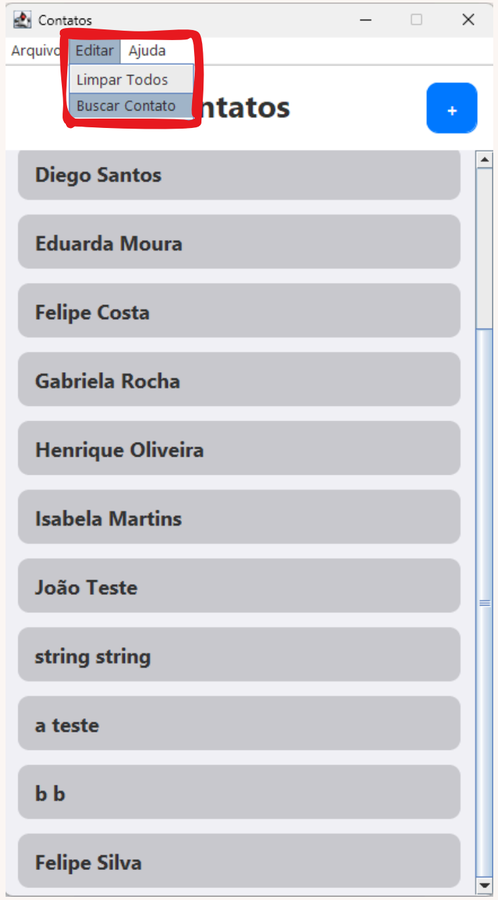
Componentes

- Interface gráfica:
  - Menu Editar **(JMenu editMenu)**
  - Item de menu **Buscar Contato (JMenuItem searchItem)**
  - Caixas de diálogo **JOptionPane**

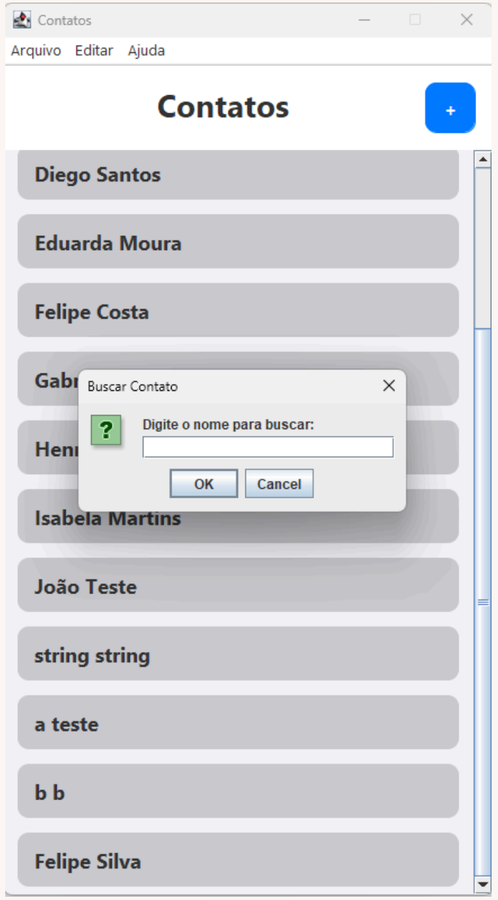
Códigos:

```
JMenuItem searchItem = new JMenuItem("Buscar Contato");
searchItem.addActionListener(e -> searchContact());
```

Como o usuário faz (via aplicação)



=>



Tarefa 5: Limpar todos os contatos

- Como o usuário faz (passo a passo):
  - Vai em Editar => Limpar Todos
  - Aparece uma caixa de confirmação (“Tem certeza?”)
  - Se confirmar, todos os contatos são deletados da lista e do arquivo

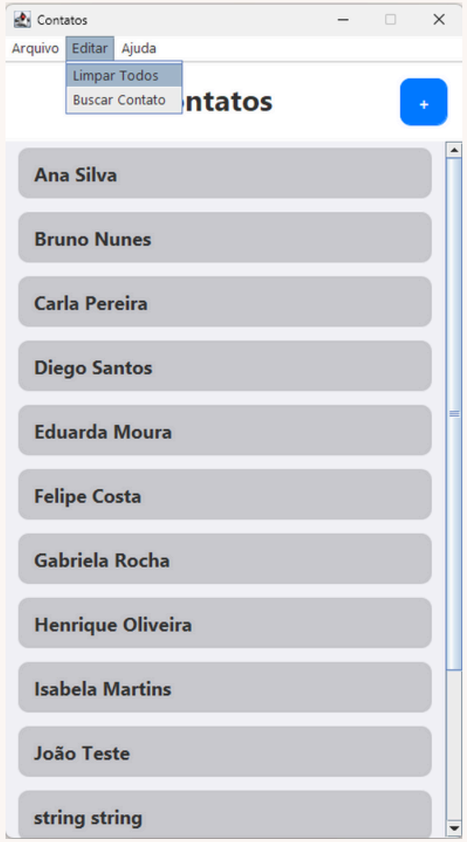
Códigos:

```
private void clearAllContacts() {
  int option = JOptionPane.showConfirmDialog(this,
      "Tem certeza que deseja limpar todos os contatos?",
      "Confirmar Limpeza",
      JOptionPane.YES_NO_OPTION,
      JOptionPane.WARNING_MESSAGE);

  if (option == JOptionPane.YES_OPTION) {
    contacts.clear();
    listModel.clear();
    accessContacts.saveContacts(contacts);
    emptyStateLabel.setVisible(true);
    JOptionPane.showMessageDialog(this, "Todos os contatos foram removidos!",
        "Sucesso", JOptionPane.INFORMATION_MESSAGE);
  }
}
```

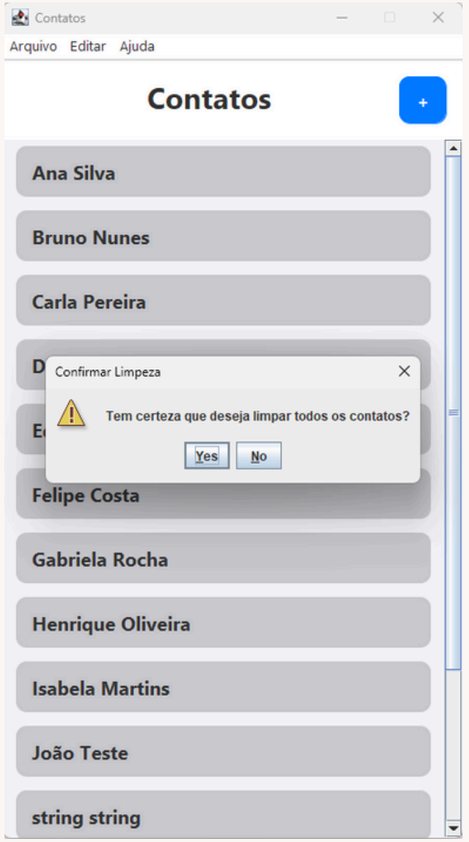
Componentes

- Interface gráfica:
  - Item de menu **Limpar Todos (JMenuItem clearAllItem)**
  - Diálogo de confirmação **(JOptionPane.showConfirmDialog)**
  - Mensagem de sucesso **(JOptionPane.showMessageDialog)**

Como o usuário faz (via aplicação)



=>



Ferramentas utilizadas

Documentação Java Swing

- Creating a GUI With Swing - The Java™ Tutorials
- https://docs.oracle.com/javase/tutorial/uiswing/
  - https://docs.oracle.com/javase/tutorial/uiswing/start/index.html
  - https://docs.oracle.com/javase/tutorial/uiswing/components/list.html
  - https://docs.oracle.com/javase/8/docs/api/index.html?javax%2Fswing%2Fpackage-summary.html

Assistência por Inteligência Artificial

- ChatGPT - OpenAI
- https://chatgpt.com/

Curso / Vídeo consultado

- Curso de Java Swing
- https://www.youtube.com/playlist?list=PLwH4Cv_WLhLbc4H-aOh3xFywPGxhaso_b