ânima EDUCAÇÃO

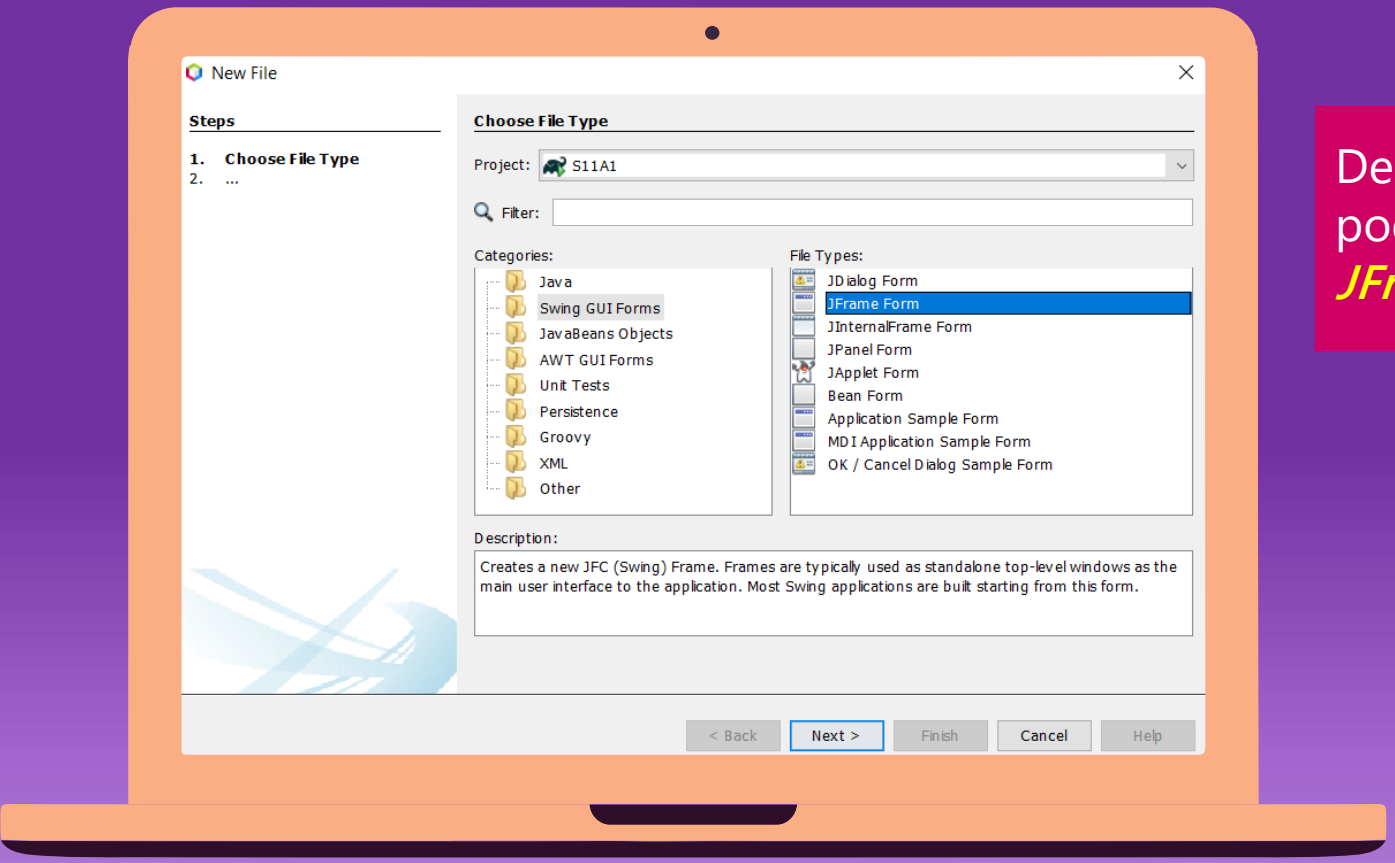
UNISUL
UNIVERSIDADE DO SUL DE SANTA CATARINA

PROGRAMAÇÃO DE SOLUÇÕES COMPUTACIONAIS

Prof. Ricardo Ribeiro Assink

MVC – *View*

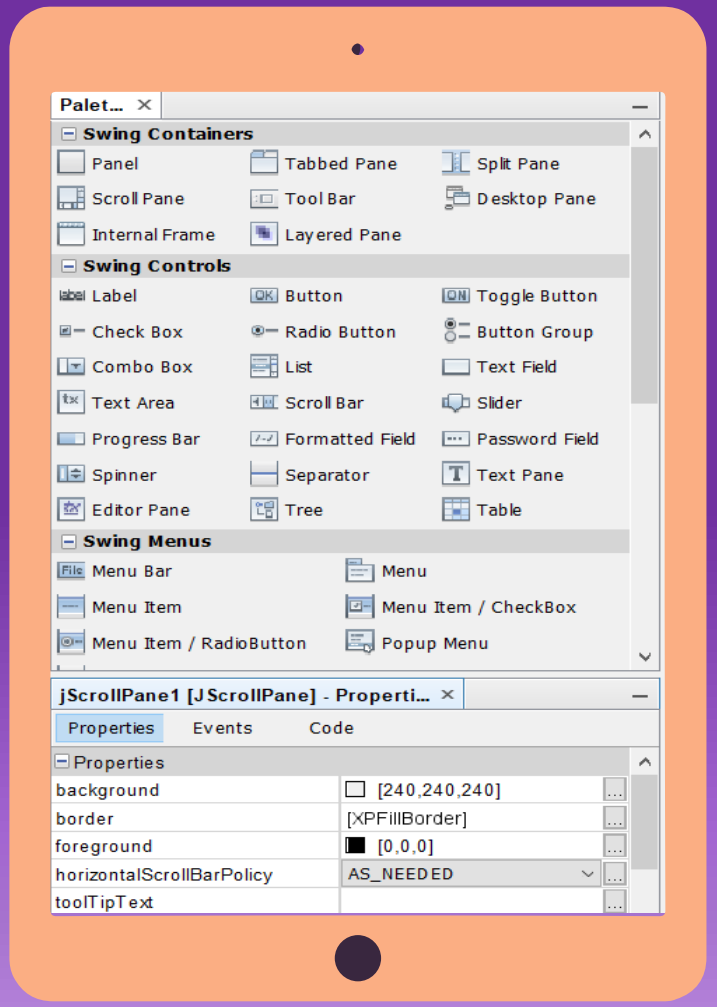
Para a construção das **interfaces gráficas** dentro da camada *View* utilizaremos a biblioteca *Swing* nativa do JDK.



Dentro de *javax.swing* existem diferentes classes que podem representar uma janela, no entanto, a classe *JFrame* fornece o padrão de janela comum em softwares.

Descrição de imagem:
Nesta aula teremos uma sequência muito grande de imagens que podem dificultar o seu entendimento, mesmo com descrição em cada imagem. Se você possui algum tipo de deficiência visual, procure o professor para orientações.

MVC – *View*



Use uma **IDE**!!

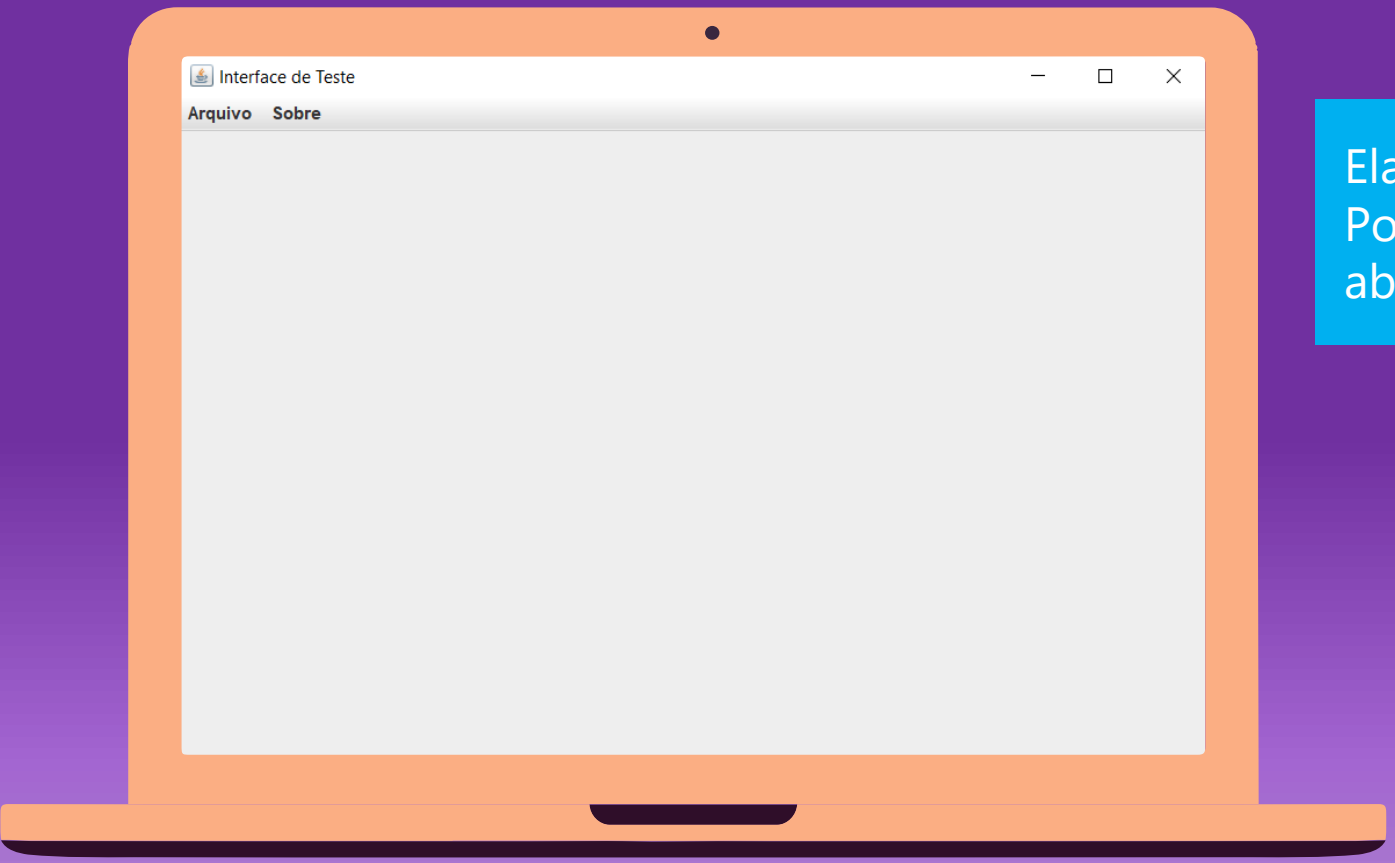
As **IDEs** fornecem conjunto de componentes e geram código automaticamente.
Isso facilita muito a construção das suas Interfaces Gráficas.

A **IDE** permitirá que você insira componentes dos mais diversos tipos com TOTAL personalização do visual e comportamento.

Descrição de imagem:
Nesta aula teremos uma sequência muito grande de imagens que podem dificultar o seu entendimento, mesmo com descrição em cada imagem. Se você possui algum tipo de deficiência visual, procure o professor para orientações.

MVC – *View*

Onde pretendemos chegar...



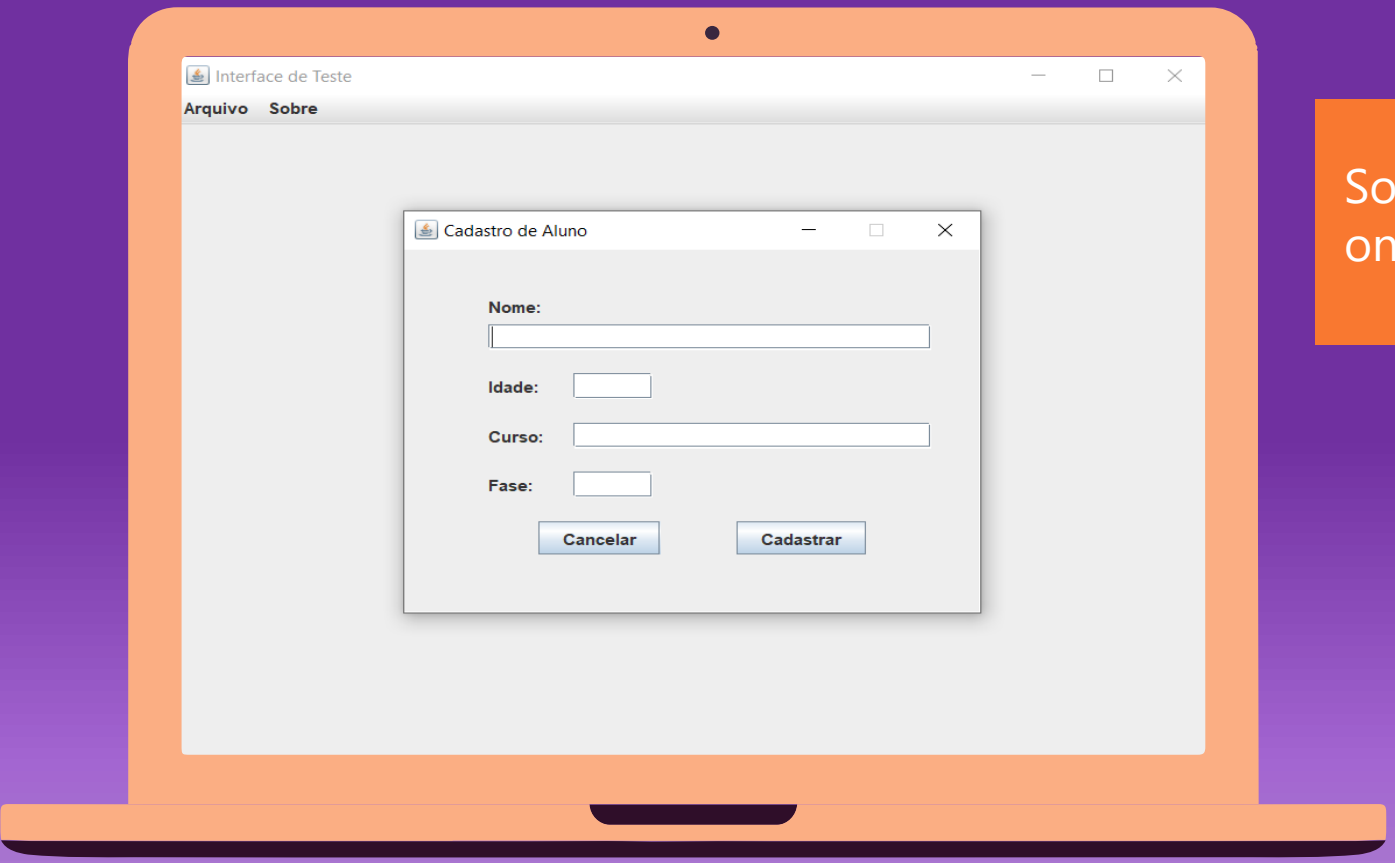
TelaPrincipal.java

Ela vai "disparar" nossa aplicação teste.
Possui um Menu superior(*jMenuBar*) com opções que abrem novas janelas ou fecham a aplicação.

Descrição de imagem:
Nesta aula teremos uma sequência muito grande de imagens que podem dificultar o seu entendimento, mesmo com descrição em cada imagem. Se você possui algum tipo de deficiência visual, procure o professor para orientações.

MVC – *View*

Onde pretendemos chegar...



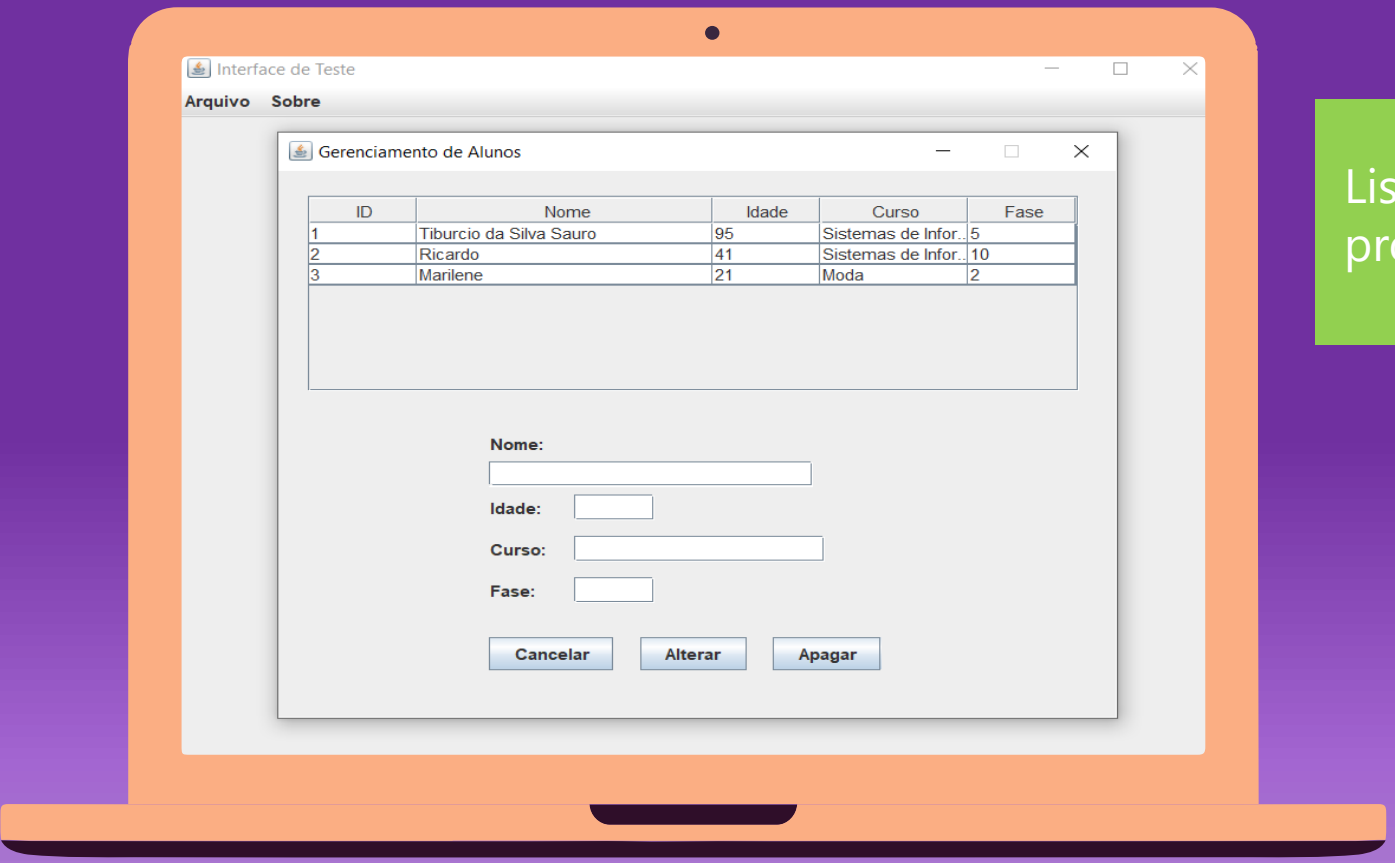
CadastroAluno.java

Solicita dados ao usuário e os envia para **AlunoControl** onde serão processados.

Descrição de imagem:
Nesta aula teremos uma sequência muito grande de imagens que podem dificultar o seu entendimento, mesmo com descrição em cada imagem. Se você possui algum tipo de deficiência visual, procure o professor para orientações.

MVC – *View*

Onde pretendemos chegar...



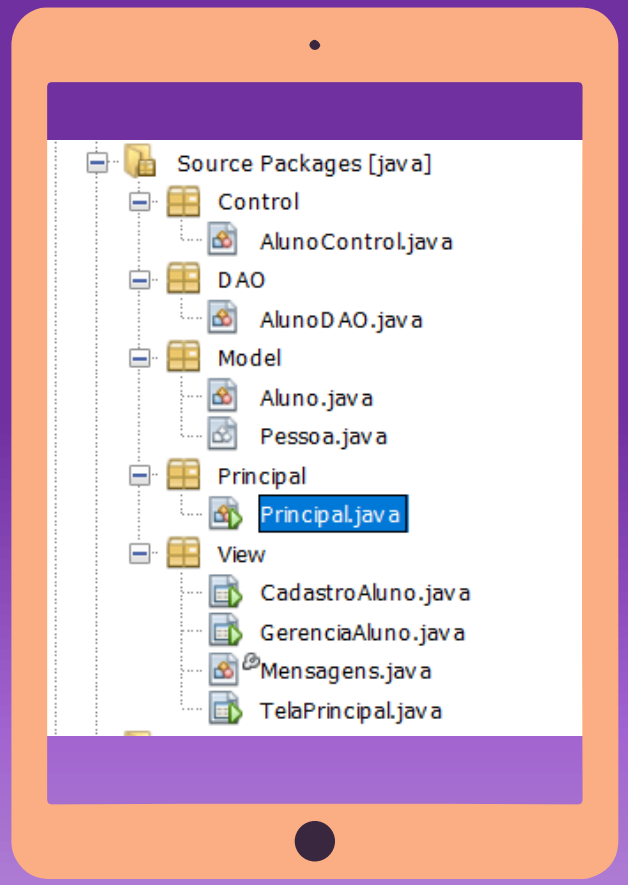
GerenciaAluno.java

Lista, Edita e Apaga Alunos utilizando **Aluno.java** para processamento das operações.

Descrição de imagem:
Nesta aula teremos uma sequência muito grande de imagens que podem dificultar o seu entendimento, mesmo com descrição em cada imagem. Se você possui algum tipo de deficiência visual, procure o professor para orientações.

MVC – *View*

Onde pretendemos chegar...



Estrutura completa dos arquivos distribuídos nas camadas.
* Não usaremos a camada Control neste semestre.

Descrição de imagem:
Nesta aula teremos uma sequência muito grande de imagens que podem dificultar o seu entendimento, mesmo com descrição em cada imagem. Se você possui algum tipo de deficiência visual, procure o professor para orientações.

MVC – *View*

Roteiro: Acompanhe o professor diretamente na IDE.

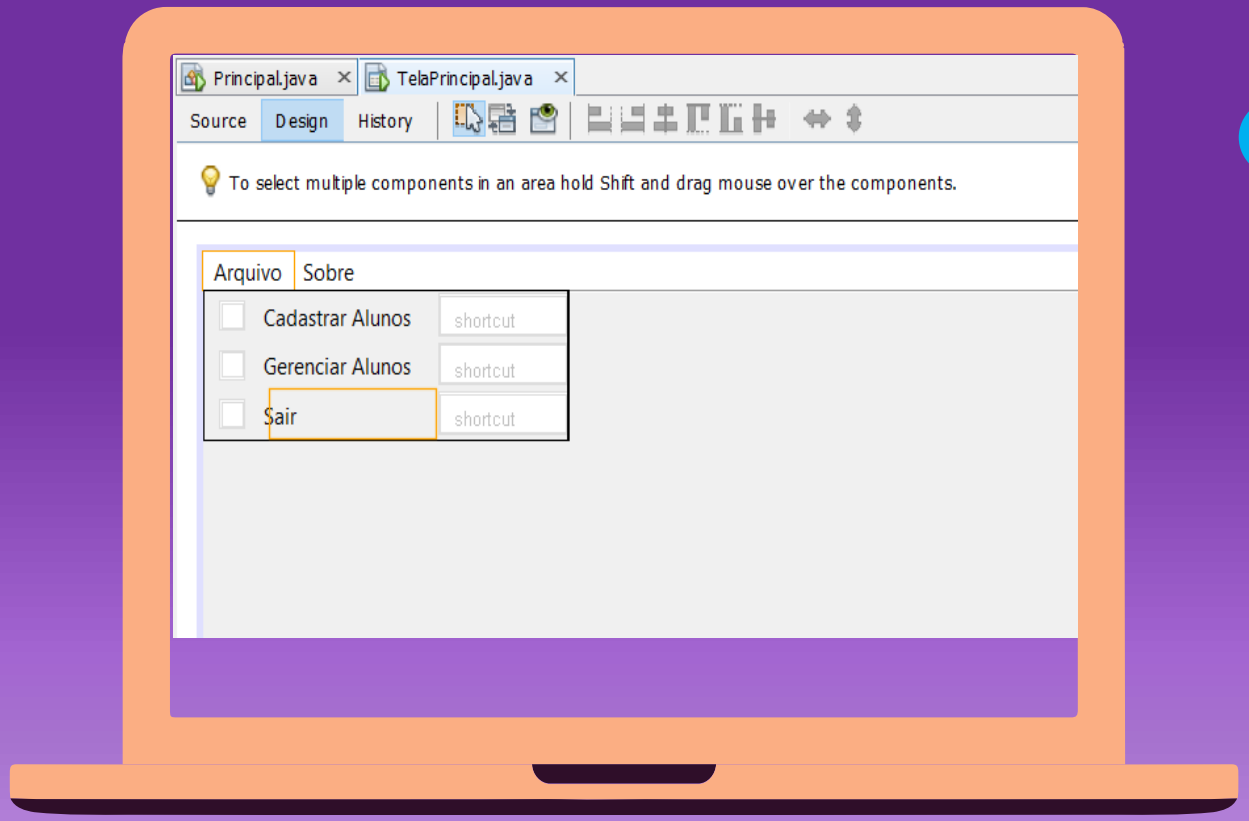
1. Criar arquivo *JFrame* dentro da camada, *View.TelaPrincipal.java*

2. Dentro de *Principal.Principal.java* referenciamos a abertura desta Tela Principal dentro do método *main*.

```
package Principal;

import View.TelaPrincipal;

public class Principal {

    public static void main(String[] args) {
        TelaPrincipal objetotela = new TelaPrincipal();  // Cria objeto do JFrame TelaPrincipal
        objetotela.setVisible(true); // Abre a TelaPrincipal
    }
}
```

MVC – *View*

Roteiro: Acompanhe o professor diretamente na IDE.

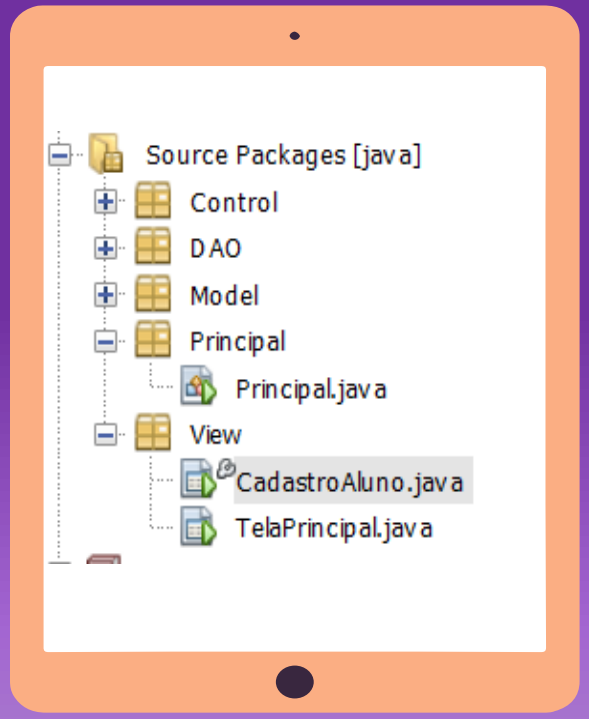


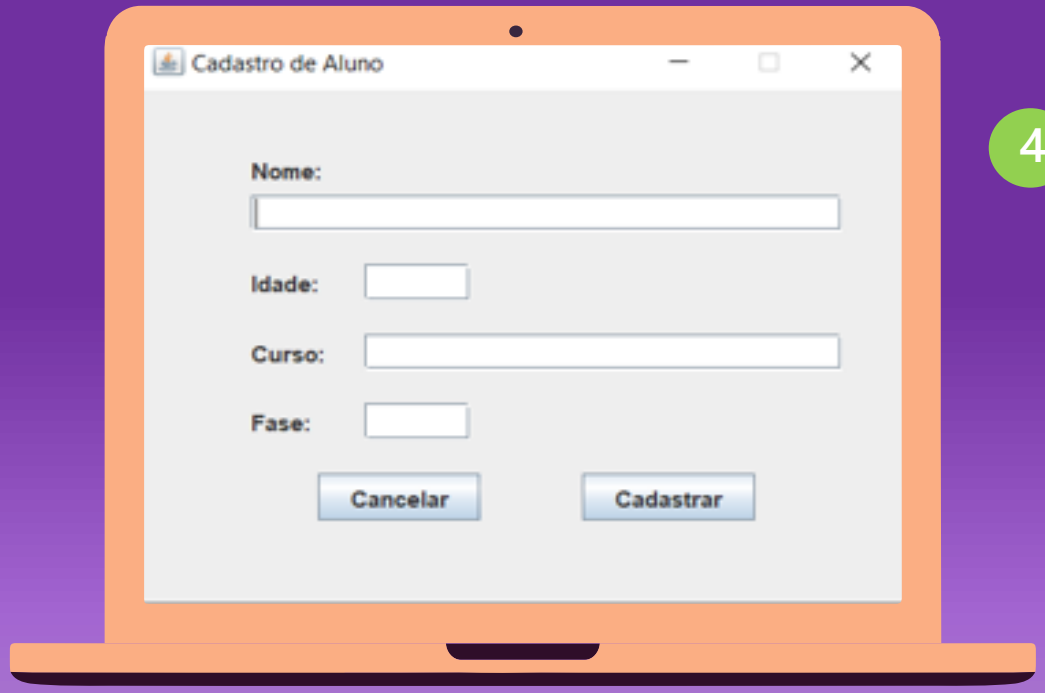
3 Incluir um *jMenuBar* em *View.TelaPrincipal.java* Você deve modificar este menu incluindo e editando seus itens.

Descrição de imagem:
Nesta aula teremos uma sequência muito grande de imagens que podem dificultar o seu entendimento, mesmo com descrição em cada imagem. Se você possui algum tipo de deficiência visual, procure o professor para orientações.

MVC – *View*

Roteiro: Acompanhe o professor diretamente na IDE.





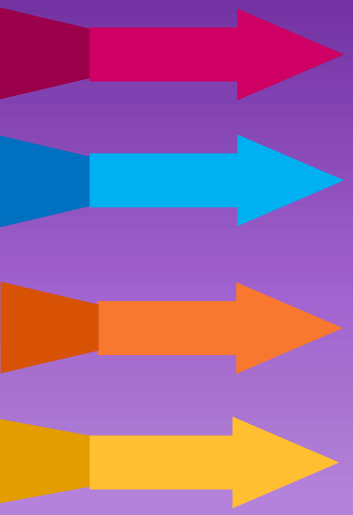
4 Vamos criar mais um arquivo *JFrame* dentro da camada, *View.CadastroAluno.java* Inclua componentes referentes aos campos que queremos cadastrar.

Descrição de imagem:
Nesta aula teremos uma sequência muito grande de imagens que podem dificultar o seu entendimento, mesmo com descrição em cada imagem. Se você possui algum tipo de deficiência visual, procure o professor para orientações.

MVC – *View*

Roteiro: Acompanhe o professor diretamente na IDE.

5 Agora, dentro do *jMenuBar* selecione o item de menu "**Cadastrar Alunos**" e configure o evento *actionPerformed*. Isso fará que a tela de cadastro de alunos seja aberta ao clicar neste item de menu.

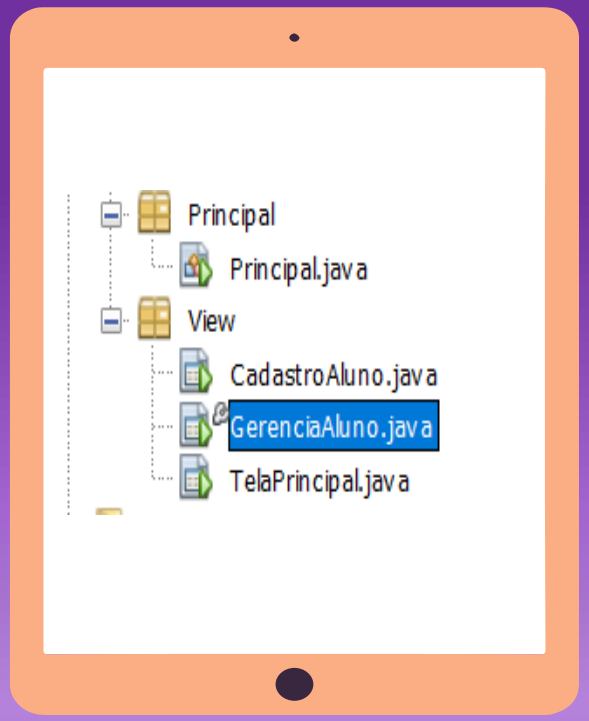


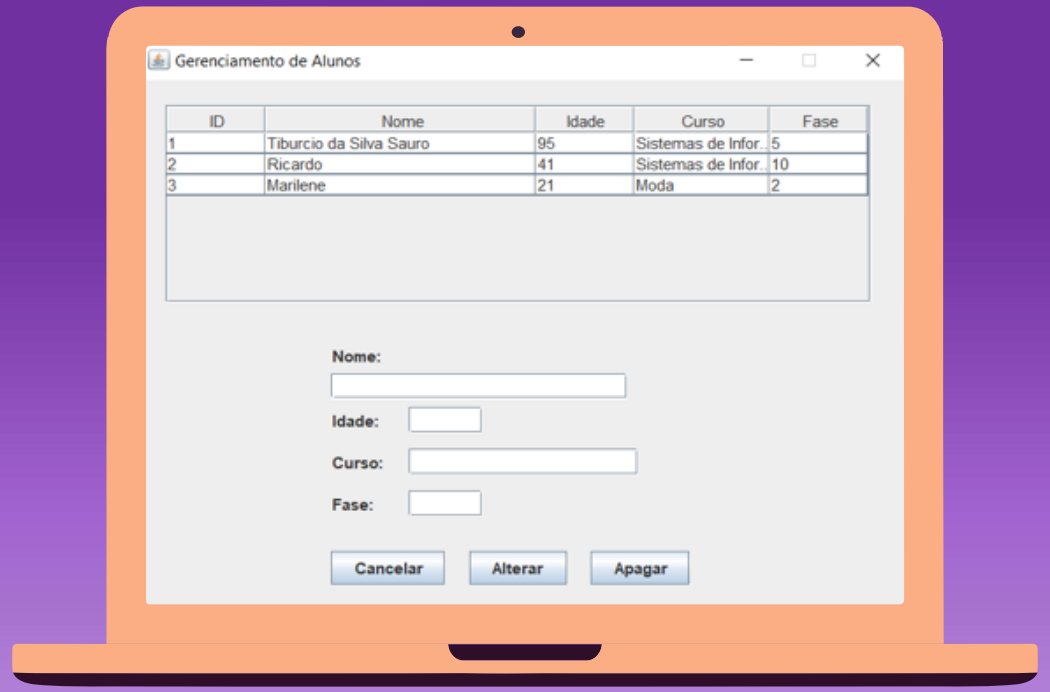
```
private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {

    CadastroAluno objeto = new CadastroAluno();
    objeto.setVisible(true);

}
```

MVC – *View*

Roteiro: Acompanhe o professor diretamente na IDE.

6 Vamos criar mais um arquivo *JFrame* dentro da camada, *View.GerenciaAluno.java*
Inclua componentes referentes aos campos que queremos editar, botões que irão apagar, editar e cancelar.
Por fim adicione uma *jTable* e configure as colunas da tabela.





Descrição de imagem:
Nesta aula teremos uma sequência muito grande de imagens que podem dificultar o seu entendimento, mesmo com descrição em cada imagem. Se você possui algum tipo de deficiência visual, procure o professor para orientações.

MVC – *View*

Roteiro: Acompanhe o professor diretamente na IDE.

7 Dentro do *jMenuBar* selecione o item de menu "**Gerenciar Alunos**" e configure o evento *actionPerformed*. Isso fará que a tela de Gerenciamento de alunos seja aberta ao clicar no item de menu.

```
private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt)
{
        GerenciaAluno objeto = new GerenciaAluno();
        objeto.setVisible(true);
    }
```

MVC – *View*

Roteiro: Acompanhe o professor diretamente na IDE.

8 Dentro do *jMenuBar* selecione o item "**Sair**" e configure o evento *actionPerformed*. Isso fará que o software feche.

```
private void jMenuItem3ActionPerformed(java.awt.event.ActionEvent evt)
{
        System.exit(0);
    }
```

MVC – *View*

Roteiro: Acompanhe o professor diretamente na IDE.

9 Neste momento, temos as interfaces gráficas construídas, já configuramos os itens de menu e já configuramos as ações dos botões cancelar.

01 Agora precisamos configurar a conexão entre as **Views** (CadastroAluno e GerenciaAluno) e a camada **Model**. Dentro das **Views**, vamos criar um atributo privado do tipo **Aluno** e vamos inicializá-lo no construtor. Toda vez que alguma classe instanciar objetos de **CadastroAluno**, automaticamente, este já instancia um objeto de **Aluno**.

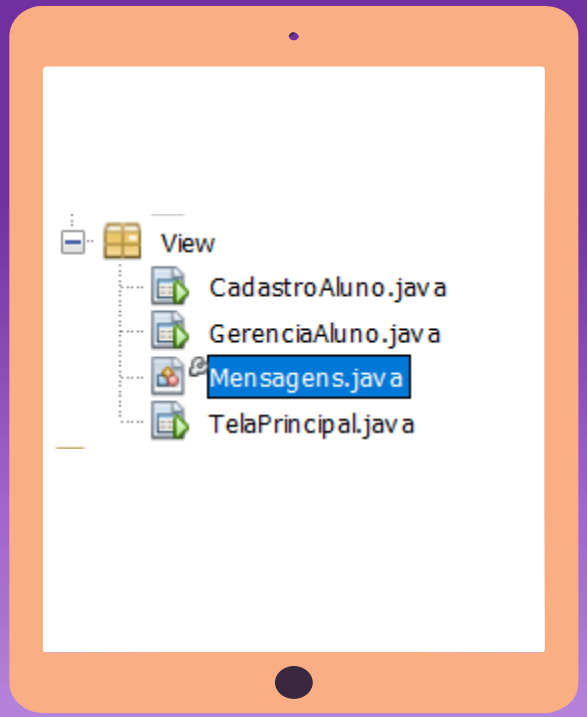
02 Veja abaixo trecho de código de *View.CadastroAluno.java*
Faça o mesmo em *View.GerenciaAluno.java*

```
public class CadastroAluno extends javax.swing.JFrame {

    private Aluno objaluno; // cria o vínculo com Aluno.java

    public CadastroAluno() {
        initComponents();
        this.objaluno = new Aluno(); // carrega o objeto de aluno
    }
```

MVC – *View*

Roteiro: Acompanhe o professor diretamente na IDE.

10 Vamos criar uma classe dentro da camada, *View.Mensagens.java* que irá nos auxiliar com os tratamentos de Erros.




```
package View;

public class Mensagens extends Exception{
    Mensagens(String msg){
        super(msg);
    }
}
```

MVC – *View*

Roteiro: Acompanhe o professor diretamente na IDE.

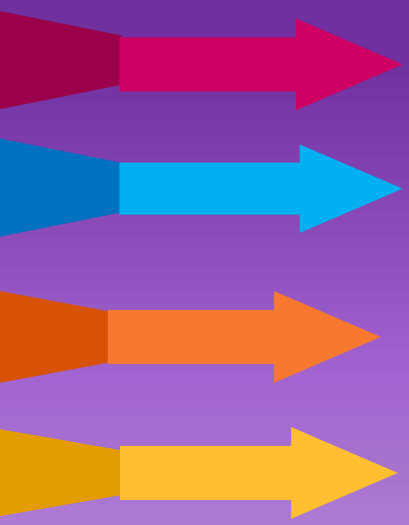
11 Dentro de *View.CadastroAluno.java* vamos implementar o evento ***actionPerformed*** do botão "**Cadastrar**".

```
private void b_cadastrarActionPerformed(java.awt.event.ActionEvent evt) {

    try {
      // recebendo e validando dados da interface gráfica.
      String nome = "";
      int idade = 0;
      String curso = "";
      int fase = 0;

      if (this.c_nome.getText().length() < 2) {
         throw new Mensagens("Nome deve conter ao menos 2 caracteres.");
      } else {
         nome = this.c_nome.getText();
      }

      if (this.c_idade.getText().length() <= 0) {
         throw new Mensagens("Idade deve ser número e maior que zero.");
      } else {
         idade = Integer.parseInt(this.c_idade.getText());
      }

      if (this.c_curso.getText().length() < 2) {
         throw new Mensagens("Curso deve conter ao menos 2 caracteres.");
      } else {
         curso = this.c_curso.getText();
      }

      if (this.c_fase.getText().length() <= 0) {
         throw new Mensagens("Fase deve ser número e maior que zero.");
      } else {
         fase = Integer.parseInt(this.c_fase.getText());
      }

      // envia os dados para cadastrar
      if (this.objaluno.InsertAlunoBD(curso, fase, nome, idade)) {
         JOptionPane.showMessageDialog(rootPane, "Aluno Cadastrado com Sucesso!");

         // limpa campos da interface
         this.c_nome.setText("");
         this.c_idade.setText("");
         this.c_curso.setText("");
         this.c_fase.setText("");
      }
      System.out.println(this.controlador.getMinhaLista().toString());
    } catch (Mensagens erro) {
      JOptionPane.showMessageDialog(null, erro.getMessage());
    } catch (NumberFormatException erro2) {
      JOptionPane.showMessageDialog(null, "Informe um número.");
    }
}
```

MVC – *View*

Roteiro: Acompanhe o professor diretamente na IDE.

12 Dentro do método *carregaTabela() , dentro da View,* receber o ArrayList<Aluno> de DAO e processar tudo dentro da *View*. Lembre-se que neste caso você precisará importar a classe **Aluno** da camada *Model* .

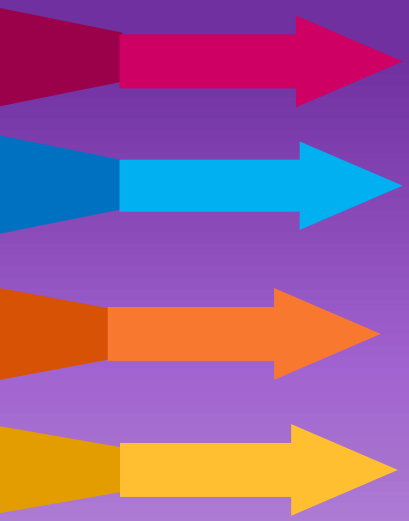


```
public void carregaTabela() {

    DefaultTableModel modelo = (DefaultTableModel) this.jTableAlunos.getModel();
    modelo.setNumRows(0);

    ArrayList<Aluno> minhalista = new ArrayList<>();
    minhalista = objaluno.getMinhaLista();

    for (Aluno a : minhalista) {
        modelo.addRow(new Object[]{
            a.getId(),
            a.getNome(),
            a.getIdade(),
            a.getCurso(),
            a.getFase()
        });
    }
}
```

MVC – *View*

Roteiro: Acompanhe o professor diretamente na IDE.

13 Agora que já estamos carregando a *jTable* com dados vamos criar um tratamento para capturar a seleção de linha no ato do clique do mouse. Usaremos para selecionar qual Aluno queremos apagar ou editar, transferindo os valores para o campos na parte de baixo do nosso exemplo.



```
private void jTableAlunosMouseClicked(java.awt.event.MouseEvent evt) {

    if (this.jTableAlunos.getSelectedRow() != -1) {

        String nome = this.jTableAlunos.getValueAt(this.jTableAlunos.getSelectedRow(), 1).toString();
        String idade = this.jTableAlunos.getValueAt(this.jTableAlunos.getSelectedRow(), 2).toString();
        String curso = this.jTableAlunos.getValueAt(this.jTableAlunos.getSelectedRow(), 3).toString();
        String fase = this.jTableAlunos.getValueAt(this.jTableAlunos.getSelectedRow(), 4).toString();

        this.c_nome.setText(nome);
        this.c_idade.setText(idade);
        this.c_curso.setText(curso);
        this.c_fase.setText(fase);

    }
}
```

MVC – *View*

Roteiro: Acompanhe o professor diretamente na IDE.

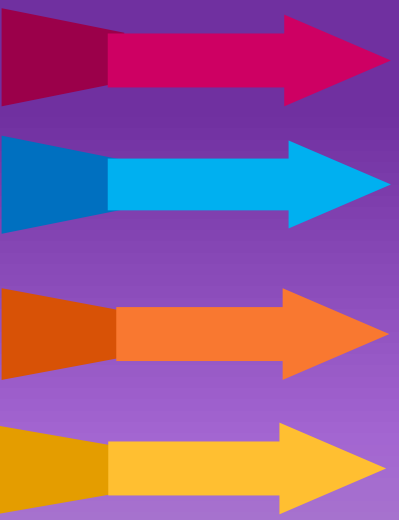
15 Dentro de *View.GerenciaAluno.java* vamos implementar o evento ***actionPerformed*** do botão "**Editar**".

```
private void b_alterarActionPerformed(java.awt.event.ActionEvent evt) {
        try {
            // recebendo e validando dados da interface gráfica.
            int id = 0;
            String nome = "";
            int idade = 0;
            String curso = "";
            int fase = 0;

            if (this.c_nome.getText().length() < 2) {
                throw new Mensagens("Nome deve conter ao menos 2 caracteres.");
            } else {
                nome = this.c_nome.getText();
            }

            if (this.c_idade.getText().length() <= 0) {
                throw new Mensagens("Idade deve ser número e maior que zero.");
            } else {
                idade = Integer.parseInt(this.c_idade.getText());
            }
            if (this.c_curso.getText().length() < 2) {
                throw new Mensagens("Curso deve conter ao menos 2 caracteres.");
            } else {
                curso = this.c_curso.getText();
            }
            if (this.c_fase.getText().length() <= 0) {
                throw new Mensagens("Fase deve ser número e maior que zero.");
            } else {
                fase = Integer.parseInt(this.c_fase.getText());
            }

            if (this.jTableAlunos.getSelectedRow() == -1) {
                throw new Mensagens("Primeiro Selecione um Aluno para Alterar");
            } else {
id = Integer.parseInt(this.jTableAlunos.getValueAt(this.jTableAlunos.getSelectedRow(), 0).toString());
            }

            // envia os dados para o Controlador processar
            if (this.objaluno.UpdateAlunoBD(curso, fase, id, nome, idade)) {
                // limpa os campos
                this.c_nome.setText("");
                this.c_idade.setText("");
                this.c_curso.setText("");
                this.c_fase.setText("");
                JOptionPane.showMessageDialog(rootPane, "Aluno Alterado com Sucesso!");

            }
            System.out.println(this.controlador.getMinhaLista().toString());
        } catch (Mensagens erro) {
            JOptionPane.showMessageDialog(null, erro.getMessage());
        } catch (NumberFormatException erro2) {
            JOptionPane.showMessageDialog(null, "Informe um número.");
        } finally {
            carregaTabela(); // atualiza a tabela.
        }     }
```

MVC – *View*

Roteiro: Acompanhe o professor diretamente na IDE.

16 Por fim, dentro de *View.GerenciaAluno.java* vamos implementar o evento ***actionPerformed*** do botão "**Apagar**".



```
private void b_apagarActionPerformed(java.awt.event.ActionEvent evt) {
        try {
            // validando dados da interface gráfica.
            int id = 0;
            if (this.jTableAlunos.getSelectedRow() == -1) {
                throw new Mensagens("Primeiro Selecione um Aluno para APAGAR");
            } else {
                id = Integer.parseInt(this.jTableAlunos.getValueAt(
this.jTableAlunos.getSelectedRow(), 0).toString()
);
            }

            // retorna 0 -> primeiro botão | 1 -> segundo botão | 2 -> terceiro botão
            int resposta_usuario = JOptionPane.showConfirmDialog(null,
"Tem certeza que deseja APAGAR este Aluno ?");

            if (resposta_usuario == 0) { // clicou em SIM

                // envia os dados para processar
                if (this.objaluno.DeleteAlunoBD(id)) {
                    // limpa os campos
                    this.c_nome.setText("");
                    this.c_idade.setText("");
                    this.c_curso.setText("");
                    this.c_fase.setText("");
                    JOptionPane.showMessageDialog(rootPane, "Aluno Apagado com Sucesso!");
                }
            }
            System.out.println(this.controlador.getMinhaLista().toString());
        } catch (Mensagens erro) {
            JOptionPane.showMessageDialog(null, erro.getMessage());
        } finally {
            // atualiza a tabela.
            carregaTabela();
        }
    }
```

EXERCÍCIO 40

Primeiro, execute o guia passo a passo apresentado neste documento.
Depois tente implementar as interfaces de ao menos 1 dos módulos solicitados para a A3.

ânima EDUCAÇÃO

UNISUL
UNIVERSIDADE DO SUL DE SANTA CATARINA

FIM