Interface Gráfica

Prof. Bruno Gomes

bruno.gomes@ifrn.edu.br

Programação Orientada a Objetos

Agenda

- JTextArea
- JMenuBar
- JTable
- JDesktopPane e JInternalFrame

Componente JTextArea

- Fornece uma área para manipulação de múltiplas linhas de texto

Componente JTextArea

```
public class Frame extends JFrame {
  JTextArea textArea;

  public Frame() {
    super("Titulo");

    textArea = new JTextArea(10, 15);
    getContentPane().add(textArea);

    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    pack();
    setVisible(true);
  }

  public static void main(String[] args) {
     Frame frame = new Frame();
  }
}
```

Componente JTextArea

```
public class Frame extends JFrame {
   JTextArea textArea;
   Box box;
   public Frame() {
      super("Titulo");

      textArea = new JTextArea(10, 15);

      box = Box.createHorizontalBox();
      box.add(new JScrollPane(textArea));

      getContentPane().add(box);

      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      pack();
      setVisible(true);
   }
}
```

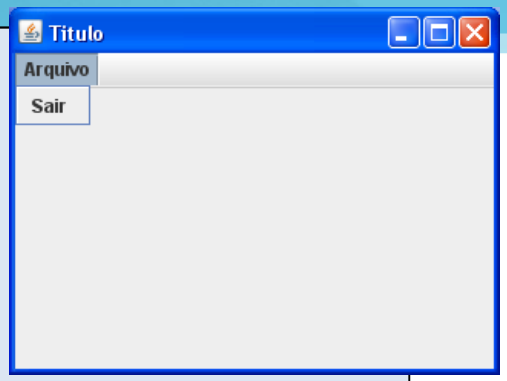
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA RIO GRANDE DO NORTE Campus Natal - Central

Componente JMenuBar

- Corresponde a um Menu
- Permite ao usuário realizar ações sem poluir desnecessariamente uma interface gráfica com o usuário
- Utiliza:
  - JMenuBar
  - Jmenu
  - JMenuItem

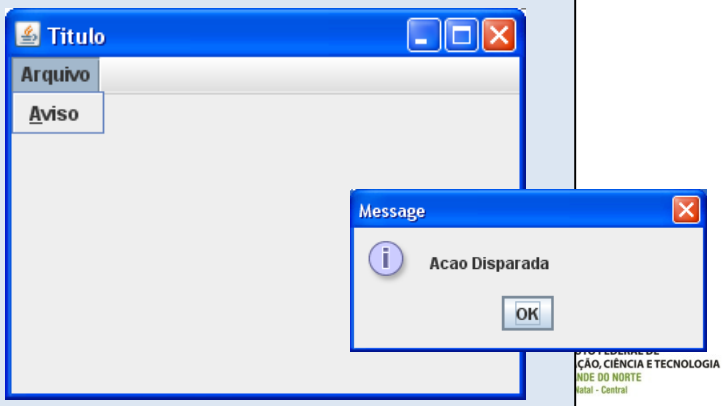
Componente JMenuBar

```
public class Frame extends JFrame {
		JMenuBar menu;
		JMenu menuArquivos;
		JMenuItem menuItemSair;

		public Frame() {
			super("Titulo");

			menuItemSair = new JMenuItem("Sair");

			menuArquivos = new JMenu("Arquivo");
			menuArquivos.add(menuItemSair);

			menu = new JMenuBar();
			menu.add(menuArquivos);

			setJMenuBar(menu);

			setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
			setLayout(null);
			setSize(320, 240);
			setVisible(true);
		}
}
```



Componente JMenuBar

```
public class Frame extends JFrame implements ActionListener{
 JMenuBar menu;
 JMenu menuArquivos;
 JMenuItem menuItemAviso;

 public Frame() {
  super("Titulo");

  menuItemAviso = new JMenuItem("Aviso");
  menuItemAviso.setActionCommand("acaoMenuAviso");
  menuItemAviso.addActionListener(this);
  menuItemAviso.setMnemonic('A');

  menuArquivos = new JMenu("Arquivo");
  menuArquivos.add(menuItemAviso);

  menu = new JMenuBar();
  menu.add(menuArquivos);

  setJMenuBar(menu);
```



Componente JMenuBar – Cont.

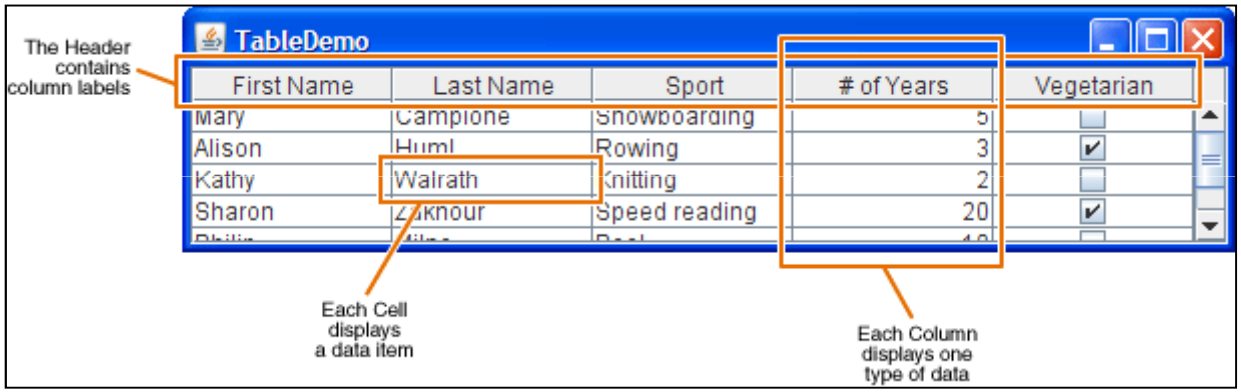
```
public void actionPerformed(ActionEvent e) {
  if("acaoMenuAviso".equals(e.getActionCommand())){
    JOptionPane.showMessageDialog(null, "Acao Disparada");
  }

}
```

JTable

- Corresponde a uma tabela
- Utilizado para visualizar dados
- Componente MVC (*Model, View, Controller*)
  - Model:
    - Controla os dados
  - View:
    - Apresentação
  - Controller:
    - Controla a apresentação dos dados

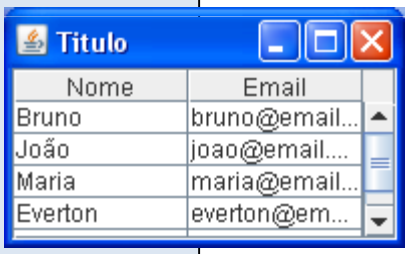
JTable

- Representação:



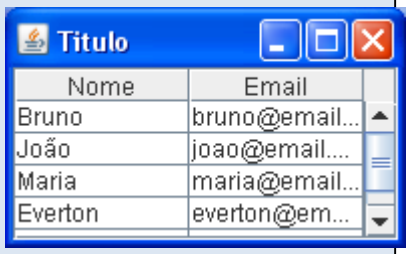
JTable

```
public class Tabela extends JFrame{
  JTable tabela = new JTable();

  public Tabela(){
    super("Titulo");

    String[] tituloColunas = new String []{"Nome", "Email"};
    String[][] dadosTabela = new String [][] {
        {"Bruno","bruno@email.com"},
        {"João","joao@email.com"},
        {"Maria","maria@email.com"},
        {"Everton","everton@email.com"},
        {"Daniel","daniel@email.com"}};

    tabela = new JTable(dadosTabela, tituloColunas);

    JScrollPane scrollPane = new JScrollPane(tabela);
    add(scrollPane, BorderLayout.CENTER);

     setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
     setSize(200, 120);
     setVisible(true);
}
```



Jtable – Utilizando Modelo

```
String[] tituloColunas = new String []{"Nome", "Email"};
String[][] dadosTabela = new String [][] {
{"Bruno","bruno@email.com"},
{"João","joao@email.com"},
{"Maria","maria@email.com"},
{"Everton","everton@email.com"},
{"Daniel","daniel@email.com"}};

DefaultTableModel modelo = new DefaultTableModel(dadosTabela, tituloColunas);
JTable tabela = new JTable(modelo);

JScrollPane scrollPane = new JScrollPane(tabela);
   add(scrollPane, BorderLayout.CENTER);

setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setSize(200, 120);
setVisible(true);
```



JTable

- Alguns métodos:

```
// Obtem o modelo da Tabela
DefaultTableModel modelo = (DefaultTableModel)tabela.getModel();

// Adiciona uma linha
modelo.addRow( new String [] {"Valor 1", "Valor 2"});

// Remove a linha
modelo.removeRow(numLinha);

// Retorna o número da linha selecionada
int linhaSelecionada = tabela.getSelectedRow();

// Total de linhas
int numLinhas = modelo.getRowCount();

//Substitui o valor da linha e coluna definida
modelo.setValueAt(novoValue, linha, coluna);
```

JTable

- Adicionando valores utilizando Vector:

```
Vector vetor = new Vector();
vetor.add("valor 1");
vetor.add("valor 2");

modelo.addRow(vetor);
```

JTable

- Percorrendo Elementos de uma Linha Selecionada na Tabela:

```
int linhaSelecionada = tabela.getSelectedRow();
for(int i=0; i<modelo.getColumnCount();i++){
   System.out.println(modelo.getValueAt(linhaSelecionada, i));
}
```

JDesktopPane e JInternalFrame

- Interface de Múltiplos Documentos (MDI – *Multiple Document Interface*)
- Uma Janela Principal, que contém outras janelas para gerenciar vários documentos abertos que estão sendo processados em paralelo

JDesktopPane e JInternalFrame

```
public class Frame extends JFrame implements ActionListener{
  JMenuBar menu;
  JMenu menuArquivos;
  JMenuItem menuItemJanela1, menuItemJanela2;

  JDesktopPane desktopPane;
  JInternalFrame internalFrame1, internalFrame2;
```

JDesktopPane e JInternalFrame

```
public Frame() {
  super("Titulo");

  menuItemJanela1 = new JMenuItem("Janela 1");
  menuItemJanela1.setActionCommand("janela1");
  menuItemJanela1.addActionListener(this);

  menuItemJanela2 = new JMenuItem("Janela 2");
  menuItemJanela2.setActionCommand("janela2");
  menuItemJanela2.addActionListener(this);

  menuArquivos = new JMenu("Arquivo");
  menuArquivos.add(menuItemJanela1);
  menuArquivos.add(menuItemJanela2);

  menu = new JMenuBar();
  menu.add(menuArquivos);

  setJMenuBar(menu);
```

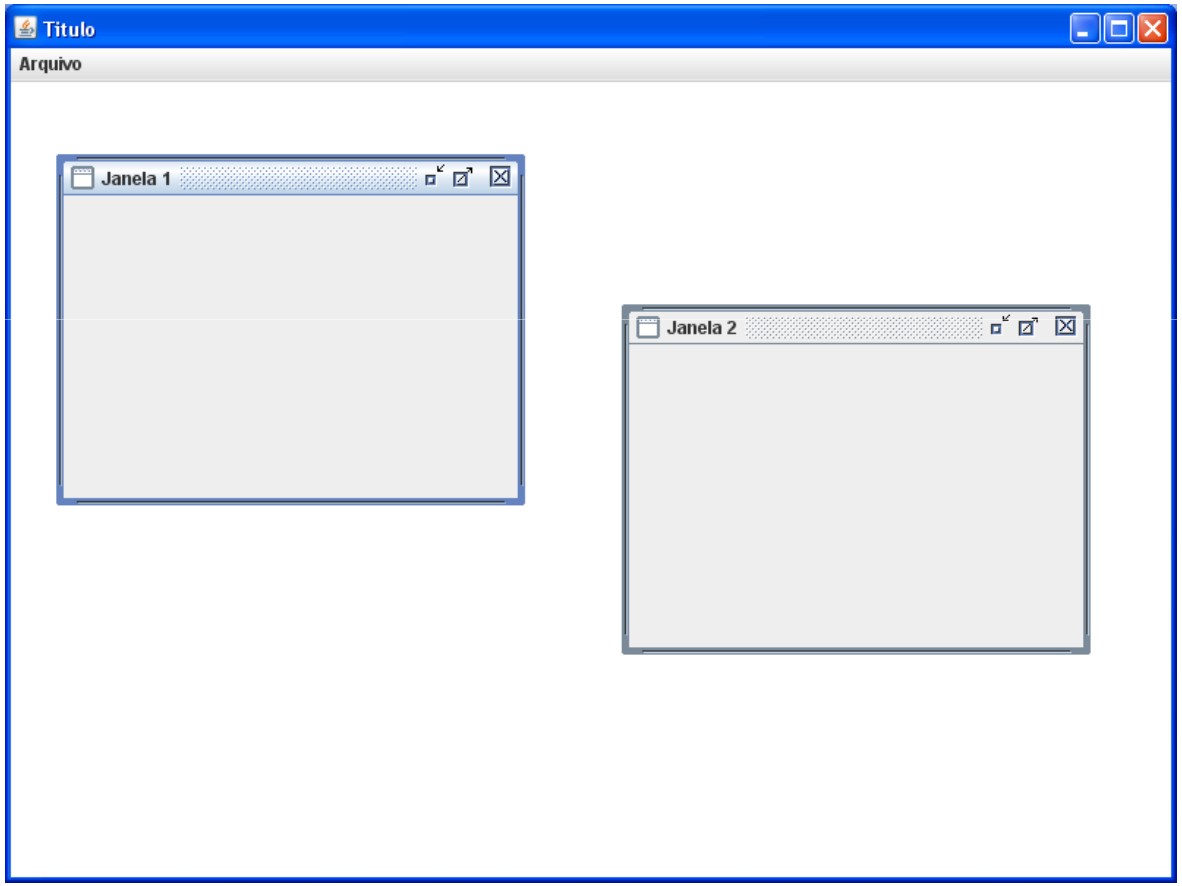
JDesktopPane e JInternalFrame

```
  desktopPane = new JDesktopPane();
  getContentPane().add(desktopPane);

  setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  setSize(800, 600);
  setVisible(true);
}
```

JDesktopPane e JInternalFrame

```
public void actionPerformed(ActionEvent e) {
 if("janela1".equals(e.getActionCommand())){

   internalFrame1 = new JInternalFrame("Janela 1", true, true, true, true);
   internalFrame1.setSize(320,240);
   desktopPane.add(internalFrame1);
   internalFrame1.setVisible(true);
   } else
      if("janela2".equals(e.getActionCommand())){
         internalFrame2 = new JInternalFrame("Janela 2", true, true, true, true);
         internalFrame2.setSize(320,240);
         desktopPane.add(internalFrame2);
         internalFrame2.setVisible(true);
   }
  }
}
```

JDesktopPane e JInternalFrame



JDesktopPane e JInternalFrame

- Outra forma:

```
public class TelaInterna extends JInternalFrame{
   public TelaInterna(){
        super("Tela Interna");
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setIconifiable(true);
        setMaximizable(true);
        setResizable(true);
        setClosable(true);
        setVisible(true);
        setSize(320,240);
    }
}
```

JDesktopPane e JInternalFrame

```
public class TelaPrincipal extends JFrame{
        JDesktopPane desktopPane;
        TelaInterna telaInterna;

        public TelaPrincipal(){
                super("Tela Principal");
                telaInterna = new TelaInterna();
                desktopPane = new JDesktopPane();
                desktopPane.add(telaInterna);
                telaInterna.setVisible(true);
                add(desktopPane);
                setDefaultCloseOperation(EXIT_ON_CLOSE);
                setSize(640, 480);
                setVisible(true);
        }
}
```