

Interfaces Gráficas

Prof. Dr. Rodrigo Plotze

rodrigoplotze@gmail.com

Conteúdo

- *Controles e Componentes em Java (Swing)*
 - *Frames (JFrame), Botões (JButton) e Rótulos (JLabel)*
 - *Campos de Texto (JTextField); CheckBox (JCheckBox) e RadioButton (JRadioButton); Listas (JList), ComboBox (JComboBox) e Tabelas (JTable); Caixas de Diálogo*
- *Eventos: Janela, Foco, Teclado e Mouse*
- *Interfaces de Múltiplos Documentos (MDI – Multiple Document Interface)*

INTERFACES GRÁFICAS

Graphical User Interface

- Graphical User Interface (GUI) é uma interface gráfica do usuário que permite interação com um computador ou outro dispositivo eletrônico por meio de ícones, botões e outros elementos visuais, em vez de depender exclusivamente de comandos de texto.

Graphical User Interface

- GUIs são comuns em sistemas operacionais de computadores, aplicativos de software, dispositivos móveis e muitos outros dispositivos eletrônicos.
- Eles facilitam a interação do usuário com o dispositivo, permitindo que tarefas complexas sejam realizadas com um mínimo de treinamento e conhecimento técnico.

Graphical User Interface

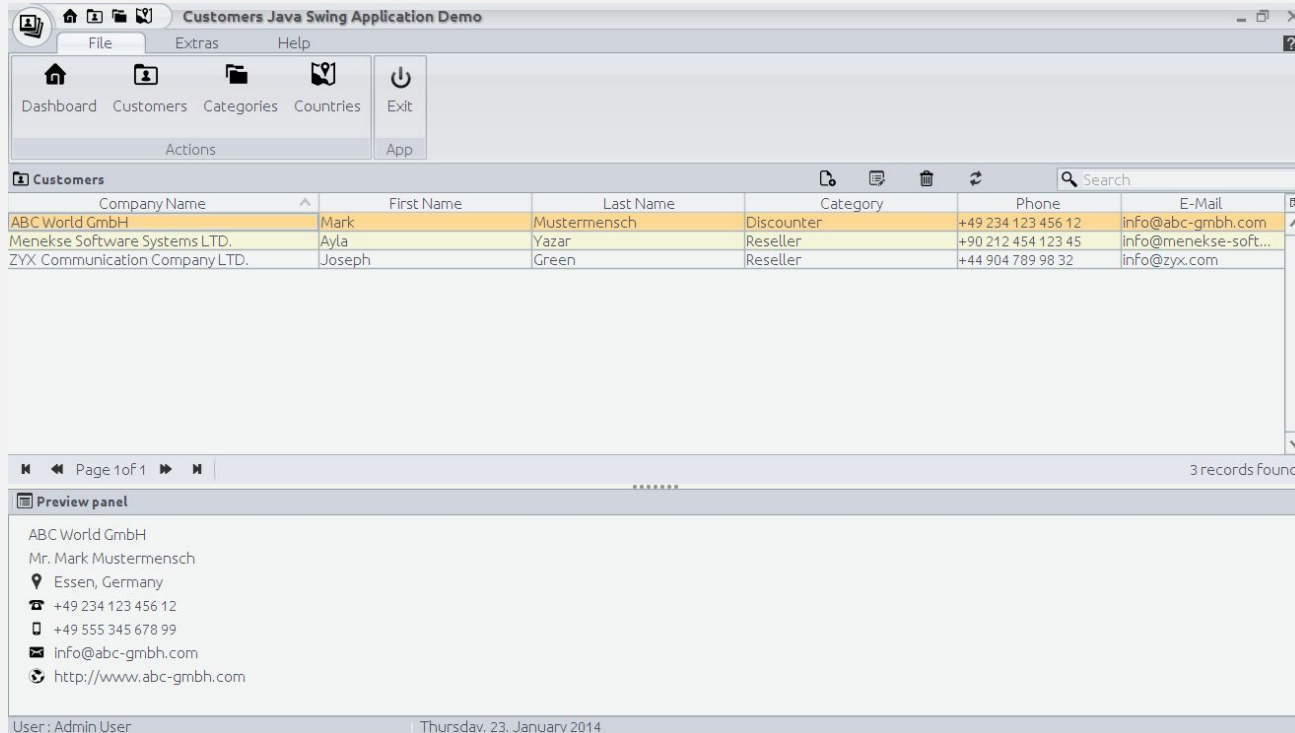
- A maioria das GUIs é projetada para ser intuitiva e fácil de usar, tornando-se uma ferramenta poderosa para muitos usuários.
- Algumas das características comuns de uma GUI incluem janelas, botões, menus suspensos, barras de rolagem, caixas de seleção e campos de entrada de texto.

■ GUI Console

SRT WERKNEMERSPROGRAMMA		Versie 3.00	
Werknemersnummer	: 6587 (Typ 0000 om terug te gaan)		
Achternaam	: Walker		
Voornaam	: OfTheDay		
Adres	: Elmstreet 8		
Postcode + woonplaats	: 666 Hell		
Geboortedatum	: 06-06-0666		
Telefoonnummer	: 0800-DEATH		
Mobiel nummer	: 06-666 666		
Dienstverband	: Vast		
Datum in dienst	: 30-05-2017		
Geslacht (M/V)	: M		
Uren per week (ma/vr)	: 8.00 8.00 8.00 8.00 8.00		
W A A R S C H U W E N B I J C A L A M I T E I T E N			
Huisarts	: Doctor Frankenstein	Tel.nr:	0800-KILL
Waarschuwen igv nood	: Trinity	Tel.nr:	06-1234566
Bij geen gehoor	:	Tel.nr:	
Alles correct ingevoerd ? (J)a / (N)ee / (S)toppen: J			
TOEVOEGEN WERKNEMER		30-05-2017 11:38	

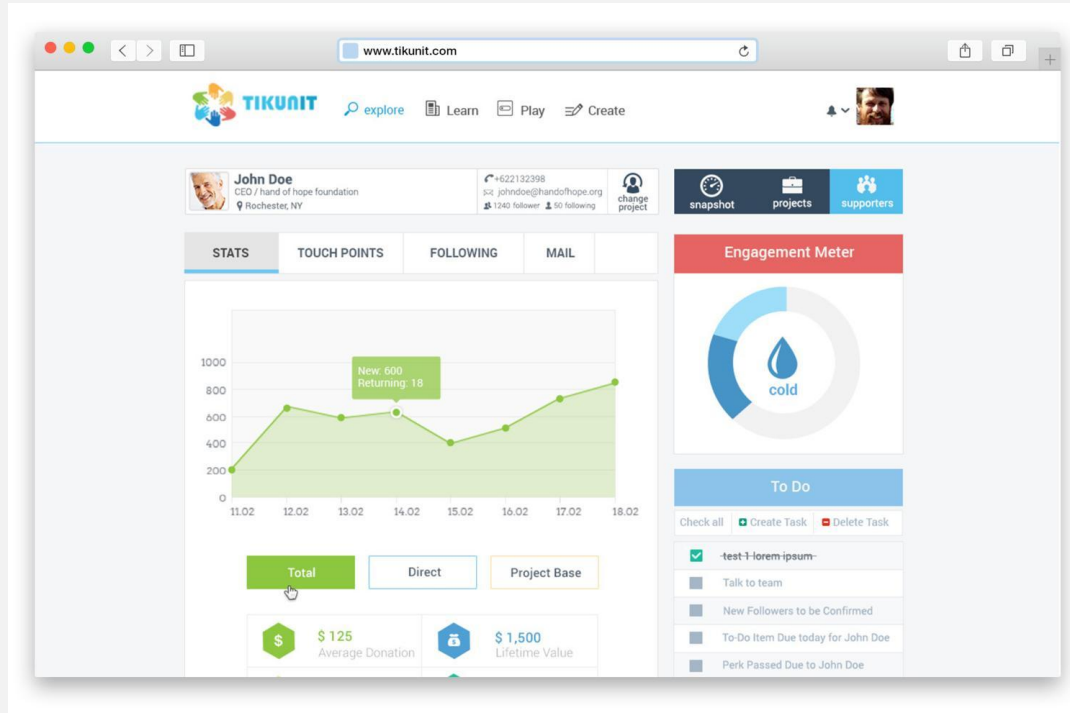
Graphical User Interface

■ GUI Desktop



Graphical User Interface

■ GUI Web



Graphical User Interface

■ GUI Mobile



INTERFACES GRÁFICAS EM JAVA

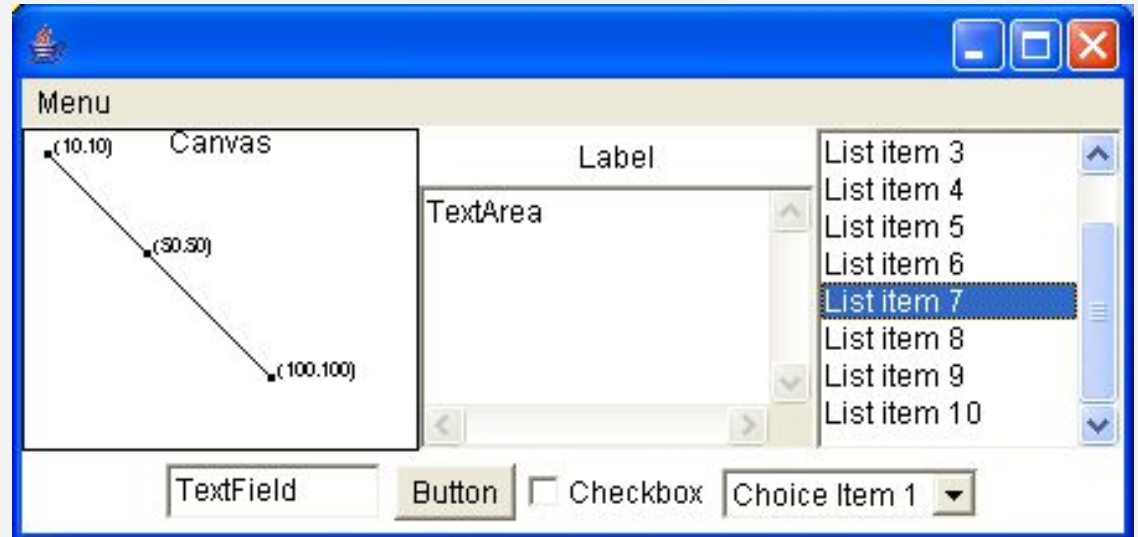
- **Java AWT (Abstract Window Toolkit)** é uma biblioteca gráfica para desenvolvimento de interfaces gráficas do usuário (GUIs) em Java.
- Ela fornece um conjunto de classes e métodos para criar e manipular componentes gráficos, como botões, caixas de texto, rótulos, janelas, menus, barras de rolagem e outros elementos da interface do usuário.

- A biblioteca AWT é parte integrante da plataforma Java desde a sua primeira versão, e é amplamente utilizada em aplicações desktop, applets, e interfaces gráficas em dispositivos embarcados.

- AWT é independente de plataforma, o que significa que os programas escritos com essa biblioteca podem ser executados em diferentes sistemas operacionais sem necessidade de modificação do código-fonte.

Java AWT

- AWT foi substituída pela biblioteca Swing, que oferece recursos mais avançados e uma aparência mais moderna.



- Java Swing é uma biblioteca de interface gráfica do usuário (GUI) em Java que foi introduzida pela primeira vez na versão 1.2 da plataforma Java.
- Ela fornece um conjunto rico de componentes gráficos, como botões, caixas de texto, rótulos, tabelas, árvores, menus, barras de ferramentas, janelas e diálogos.

- Ao contrário da biblioteca AWT, que é uma abstração da interface de usuário nativa do sistema operacional, o ***Swing é uma biblioteca totalmente desenvolvida em Java***, o que significa que os componentes Swing têm aparência e comportamento consistentes em todas as plataformas suportadas pelo Java.

- Swing oferece recursos avançados, como temas, que permitem personalizar a aparência dos componentes gráficos e torná-los visualmente atraentes e modernos.

- Swing é que ele é altamente configurável e extensível.
- Ele fornece uma arquitetura MVC que permite separar a lógica da aplicação da apresentação visual.
- Isso torna mais fácil personalizar a aparência e o comportamento dos componentes Swing para atender às necessidades específicas da aplicação.

- ***Swing é usado para criar aplicativos desktop em Java***, desde aplicativos simples até aplicativos complexos de missão crítica.
- Além disso, outras bibliotecas de interface gráfica, como o JavaFX e o SWT, são baseadas no Swing.
- Portanto, o conhecimento do Swing é fundamental para qualquer desenvolvedor Java que trabalhe com interface gráfica do usuário.

- ***Swing é usado para criar aplicativos desktop em Java***, desde aplicativos simples até aplicativos complexos de missão crítica.
- Além disso, outras bibliotecas de interface gráfica, como o JavaFX e o SWT, são baseadas no Swing.
- Portanto, o conhecimento do Swing é fundamental para qualquer desenvolvedor Java que trabalhe com interface gráfica do usuário.

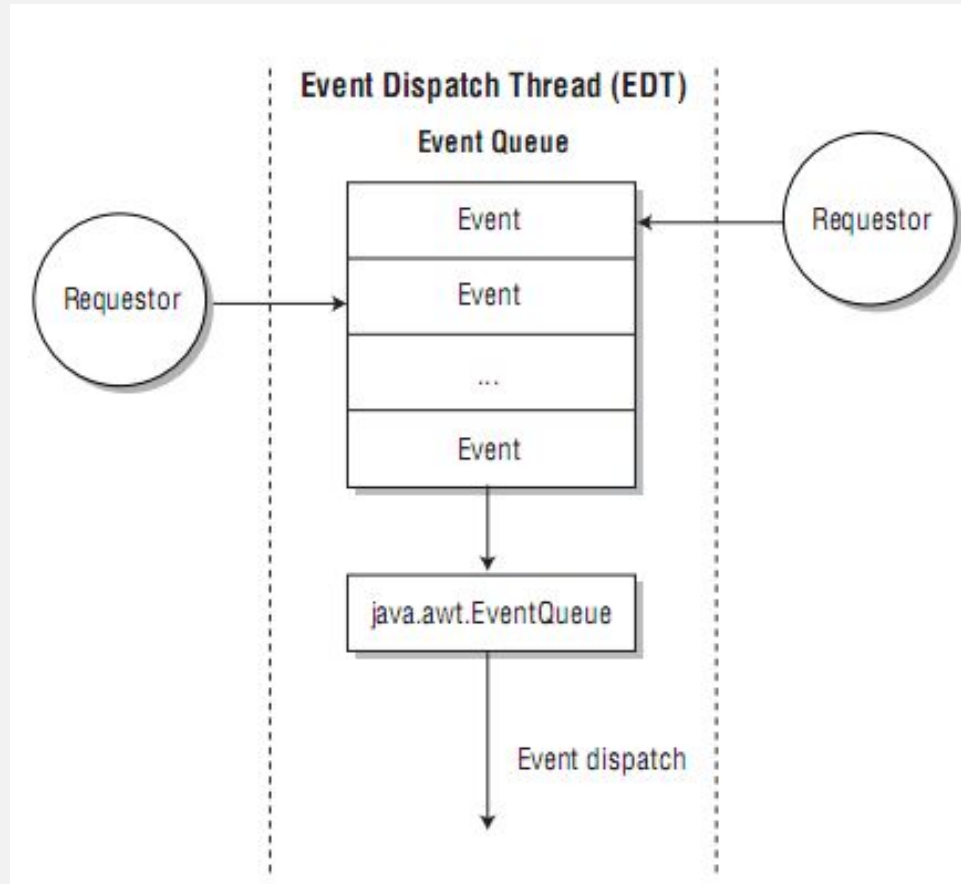
- ***Tratamento de eventos*** é um processo pelo qual a ***aplicação responde a eventos gerados por componentes gráficos***, como cliques de mouse, ações de teclado, seleções de menu e outros eventos de interface do usuário.

- Quando um evento é gerado por um componente Swing, ele é ***colocado na fila de eventos e aguarda para ser processado pelo thread de eventos da interface do usuário.***
- O thread de eventos verifica periodicamente a fila de eventos e processa os eventos conforme necessário.

- Para tratar eventos em Java Swing, é necessário ***implementar um ou mais ouvintes de eventos (event listeners) que monitoram eventos específicos*** gerados pelos componentes Swing.

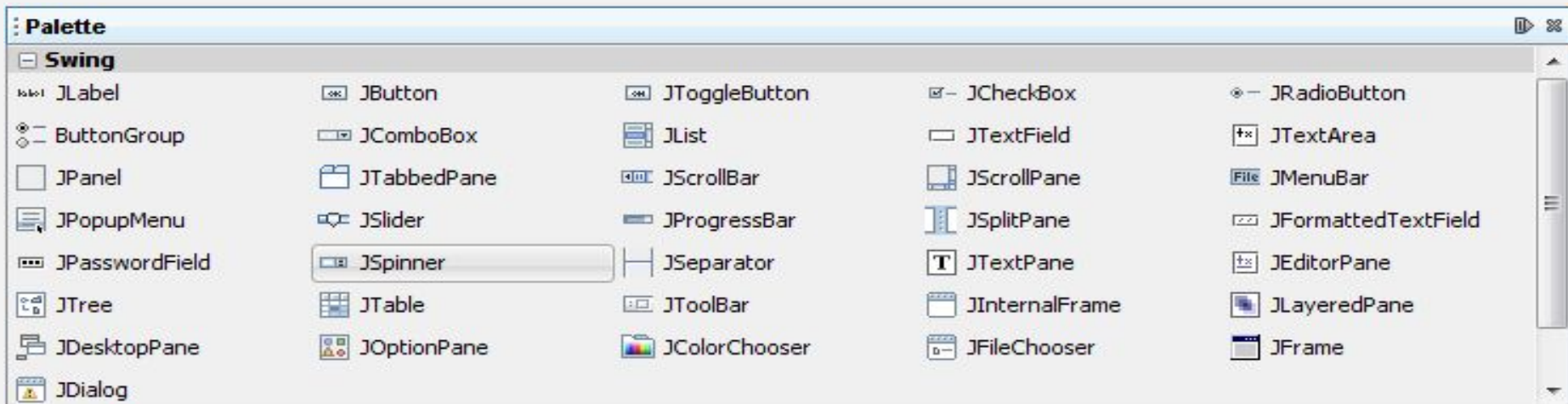
- Um ***ouvinte de eventos*** é uma classe que implementa uma interface de ouvinte de eventos específica, como ***ActionListener***, ***MouseListener*** ou ***KeyListener***.
- Quando um evento ocorre, o ouvinte de eventos correspondente é notificado e executa um código de tratamento de evento específico.

Eventos



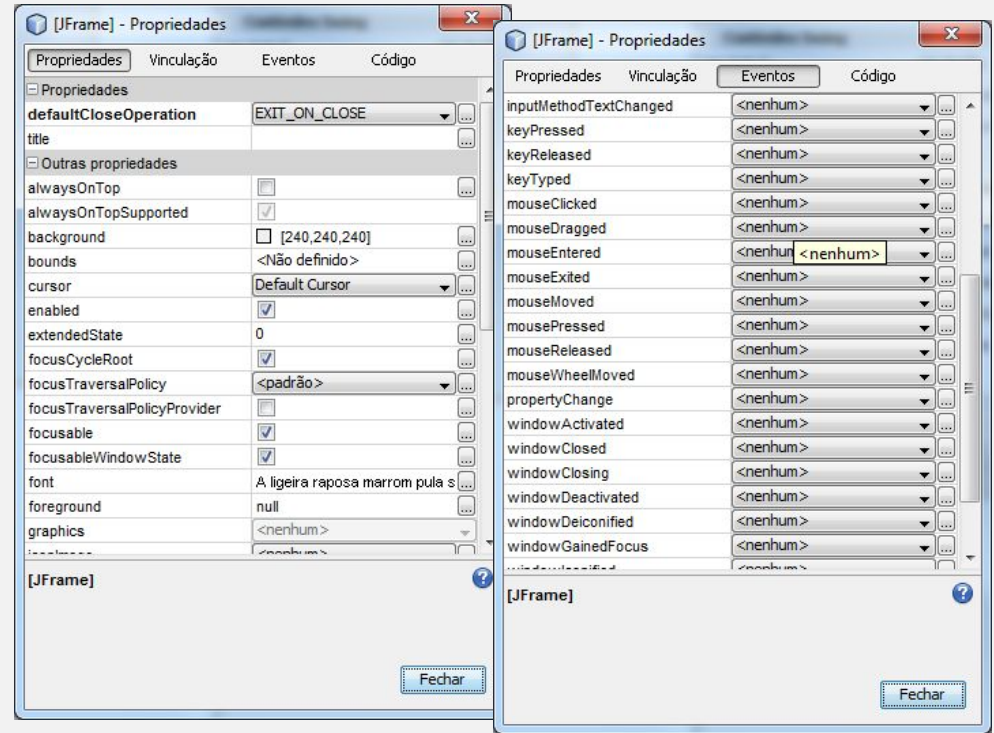
Paleta de Componentes

- Componentes gráficos pré-definidos utilizados na elaboração das interfaces gráficas.



Propriedades

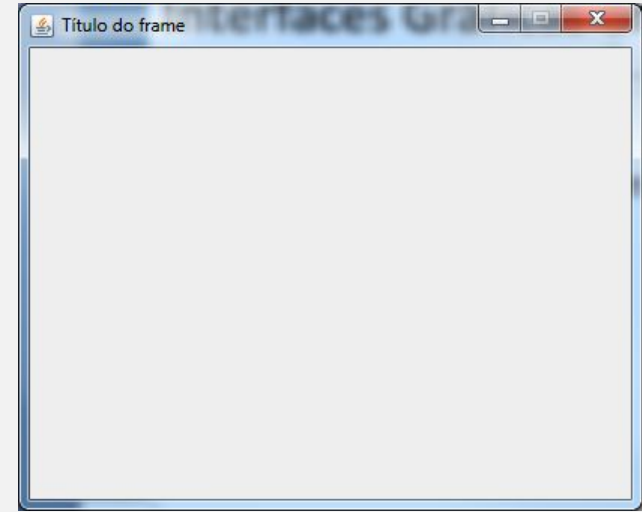
- Recurso que permite a alterações das características e comportamentos dos componentes



CONTROLES

JFrame

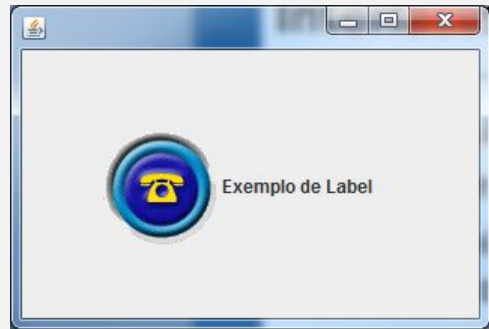
- JFrame ***representa uma janela ou moldura em um aplicativo gráfico*** em Java.
- Ele fornece um recipiente para os componentes Swing e é a base para a maioria das janelas em um aplicativo Java Swing.
- Propriedades: Title e Resizable



Controles para GUI

■ *Classe JLabel*

- Definição de texto que pode ser adicionado a um outro controle.
- Pode exibir: uma única linha de texto, uma imagem ou tanto texto quanto imagem;



Método	Função
JLabel()	Cria um JLabel vazio (sem texto)
JLabel(String)	Cria um JLabel com o texto dado
JLabel(String,int)	Cria um JLabel com o texto e o alinhamento dados
JLabel(String, Image)	Cria um JLabel com o texto e a imagem dados
JLabel(String, Image, int)	Cria um JLabel com o texto, a imagem e o alinhamento dados
getText()	Obtém o texto do JLabel
setText()	Especifica o texto do JLabel

Controles para GUI

■ **Classe *JButton***

- Componentes que o usuário interage por meio de cliques do mouse
- É gerado um ***ActionEvent***
- Propriedades Principais
 - Icon (PressedIcon, RolloverIcon)
 - Mnemonic
 - Text
 - ToolTipText □ dicas
 - ContentAreaFilled, BorderPainted

Controles para GUI

■ *Classe JButton*

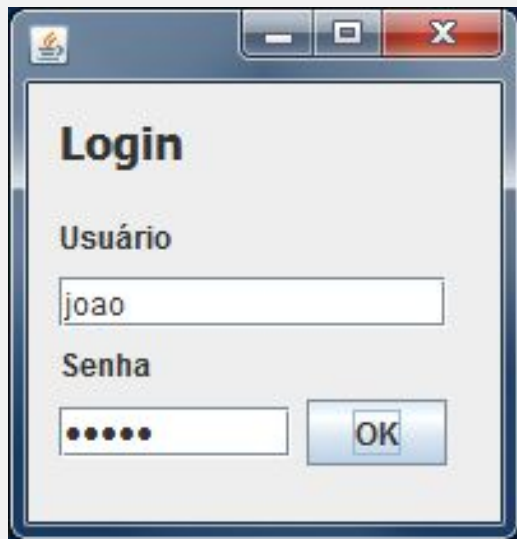
Método	Função
Button()	Cria um botão sem texto
Button(String)	Cria um botão com o texto dado
Button(String, Image)	Cria um botão com o texto e a imagem dados
getText()	Obtém o texto do botão
setText(String)	Especifica o texto do botão
setEnabled (boolean)	Define se o botão está habilitado (true) ou desabilitado (false)
setHorizontalTextPosition()	Define o tipo de alinhamento horizontal do texto em relação a uma imagem. Pode assumir LEFT (esquerda) ou RIGHT (direita)
setMnemonic(char)	Define uma letra que será usada como acionadora do evento clique, em conjunto com a tecla ALT
setToolTipText(String)	Possibilita atrelar uma mensagem ao botão. Quando o ponteiro do mouse estaciona sobre o botão, a mensagem é apresentada
setVerticalTextPosition()	Define o tipo de alinhamento vertical do texto em relação a uma imagem. Pode assumir TOP (topo) ou BOTTOM (abaixo)

Controles para GUI

■ *Classe JTextField*

- Permite criar uma caixa de texto gráfica para entrada de dados do usuário.

Método	Função
<code>JTextField()</code>	Cria uma caixa de texto vazia
<code>JTextField(String)</code>	Cria uma caixa de texto com a string dada
<code>JTextField(String,int)</code>	Cria uma caixa de texto com a string e a quantidade de colunas especificada
<code>JTextField(int)</code>	Cria uma caixa de texto com a quantidade de colunas especificada
<code>getText()</code>	Obtém o texto do objeto
<code>getSelectedText()</code>	Obtém o texto selecionado no objeto
<code>isEditable()</code>	Verifica se o componente é editável ou não
<code>selectAll()</code>	Seleciona todo o texto
<code>setEditable(boolean)</code>	Especifica se o componente é editável ou não
<code>setText()</code>	Especifica o texto contido no componente



Controles para GUI

■ *Classe JPasswordField*

- Variação do JTextField para entrada de senhas.
- Não permite a visualização do que foi digitado.



A screenshot of a login form titled "Login" with a user icon. It contains two input fields: "Usuário" with the text "joao" and "Senha" with masked characters "*****". Below the fields are two buttons labeled "entrar" and "sair".

Método	Função
getPassword()	Obtém o texto do objeto, porém retornando um array do tipo char. Cada caractere é armazenado num elemento do array
getEchoChar()	Obtém o caractere usado na substituição dos caracteres digitados
setEchoChar()	Define o caractere a ser usado em substituição aos caracteres digitados

ATIVIDADE PRÁTICA

Atividade Prática

■ **Exercício 1**

- Construir uma interfaces gráfica para Cadastro de Livros

Window Title

Cadastro de Livros

Título

Autor

Editora

gravar

limpar

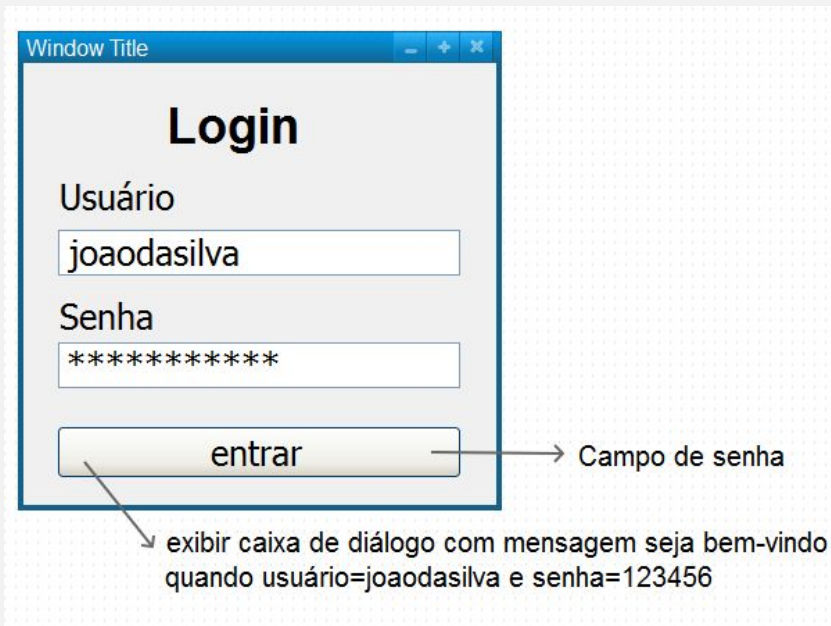
limpar o conteúdo dos campos de texto

exibir em uma caixa de diálogo o conteúdo informado nos campos

Atividade Prática

■ **Exercício 2**

- Desenvolver uma tela de Login que verifica o usuário e senha digitados



Atividade Prática

■ **Exercício 3**

- Elaborar uma GUI para registro de notas fiscais.

The image shows a screenshot of a graphical user interface (GUI) window titled "Registro de Notas Fiscais". The window has a blue title bar with standard Windows window controls (minimize, maximize, close). The main content area is light gray and contains the following elements:

- Valor R\$**: A text label above a text input field containing "890.00".
- OK**: A button next to the "Valor R\$" input field.
- Informações sobre as Notas**: A section header.
- Qtde. de Notas**: A text label next to an empty text input field.
- Maior valor R\$**: A text label next to an empty text input field.
- Menor valor R\$**: A text label next to an empty text input field.
- Soma total R\$**: A text label next to an empty text input field.
- exibir**: A wide button at the bottom of the form.

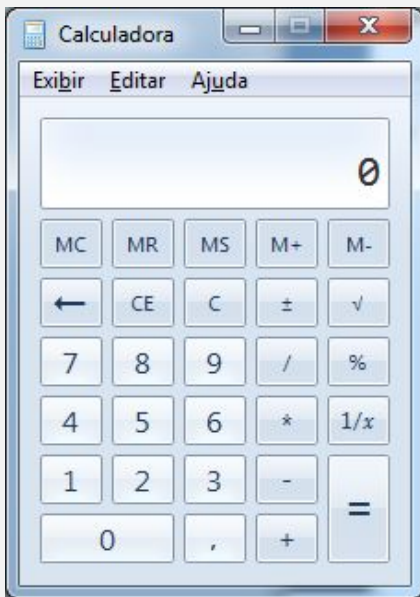
Annotations with arrows point to specific elements:

- An arrow points from the "OK" button to the text "Armazenar um número indeterminado de notas".
- An arrow points from the "Qtde. de Notas" input field to the text "somente leitura".
- An arrow points from the "Maior valor R\$" input field to the text "somente leitura".
- An arrow points from the "Menor valor R\$" input field to the text "somente leitura".
- An arrow points from the "Soma total R\$" input field to the text "somente leitura".

Atividade Prática

■ **Exercício 4**

- Construir uma Calculadora inspirada na *Calculadora do Windows*.



Atividade Prática

■ **Exercício 5**

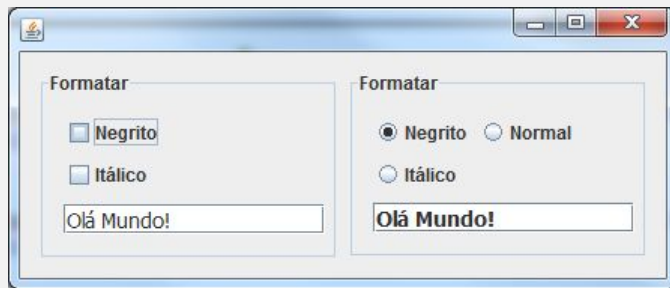
- Faça uma pesquisa sobre a classe ***JFormattedTextField***, a qual permite a definição de campos de texto com máscaras. Em seguida elabore uma aplicação baseada na interface:



Controles para GUI

■ ***Classe JCheckBox (Caixa de Opção)***

- Permitem a seleção múltipla (ou não) de um conjunto de itens



Método	Função
JCheckBox(String)	Cria um checkbox com o texto especificado e com a opção não selecionada
JCheckBox(String, boolean)	Cria um checkbox com o texto e estado especificados com a seleção definida em boolean (true ou false)
getStateChange()	Obtém o estado do checkbox, retornando verdadeiro (true) ou falso (false)
setSelected(boolean)	Especifica o estado do checkbox: true marca a caixa, false desmarca

Controles para GUI

■ ***Classe JRadioButton (Botões de Rádio)***

- Seleção múltipla (ou não) de um grupo de opções

Método	Função
JRadioButton(String)	Cria um botão de rádio com o texto especificado
JRadioButton(String, boolean)	Cria um botão de rádio com o texto especificado e com a seleção definida (true ou false)
setSelected(boolean)	Define se o botão está ou não selecionado (true ou false)
ButtonGroup()	Cria um grupo para botões de rádio
<nome do grupo>.add()	Adiciona cada botão de rádio a um determinado grupo

Controles para GUI

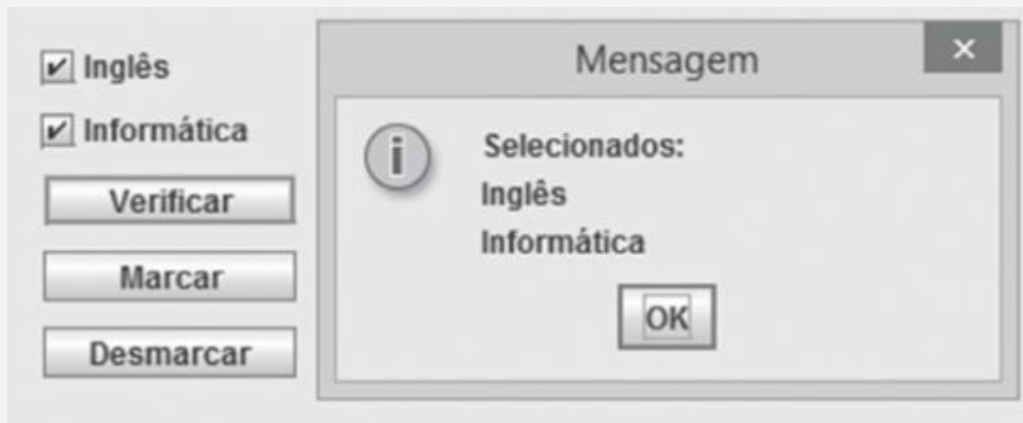
```
private void ckNegritoActionPerformed(java.awt.event.ActionEvent evt) {  
    neg = (neg == Font.PLAIN) ? Font.BOLD : Font.PLAIN;  
    txtCheckBox.setFont(new Font("Tahoma", neg + ita, 14));  
}  
  
private void ckItalicoActionPerformed(java.awt.event.ActionEvent evt) {  
    ita = (ita == Font.PLAIN) ? Font.ITALIC : Font.PLAIN;  
    txtCheckBox.setFont(new Font("Tahoma", neg + ita, 14));  
}  
  
private void rbNegritoActionPerformed(java.awt.event.ActionEvent evt) {  
    txtRadioButton.setFont( new Font("Tahoma", Font.BOLD, 14));  
}  
  
private void nbItalicoActionPerformed(java.awt.event.ActionEvent evt) {  
    txtRadioButton.setFont( new Font("Tahoma", Font.ITALIC, 14));  
}  
  
private void rbNormalActionPerformed(java.awt.event.ActionEvent evt) {  
    txtRadioButton.setFont( new Font("Tahoma", Font.PLAIN, 14));  
}
```

ATIVIDADE PRÁTICA

Atividade Prática

■ **Exercício 1**

- Elabore uma aplicação baseada na seguinte interface.



Atividade Prática

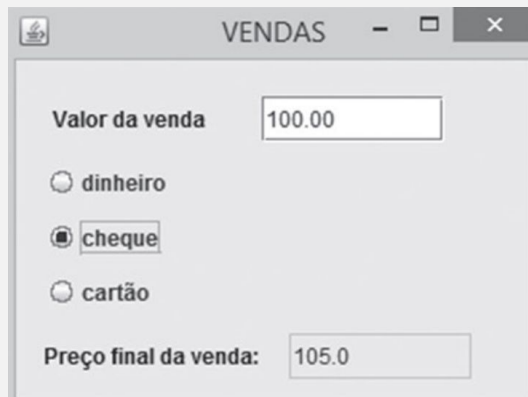
▪ **Exercício 2**

- Elabore uma aplicação que:
 - Permita o usuário selecionar um país do conjunto {Brasil, Argentina, México, Equador, Estados Unidos, Canadá, entre outros}.
 - A partir da seleção do usuário exibir a imagem da bandeira.

Atividade Prática

■ *Exercício 3*

- Crie uma classe que simule vendas contendo três formas de pagamento, de acordo com a Figura.
- O usuário entra com um valor, escolhe a forma de pagamento, e o cálculo do preço final é realizado conforme os seguintes critérios:
 - dinheiro, desconto de 5%;
 - cheque, acréscimo de 5%;
 - cartão, acréscimo de 10%.



The screenshot shows a Java Swing window titled "VENDAS". Inside the window, there is a text field labeled "Valor da venda" containing the value "100.00". Below this, there are three radio buttons for payment methods: "dinheiro", "cheque", and "cartão". The "cheque" radio button is selected. At the bottom, there is a text field labeled "Preço final da venda:" containing the value "105.0".

Controles para GUI

■ ***Classe JList (Lista de Seleção)***

- Possibilitam a escolha de um ou vários valores armazenados em uma lista de opções.

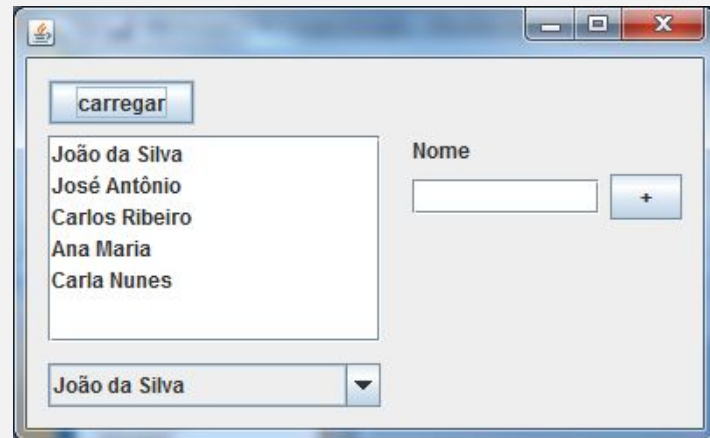
Método	Função
JList()	Cria uma caixa de seleção
getSelectedvalue()	Obtém o texto do item selecionado
getSelectedIndex()	Obtém o índice do item selecionado
setSelectedIndex(int)	Seleciona o índice especificado
setSelectedInterval(int, int)	Seleciona diversos índices dentro do intervalo especificado
isSelectionEmpty()	Verifica se existe algum item selecionado na lista, retornando verdadeiro ou falso
isSelectedIndex(int)	Verifica se o índice especificado está selecionado, retornando verdadeiro ou falso

Controles para GUI

■ *Classe JComboBox*

- Uso semelhante ao JList
- Permite a seleção de itens do conjunto

Método	Função
JComboBox(String)	Cria uma caixa de seleção JComboBox com um array do tipo string
addItem(String)	Adiciona o texto como um novo item
getSelectedItem()	Obtém o texto do item selecionado
getItemCount()	Obtém a quantidade total de itens
getSelectedIndex()	Obtém o índice do item selecionado
removeItemAt(int)	Remove o item com o índice especificado
removeAllItems()	Remove todos os itens da lista



Controles para GUI

■ *Classe JTextArea*

- Permite manipular diversas linhas de texto.

Método	Função
JTextArea()	Cria uma área de texto
JTextArea(int,int)	Cria uma área de texto de acordo com o número de linhas e colunas especificadas
JTextArea(String)	Cria uma área de texto de acordo com o texto especificado
JTextArea(String,int, int)	Cria uma área de texto de acordo com o texto, o número de linhas e o número de colunas especificadas
getColumns()	Obtém o comprimento ou colunas da área de texto em caracteres
getRows()	Obtém a largura ou linhas da área de texto em caracteres
getSelectedText()	Obtém o texto selecionado na área de texto
setColumns()	Define o comprimento ou colunas da área de texto
setRows()	Define a largura ou linhas da área de texto
insert(String, int)	Insere a string especificada na posição indicada por uma variável inteira
replaceRange(String, int, int)	Substitui o texto fornecido na variável string pelo texto contido entre as posições definidas (início e fim)
setText(), getText(), setEditable()	Métodos da classe JTextComponent que funcionam da mesma forma que em JTextField

ATIVIDADE PRÁTICA

Atividade Prática

■ **Exercício 1**

- Elabore uma aplicação baseada na seguinte interface.

The image shows a Java Swing window titled "Valor". It contains the following elements:

- A text input field at the top containing the value "100".
- A list box below it containing three items: "10%", "20%", and "30%". The "30%" item is currently selected and highlighted.
- A button labeled "Calcular" (Calculate) positioned below the list box.
- A text output field at the bottom containing the result "70.0".

Atividade Prática

■ **Exercício 2**

- Construir uma aplicação para demonstrar o uso do JComboBox.



The screenshot shows a Java Swing window titled "Window Title". Inside the window, there is a label "Escolha uma fruta" above a JComboBox component. The JComboBox has a text field containing "Combo box" and a downward arrow. Below the JComboBox, there is a label "Fruta selecionada". Under this label, there are two text boxes: one labeled "Nome" and another labeled "Posição". At the bottom of the window, there is a button labeled "restaurar".

Selecione a fruta de índice 0

Atividade Prática

■ **Exercício 3**

- Elaborar uma GUI para Cadastro de Clientes

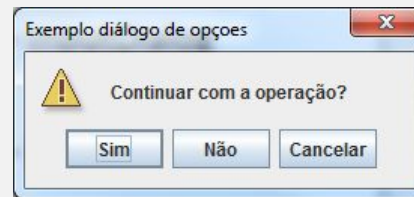
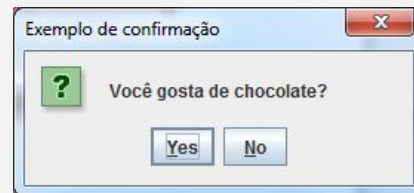
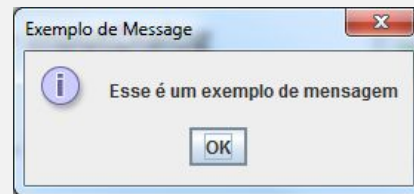
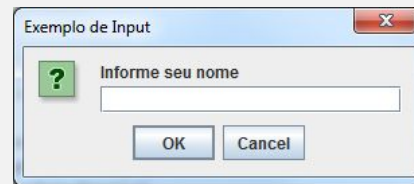
The screenshot shows a Windows-style application window titled "Window Title" with a standard minimize, maximize, and close button set. The main content area is titled "Cadastro" in bold. It contains the following elements:

- Nome:** A single-line text input field.
- Endereço:** A single-line text input field.
- Cidade:** A single-line text input field.
- Estado:** A dropdown menu with a downward arrow.
- Sexo:** Two radio buttons labeled "Masculino" and "Feminino".
- Período:** Three checkboxes labeled "Manhã", "Tarde", and "Noite".
- Cursos de Interesse:** A section containing two list boxes, each with "Item1", "Item2", and "Item3". Between the list boxes are four buttons: ">", ">>", "<", and "<<".
- Buttons:** At the bottom of the window are two buttons labeled "novo" and "salvar".

Controles para GUI

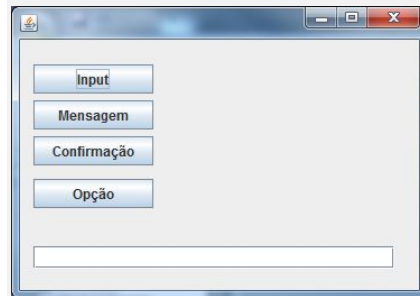
■ *Caixas de Diálogo*

- InputDialog
- MessageDialog
- ConfirmDialog
- OptionDialog



Controles para GUI

```
private void btnInputActionPerformed(java.awt.event.ActionEvent evt) {  
  
    String res = JOptionPane.showInputDialog(null,  
        "Informe seu nome", "Exemplo de Input",  
        JOptionPane.QUESTION_MESSAGE);  
  
    if (res != null) {  
        txtResultado.setText("Seu nome é " + res);  
    } else {  
        txtResultado.setText("O nome não foi informado");  
    }  
}  
  
private void btnMensagemActionPerformed(java.awt.event.ActionEvent evt) {  
    JOptionPane.showMessageDialog(null, "Esse é um exemplo de mensagem",  
        "Exemplo de Message",  
        JOptionPane.ERROR_MESSAGE);  
}
```

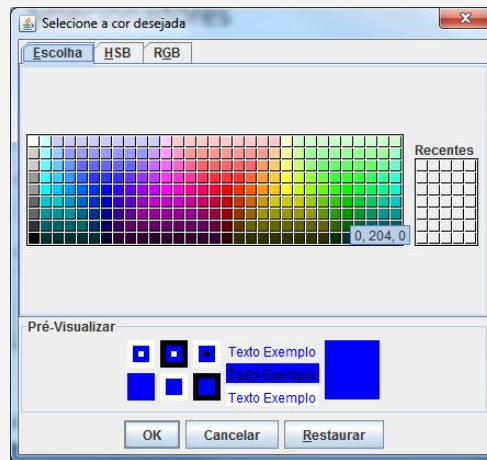


Controles para GUI

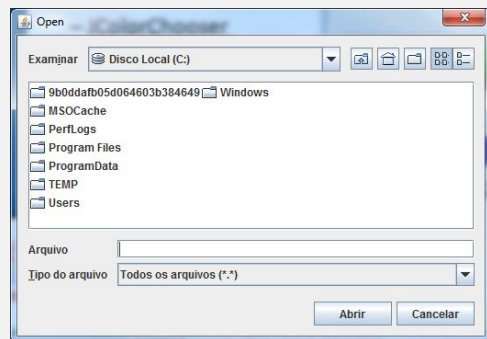
```
private void btnConfirmacaoActionPerformed(java.awt.event.ActionEvent evt) {  
    int res = JOptionPane.showConfirmDialog(null, "Você gosta de chocolate?",  
        "Exemplo de confirmação", JOptionPane.YES_NO_OPTION);  
  
    if (res == JOptionPane.YES_OPTION) {  
        txtResultado.setText("Sim, você gosta de chocolate.");  
    } else if (res == JOptionPane.NO_OPTION) {  
        txtResultado.setText("Não, você não gosta de chocolate.");  
    }  
}  
  
private void btnOpcaoActionPerformed(java.awt.event.ActionEvent evt) {  
    String[] opcoes = new String[]{"Sim", "Não", "Cancelar"};  
    int res = JOptionPane.showOptionDialog(null,  
        "Continuar com a operação?",  
        "Exemplo diálogo de opções",  
        JOptionPane.YES_NO_CANCEL_OPTION,  
        JOptionPane.WARNING_MESSAGE,  
        null, opcoes, opcoes[0]);  
  
    switch(res){  
        case 0: txtResultado.setText("Sim"); break;  
        case 1: txtResultado.setText("Não"); break;  
        default:  
            txtResultado.setText("Cancelar"); break;  
    }  
}
```

Controles para GUI

- Seleccionadores
 - Seleção de Cor (*JColorChooser*)



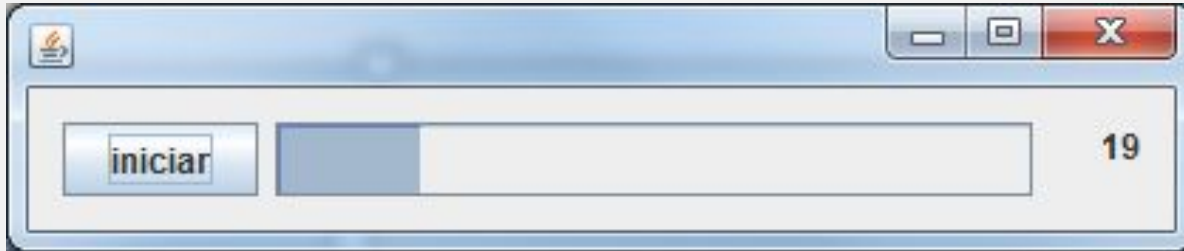
- Seleção de Arquivo (*JFileChooser*)



Controles para GUI

- ***Classe JProgressBar***

- Barra de Progresso



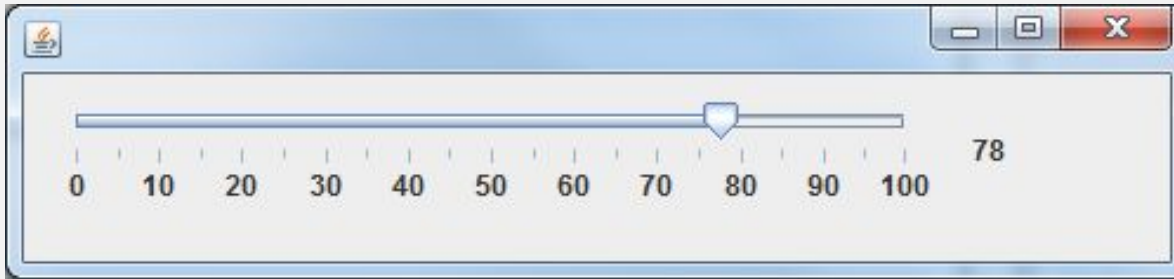
```
private void btnIniciarActionPerformed(java.awt.event.ActionEvent evt) {  
    BarraProgressoTemporizador t = new BarraProgressoTemporizador();  
    t.BARRA = barraProgresso;  
    t.VALOR = lblValor;  
    t.start();  
}
```

```
public class BarraProgressoTemporizador extends Thread {  
  
    public JProgressBar BARRA=null;  
    public JLabel VALOR=null;  
    private int POSICAO=0;  
  
    public BarraProgressoTemporizador(){  
        POSICAO = 0;  
    }  
  
    public void run() {  
        while(true && POSICAO < 100) {  
            try {  
                BARRA.setValue(POSICAO);  
                BARRA.getUI().update(BARRA.getGraphics(),BARRA);  
                sleep(50);  
                POSICAO++;  
                VALOR.setText(String.valueOf(POSICAO));  
                VALOR.getUI().update(VALOR.getGraphics(),VALOR);  
            } catch (InterruptedException ex) {  
                ex.printStackTrace();  
            }  
        }  
    }  
}
```

Controles para GUI

■ *Classe JSlider*

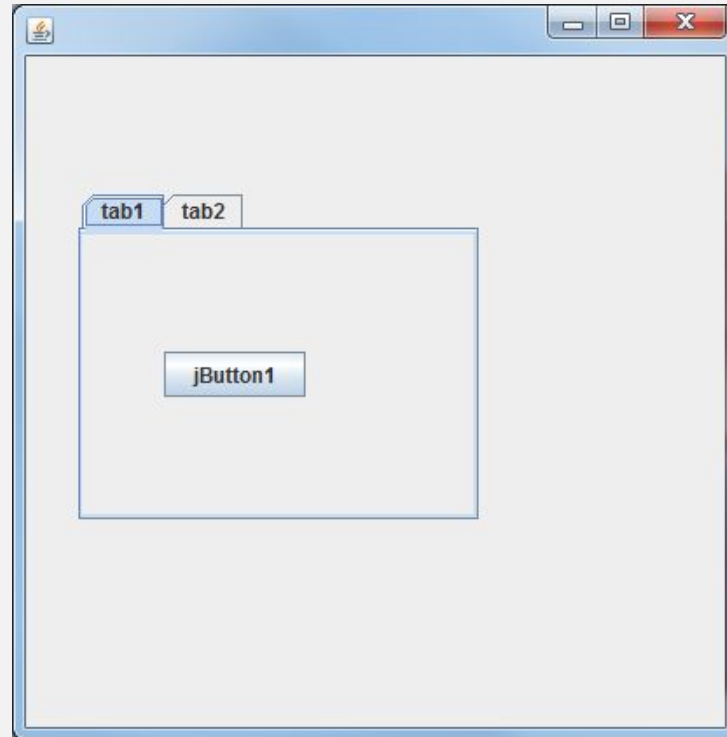
- Propriedades
 - Maximum, Minimum, Ticks



Controles para GUI

▪ ***Classe JTabbedPane (Painéis Tabulados)***

- Propriedades
 - tabPlacement

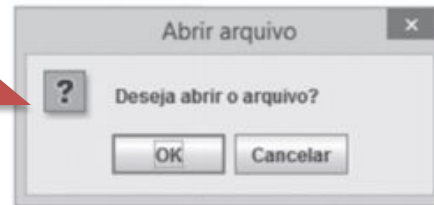
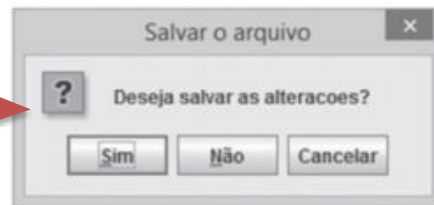
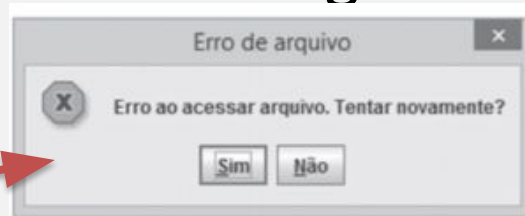
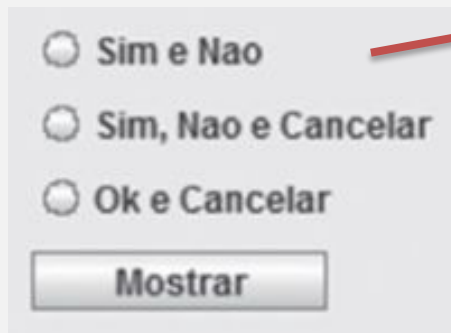


ATIVIDADE PRÁTICA

Atividade Prática

■ **Exercício 1**

- Elabore uma aplicação baseada na seguinte interface.



Atividade Prática

▪ **Exercício 2**

- Elabore uma aplicação baseada na seguinte interface.



Controles para GUI

■ *Classe JTable*

- Permite apresentar dados na forma de tabelas em duas dimensões: linhas e colunas.

Método	Função
JTable(int, int)	Cria uma tabela contendo um determinado número de linhas e colunas fornecidas com células vazias e usando o DefaultTableModel
JTable(TableModel)	Cria uma tabela a partir do TableModel fornecido
getColumnCount()	Retorna um número inteiro referente ao número de colunas existentes
getColumnName(int)	Retorna uma string referente ao nome da coluna cujo número foi fornecido
getRowCount()	Retorna um número inteiro referente ao número de linhas existentes
getSelectedColumns()	Retorna um array de inteiros contendo os índices de todas as colunas selecionadas
getSelectedRows()	Retorna um array de inteiros contendo os índices de todas as linhas selecionadas
getValueAt(int, int)	Retorna um objeto com o conteúdo da célula cujas linha e coluna foram fornecidas
selectAll()	Seleciona todas as linhas, colunas e células da tabela
setValueAt(Object, int, int)	Define um valor presente em Object para a célula cujas linha e coluna são fornecidas

Controles para GUI

■ *Classe JTable*



ATIVIDADE PRÁTICA

■ **Exercício 1**

- Desenvolver uma aplicação para Gerenciar as Notas de um Aluno.
- Requisitos Funcionais: Controlar as notas das disciplinas considerando as seguintes informações: etapa, nome da disciplina, nota da prova parcial, nota da prova final e média final.

Etapa	Disciplina	Nota Parcial	Nota Final	Média
3	Lab. Prog III	10,00	8,00	8,80
3	Estrutura de Dados	5,00	10,00	8,00

■ ***Exercício 1 (continuação)***

— Requisitos Funcionais

- Utilizar os controles: JTable , JComboBox ou JList

Atividade Prática

■ **Exercício 2**

- Desenvolver uma aplicação Java baseada na seguinte interface gráfica:

The screenshot shows a Java application window titled "Merceria do João". Inside the window, there is a section titled "Venda de Produtos". At the top right of this section is a "sair" button. Below the title, there are three input fields: "Qtde" (with a text box), "Produtos" (with a dropdown menu showing "Arroz"), and "Vl.Unit." (with a text box). To the right of these fields are two buttons: "adicionar" and "limpar". Below these inputs is a table with the following data:

Qtde.	Produto	Vl.Unit.	Vl.Total
2	Arroz	9,20	18,40
3	Macarrão	2,45	7,35
2	Farinha de Trigo	4,60	9,20

Below the table is a large empty rectangular area. At the bottom left of the window is a "sobre" button. At the bottom right, there is a label "Total (R\$)" followed by a text box containing the value "34,95".

Atividade Prática

■ ***Exercício 2 (continuação)***

– Requisitos Funcionais

- *Sair*: Finalizar a aplicação, mas antes pergunta ao usuário: “Tem certeza que deseja sair?”
- *Sobre*: Exibir um novo formulário JFrame com informações sobre o desenvolvedor: nome e código.

INTERFACES DE MÚLTIPLOS DOCUMENTOS - MDI

Interfaces de Múltiplos Documentos

■ *Classe JInternalFrame*

- Criação de frames internos
- São mais “leves” que os JFrame e possuem muitas características em comum.
- Necessário o uso de uma Área de Trabalho (*JDesktopPane*)

setClosable(boolean)	Define se o frame interno pode ser fechado pelo usuário
setIconifiable(boolean)	Define se o frame interno pode ser minimizado pelo usuário
setMaximizable(boolean)	Define se o frame interno pode ser maximizado pelo usuário
setResizable(boolean)	Define se o frame interno pode ser redimensionado
show()	Torna o frame interno visível, o mesmo que “setVisible(true)”

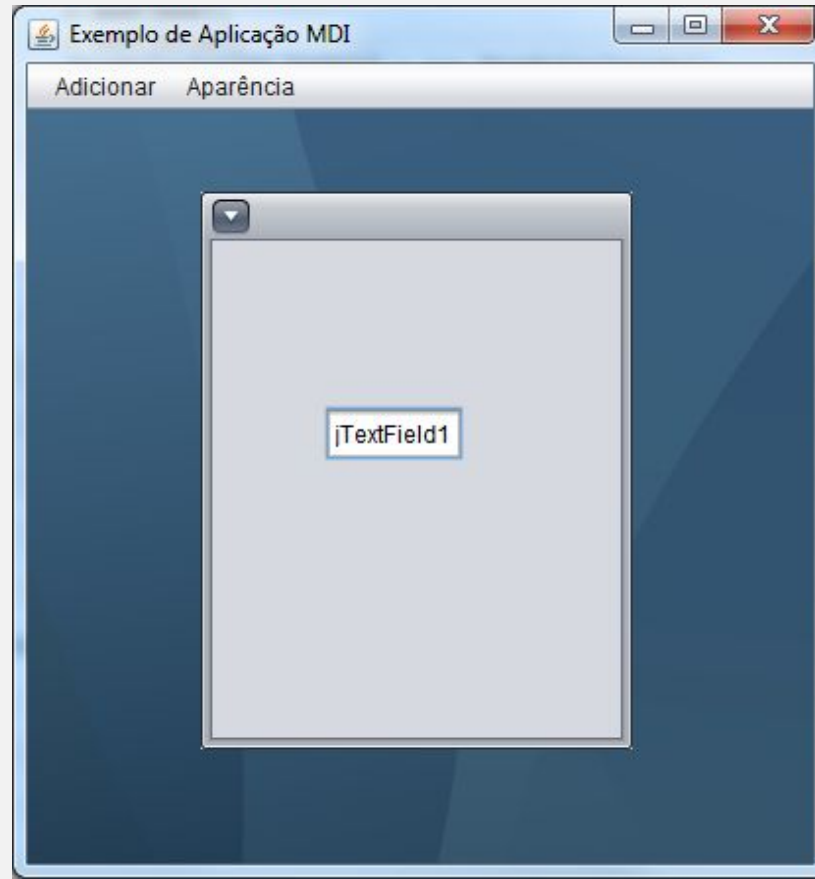
Interfaces de Múltiplos Documentos

■ *Classe JInternalFrame*

– Criação do frame

```
private void btnNovoActionPerformed(java.awt.event.ActionEvent evt) {  
    AppDesktopInterno f = new AppDesktopInterno();  
    AreaTrabalho.add(f);  
    f.setVisible(true);  
}
```

Interfaces de Múltiplos Documentos



ATIVIDADE PRÁTICA

Atividade Prática

■ **Exercício 1**

- Desenvolver uma aplicação MDI capaz de trocar dados entre janelas.
- Para isso, utilize dois formulários internos denominados ***FormCadastro*** e ***FormVisualizar***.
- A aplicação deverá conter um menu para acesso as janelas.

Atividade Prática

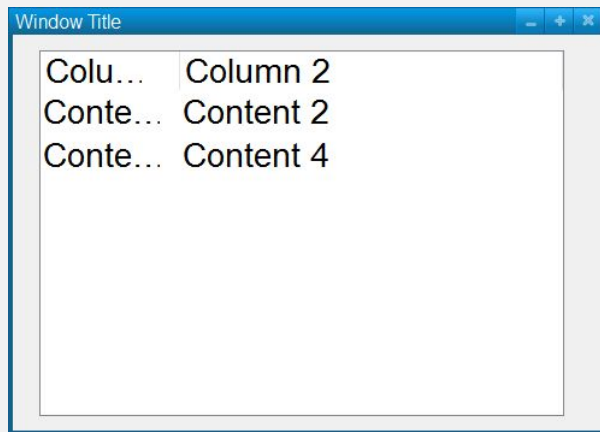
■ **Exercício 1 (continuação)**

– Para isso, considere os seguintes formulários internos:



A screenshot of a Windows application window titled "Window Title". The window contains a form with two text input fields. The first field is labeled "Nome" and the second is labeled "Idade". Both fields have the placeholder text "Text". Below the input fields is a button labeled "salvar".

FormCadastro



A screenshot of a Windows application window titled "Window Title". The window contains a table with two columns and three rows of data. The first column is labeled "Colu..." and the second column is labeled "Column 2". The rows contain "Content 2" and "Content 4".

Colu...	Column 2
Conte...	Content 2
Conte...	Content 4

FormVisualizar

CONTROLES DE EVENTOS

Controles de Eventos

Evento	Descrição
ActionListener	Eventos de ação como o clique do mouse sobre um botão, conforme demonstrado até aqui
AdjustmentListener	Eventos de ajuste gerados por um componente ajustado, por exemplo, quando o valor de uma barra de rolagem é mudado
DocumentListener	Evento para receber notificações quando ocorre alguma mudança no documento
FocusListener	Eventos de foco, gerados quando um componente recebe ou perde o foco, por exemplo, quando um campo texto recebe o foco
ItemListener	Eventos gerados quando o item selecionado de uma lista é mudado, por exemplo, quando o usuário escolhe um item de um componente List ou Combo
KeyListener	Refere-se aos eventos do teclado que ocorrem quando uma tecla é pressionada, quando é solta etc.
MouseListener	Refere-se aos eventos gerados pelo mouse quando é clicado, entra na área de um componente, sai da área do componente etc.
MouseMotionListener	Refere-se a eventos do mouse, gerados pela movimentação dele sobre um componente
WindowListener	Refere-se a eventos de janela, gerados quando uma janela é maximizada, minimizada etc.
ComponentListener	Refere-se a qualquer componente de uma janela, gerado quando o componente se torna visível, oculto, é movido ou redimensionado. Por exemplo, quando um botão é redimensionado, ou quando uma janela é oculta

Controles de Eventos

■ Manipuladores de Eventos

Método	Alguns componentes gráficos que podem usá-lo
<code>addActionListener()</code>	JButton, JCheckBox, JComboBox, JTextField e JRadioButton
<code>addItemListener()</code>	JButton, JCheckBox, JComboBox, JRadioButton
<code>addDocumentListener()</code>	JTexttField, JTextArea
<code>addFocusListener()</code>	Todos os componentes do swing
<code>addAdjustmentListener()</code>	JScrollBar
<code>addMouseListener()</code>	Todos os componentes do swing
<code>addMouseMotionListener()</code>	Todos os componentes do swing
<code>addKeyListener()</code>	Todos os componentes do swing
<code>addWindowListener()</code>	JWindow e JFrame
<code>addComponentListener()</code>	Todos os componentes do swing

FIM