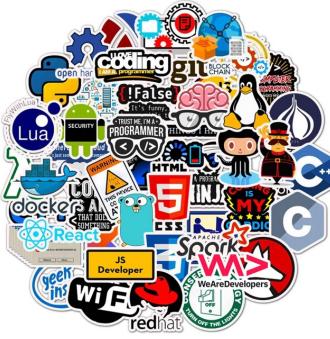




Quem se prepara, não para.



Linguagens de Programação

3º período

Prof. Dr. João Paulo Aramuni

Sumário

- Java
 - Interface Gráfica com Usuário
 - SWING
 - Componentes
 - Ambiente de desenvolvimento

Sumário

- Java
 - **Interface Gráfica com Usuário**
 - SWING
 - Componentes
 - Ambiente de desenvolvimento

- **Conceitos básicos de Interface com Usuário**

- Geralmente quando se está começando a programar, o desenvolvedor começa a fazer códigos que são retornados no console em formato de texto, pois muitos são códigos de aprendizagem.
- Mas quando é necessário desenvolver sistemas que precisam de alguma interação mais aprimorada com o usuário, utiliza-se as interfaces gráficas.

- **Conceitos básicos de Interface com Usuário**

- A **Interface Gráfica com Usuário** (Graphical User Interface – GUI) é formada através de componentes GUI.
- Esses componentes são objetos que fazem a interação com usuário por teclado, mouse ou outros dispositivos que venham a servir para entrada de dados.

- **Conceitos básicos de Interface com Usuário**

- Ao projetar os formulários da aplicação, o desenvolvedor deve considerar:
 - **Conteúdo e Estética**
 - Se os dados que estão no formulário atendem as necessidades dos usuários.
 - Distribuição, Agrupamento, Dimensionamento, Fonte, Cor, Espaçamento e Alinhamento.
 - **Facilidade Operacional (produtividade e confiabilidade)**
 - Solicitação de serviços do aplicativo
 - (Menu, Barra de Ferramentas, Botões, Atalhos...)
 - Preenchimento dos dados (Validação)
 - Interpretação de resultados (Formatação)

- **Conceitos básicos de Interface com Usuário**

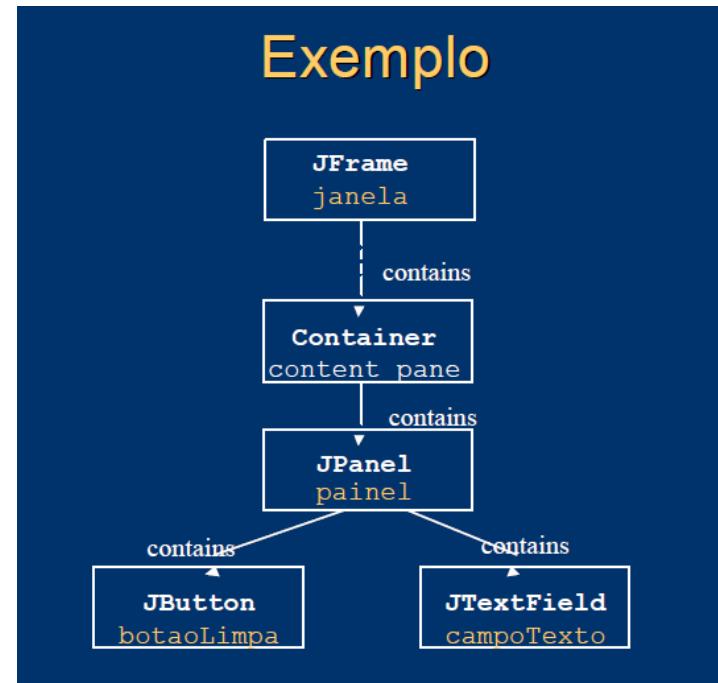
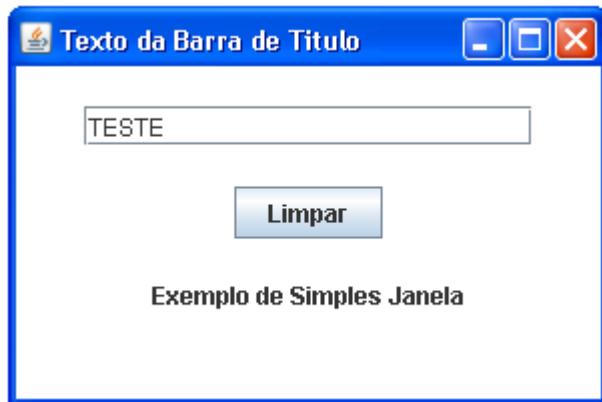
- Processo Básico: OO + Eventos
 - **1) Instanciar os componentes de interface**
 - Por exemplo, janelas, botões, campos de textos, etc.
 - **2) Adicionar os componentes em containers**
 - Por exemplo, como os componentes podem ser agrupados e qual o layout de diagramação.
 - **3) Estabelecer o tratamento de eventos de interface**
 - Por exemplo, o que deve ocorrer quando o usuário clicar em um botão ou como alterar o conteúdo de um componente quando um outro sofre alguma alteração.

Sumário

- Java
 - Interface Gráfica com Usuário
 - **SWING**
 - Componentes
 - Ambiente de desenvolvimento

- **Swing**

- Exemplo



Java

- Código

```
ExemploJanela.java X
7 import javax.swing.JFrame;
8 import javax.swing.JLabel;
9 import javax.swing.JPanel;
10 import javax.swing.JTextField;
11
12 public class ExemploJanela extends JFrame {
13
14     //Atributos
15     private JPanel painel = new JPanel();
16     private JButton jButtonLimpar = new JButton("Limpar");
17     private JTextField jTextFieldTexto = new JTextField("Teste", 20);
18     private JLabel jLabelMensagem = new JLabel("Exemplo Simples de Janela");
19
20     //Construtor
21     public ExemploJanela() {
22         this.setTitle("Texto da Barra de Título");
23         this.setSize(300,200);
24         painel.setLayout(new FlowLayout(FlowLayout.CENTER, 40, 20));
25         painel.setBackground(new Color(255,255,255));
26         painel.add(jTextFieldTexto);
27         painel.add(jButtonLimpar);
28         painel.add(jLabelMensagem);
29         this.getContentPane().add(painel);
30         this.setLocationRelativeTo(null); // Centralizar janela
31         this.setVisible(true); // Exibir janela
32     }
33
34
35     public static void main(String[] args) {
36         new ExemploJanela();
37     }
38 }
```



- **Swing**

- O Java suporta, oficialmente, dois tipos de bibliotecas gráficas: **AWT** e **Swing**.
- A AWT (Abstract Window Toolkit) foi a primeira API para interfaces gráficas a surgir no Java e foi, mais tarde, superada pelo Swing, que possui diversos benefícios em relação ao seu antecessor.
- O JavaFX surgiu posteriormente como uma promessa para substituir o Swing, mas foi descontinuado.

- **Swing**

- As bibliotecas gráficas são bastante simples no que diz respeito a conceitos necessários para usá-las. A complexidade no aprendizado de interfaces gráficas em Java reside no tamanho das bibliotecas e no enorme mundo de possibilidades; isso pode assustar, em um primeiro momento.
- AWT e Swing são bibliotecas gráficas oficiais inclusas em qualquer JRE ou JDK. Além destas, existem algumas outras bibliotecas de terceiros, sendo a mais famosa, o SWT - desenvolvida pela IBM e utilizada no Eclipse e em vários outros produtos.

- **Swing**

- Grande parte da complexidade das classes e métodos do Swing está no fato da API ter sido desenvolvida tendo em mente o máximo de portabilidade possível.
- Favorece-se, por exemplo, o posicionamento relativo de componentes, em detrimento do uso de posicionamento fixo, que poderia prejudicar usuários com resoluções de tela diferentes da prevista.

- **Swing**

- Com Swing, não importa qual sistema operacional, qual resolução de tela, ou qual profundidade de cores: sua aplicação **se comportará da mesma forma** em todos os ambientes.
- Os componentes AWT são mais pesados, pois requerem uma interação direta com o sistema de janela local, podendo restringir na aparência e funcionalidade, ficando menos flexíveis do que os componentes GUI Swing.
- O Swing traz muitos componentes para usarmos: **botões, entradas de texto, tabelas, janelas, abas, scroll, árvores de arquivos** e muitos outros. Iremos explorar a maioria deles.

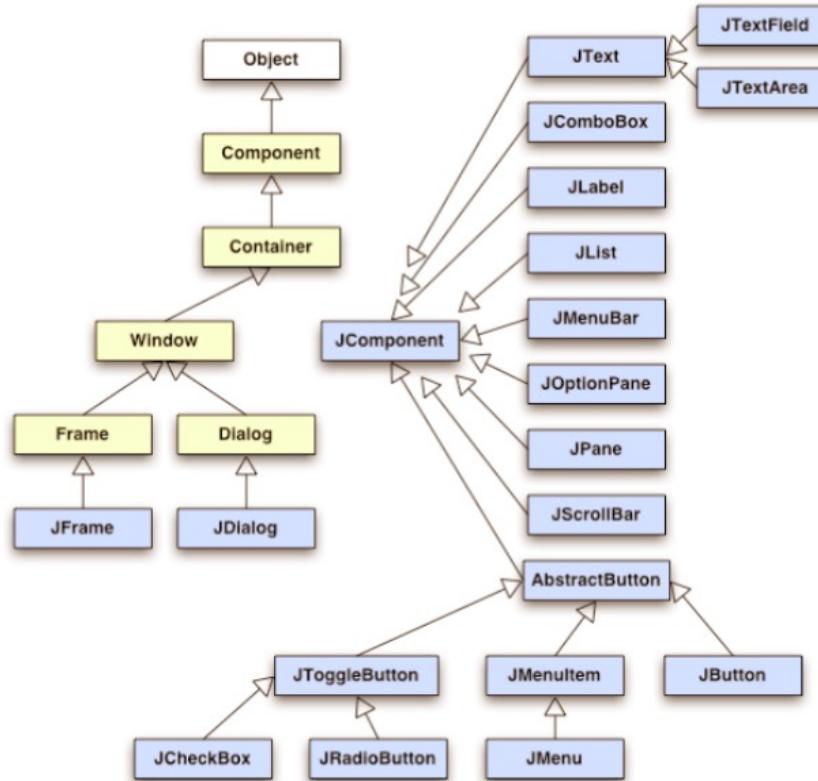
- **Swing**

- A biblioteca do Swing está no pacote javax.swing (inteira, exceto a parte de acessibilidade, que está em javax.accessibility).
- Alguns componentes não são do tipo GUI Swing e sim componentes **AWT**.
- A diferença entre o GUI Swing e AWT, está na aparência e comportamento dos componentes, ou seja, quando criado por AWT, a aparência e comportamento de seus componentes são diferentes para cada plataforma e enquanto feito por GUI Swing, a aparência e comportamento funcionam da mesma forma para todas as plataformas.

- **Swing**

- Componentes
- Diagrama resumido de classes

AWT e Swing



- **Swing**

- Os componentes AWT são mais pesados, pois requerem uma interação direta com o sistema de janela local, podendo restringir na aparência e funcionalidade, ficando menos flexíveis do que os componentes GUI Swing.
- Vejamos alguns componentes mais utilizados.

- **Swing**

- **JLabel** - Exibe texto não editável ou ícones.
- **JTextField** – Insere dados do teclado e serve também para exibição do texto editável ou não editável.
- **JButton** – Libera um evento quando o usuário clicar nele com o mouse.
- **JCheckBox** – Especifica uma opção que pode ser ou não selecionada.
- **JComboBox** – Fornece uma lista de itens onde possibilita o usuário selecionar um item ou digitar para procurar.
- **JList** – Lista de itens onde pode ser selecionado vários itens.
- **JPanel** – É a área onde abriga e organiza os componentes inseridos.

Sumário

- Java
 - Interface Gráfica com Usuário
 - SWING
 - **Componentes**
 - Ambiente de desenvolvimento

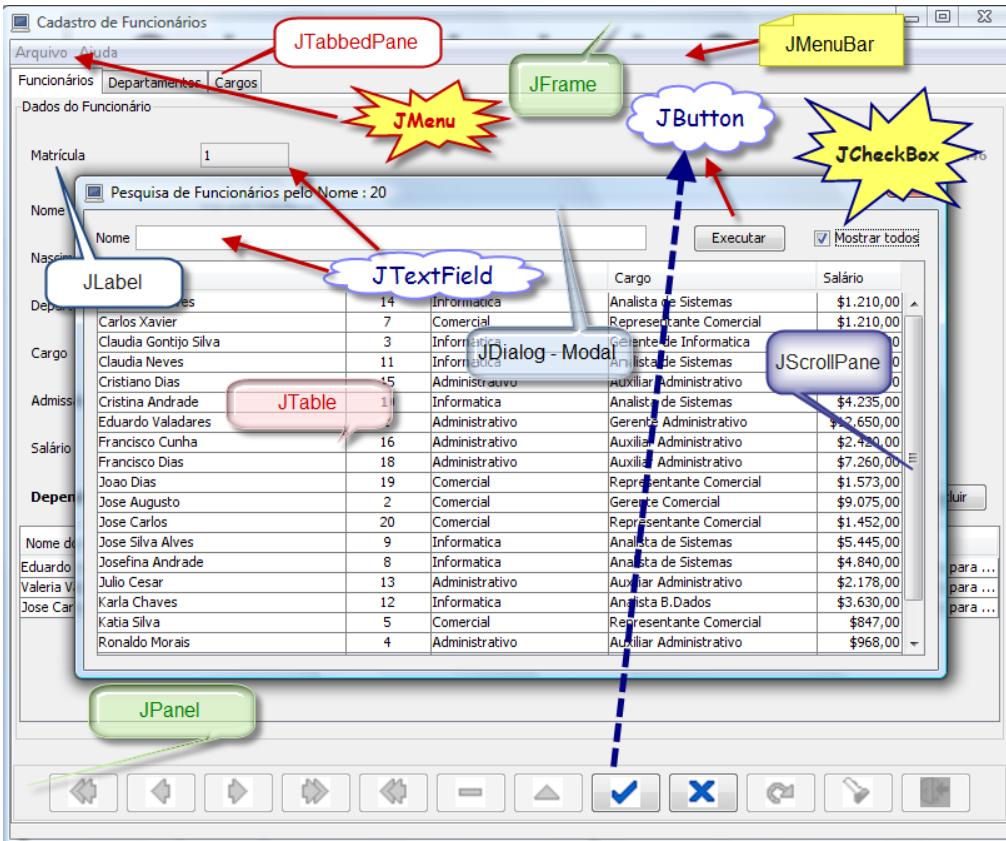
- **Outros componentes:**

Componente	Classe Swing
Janela (Container Principal)	JFrame / JDialog
Painel (Container Intermediário)	JPanel
Painel com Scroll	JScrollPane
Painel Tabulado	JTabbedPane
Barra de Menu	JMenuBar
Barra de Ferramentas	JToolBar
Menu	JMenu
Item de Menu	JMenuItem
Separação de item de Menu	JSeparator
Popup Menu	JPopupMenu

- **Outros componentes:**

Componente	Classe Swing
Rótulo	JLabel
Campo de Texto / Formatado	JTextField / JFormattedTextField
Campo de Senha	JPasswordField
Área de Texto	JTextArea
Botão (c/ ou s/ imagem)	JButton
Caixa de Opção	JCheckBox
Botão de Rádio	JRadioButton , ButtonGroup
Lista	JList
Caixa de Seleção	JComboBox
Tabela (Grid)	JTable
Árvore	JTree

• Exemplo

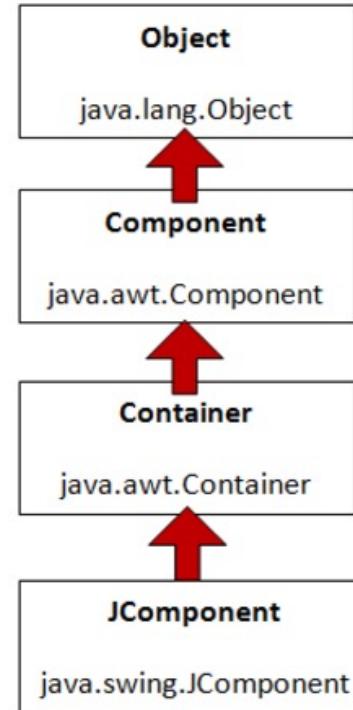


- **Componentes**

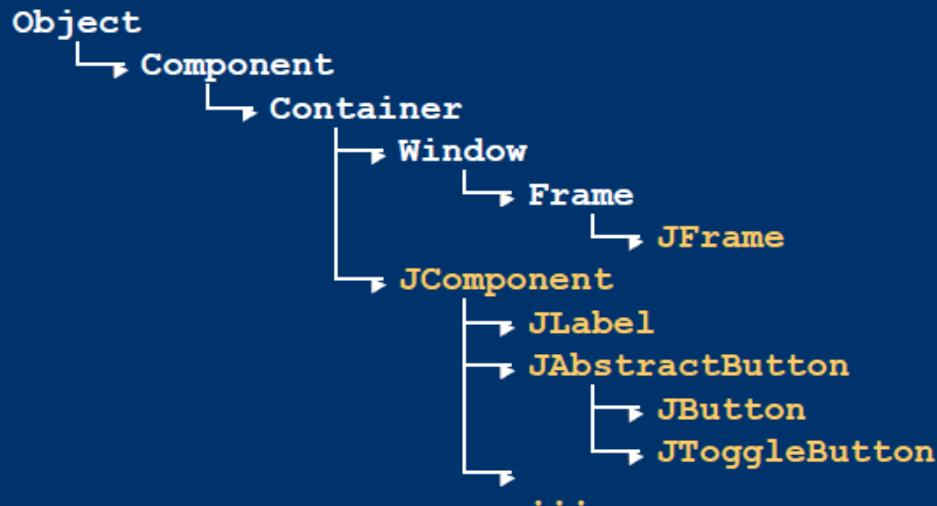
- Principais métodos:
 - **JLabel** – Nome de campo na tela
 - setText(String)
 - **JTextField / JTextArea** – Entrada e Saída de Dados
 - getText() / setText(String) / requestFocus()
 - **JButton** – Botão de comando
 - setText(String) ; isEnabled() ; setEnabled(boolean)
 - **jCheckBox** – Opções Múltiplas
 - boolean isSelected()
 - setSelected(boolean)
 - **ButtonGroup / JRadioButton** – Opções Exclusivas
 - objButtonGroup.clearSelection()
 - objRadioButton.isSelected()

- **Componentes**

- Os componentes possuem classes que definem quais serão seus estados e comportamentos.
- Veja ao lado como é a hierarquia dessas classes:



Hierarquia de Classes



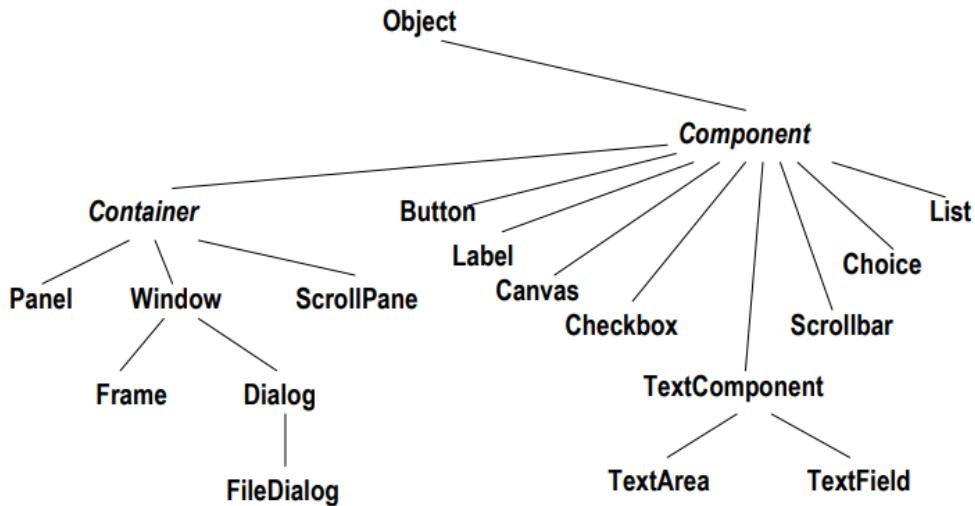
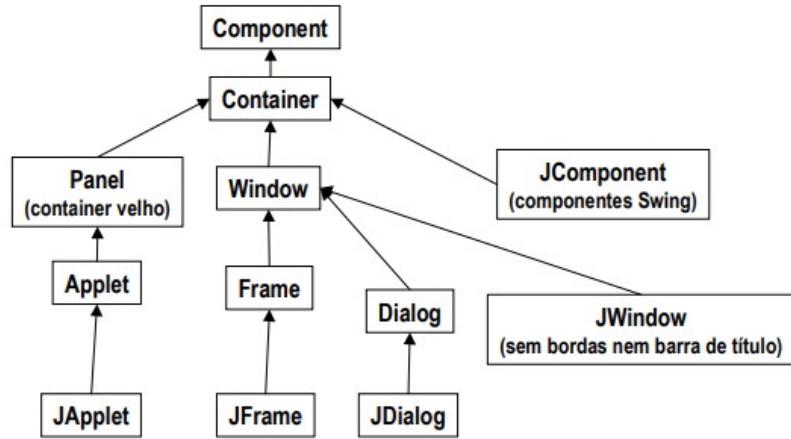
- **Componentes**

- **Componente** é qualquer elemento de interface.
- **Container** é um componente que agrega Componentes.
- Uma **janela** é um *top-level* container: onde os outros componentes são desenhados.
- Um **painel** é um container intermediário: serve para facilitar o agrupamento de outros componentes.
- Botões e campos de texto são exemplos de **componentes atômicos**: elementos de interface que não agrupam outros componentes. Mostram para o usuário e/ou obtêm dele alguma informação. A API do Swing oferece muitos componentes atômicos.

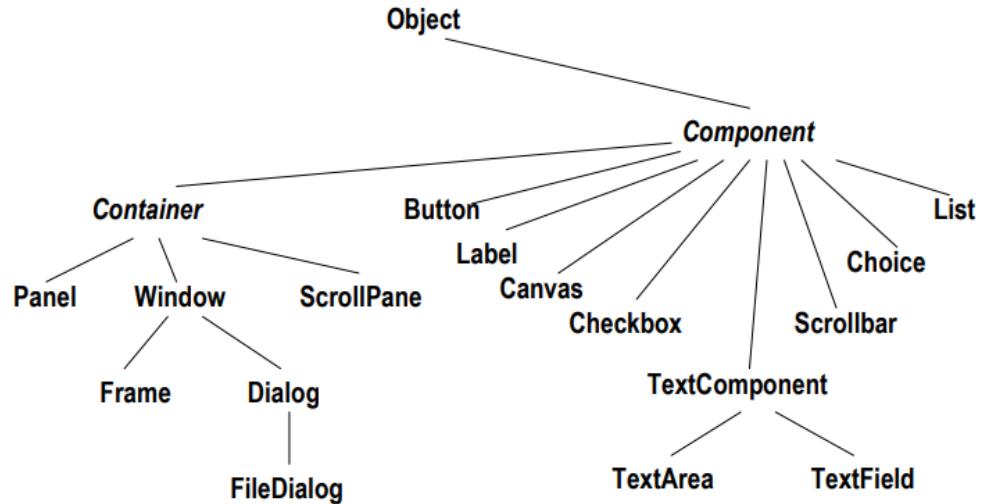
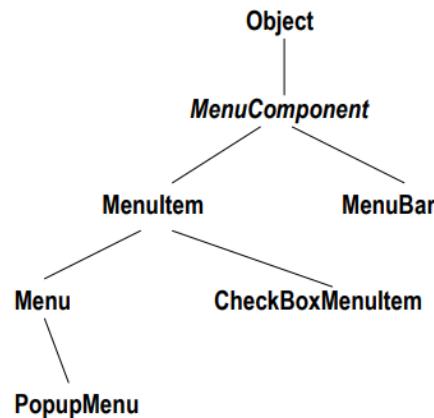
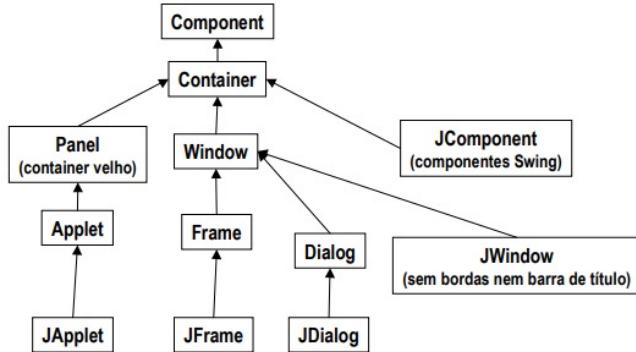
- **Componentes**

- Uma interface gráfica em Java é baseada em dois elementos:
 - – containers: servem para agrupar e exibir outros componentes
 - – componentes: botões, labels, scrollbars, etc.
- Dessa forma, todo programa que ofereça uma interface vai possuir pelo menos um container, que pode ser:
 - – JFrame: janela principal do programa
 - – JDialog: janela para diálogos
 - – JApplet: janela para Applets
- Para **construirmos uma interface gráfica em JAVA**, adicionamos componentes (Botões, Menus, Textos, Tabelas, Listas, etc.) sobre a área da janela.
 - Por essa razão a área da janela é um container, ou seja, um elemento capaz de armazenar uma lista de componentes.

- **Componentes**



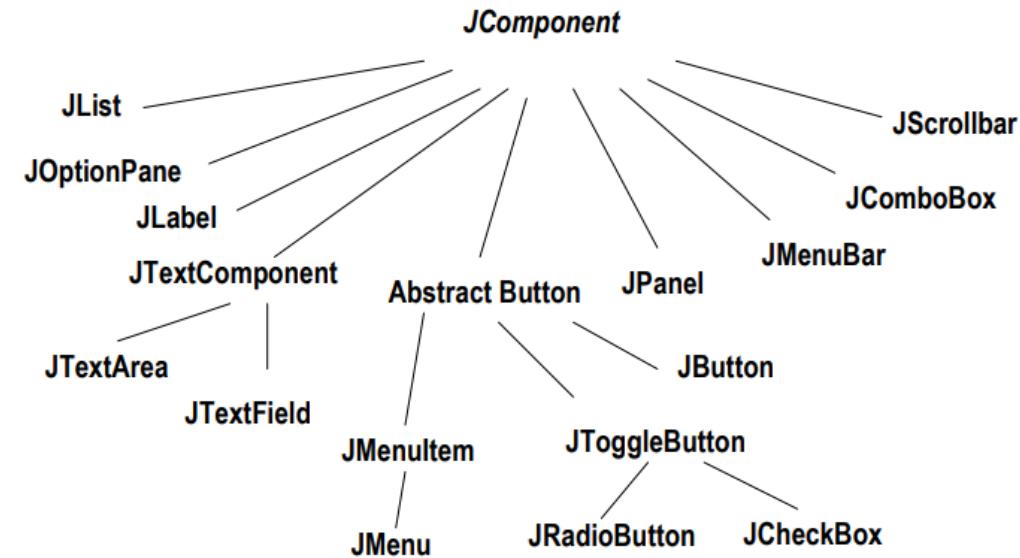
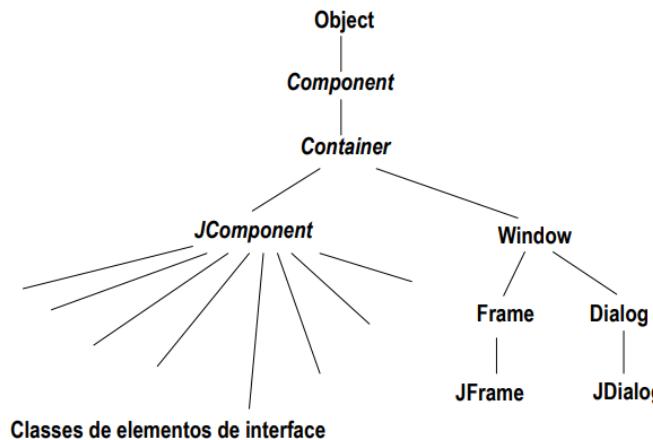
AWT



AWT

• Componentes

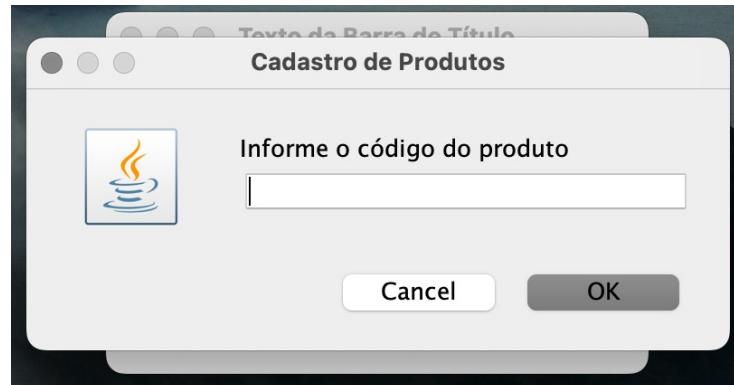
- Visuais:
 - botões, menus, barras de ferramentas, etc.
- Não-visuais, de auxílio aos outros:
 - root pane, panel, layered pane, etc.



- Componentes

- Exemplo Caixa de Entrada e de Caixa de Mensagem:

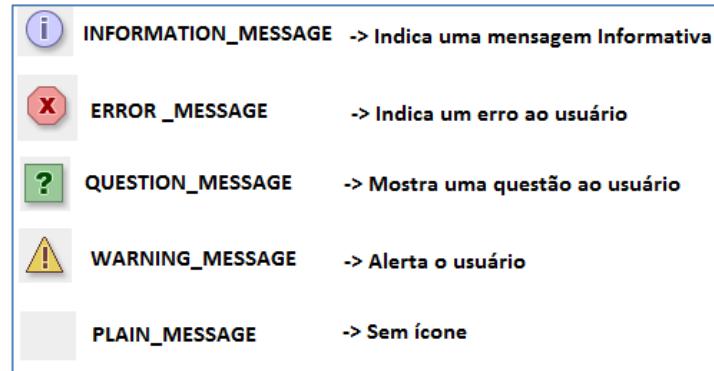
```
String codigo = JOptionPane.showInputDialog(null,"Informe o código do produto",  
        "Cadastro de Produtos", JOptionPane.QUESTION_MESSAGE);  
  
if ( codigo == null || !codigo.matches("[0-9]*") ) return;  
JOptionPane.showMessageDialog(painel, "" + Integer.parseInt(codigo));
```



- Componentes

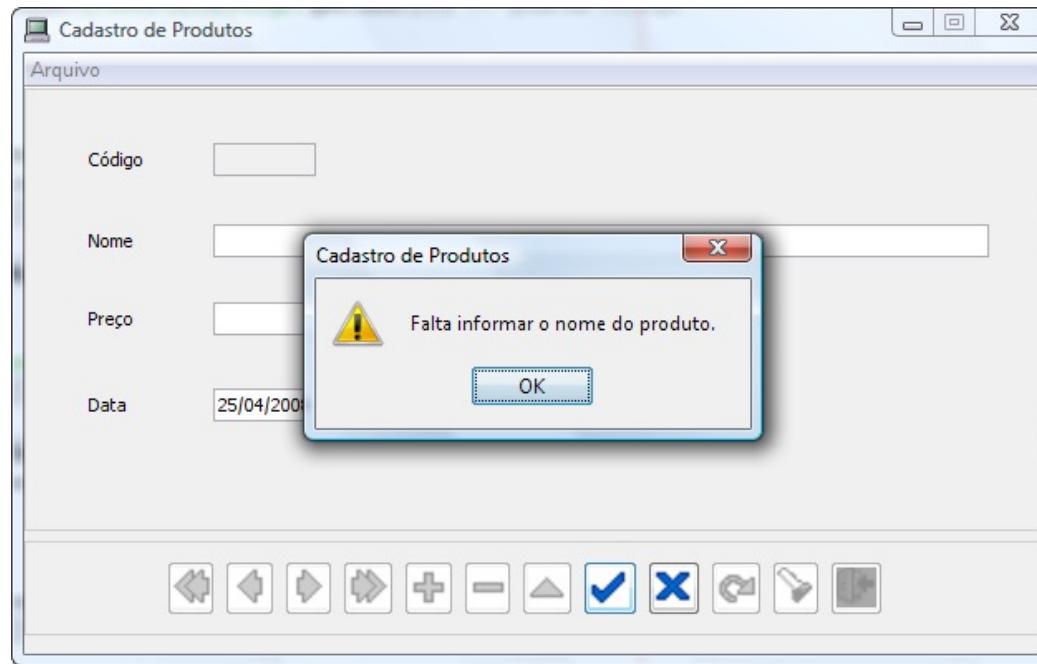
- Outro exemplo de Caixa de Mensagem:

```
if (jTextFieldTexto.getText().trim().equals("")) {  
    JOptionPane.showMessageDialog(null, "Falta informar o código do produto.",  
        "Cadastro de Produtos", JOptionPane.WARNING_MESSAGE);  
    jTextFieldTexto.requestFocus();  
}
```



- Componentes

- Outro exemplo de Caixa de Mensagem:



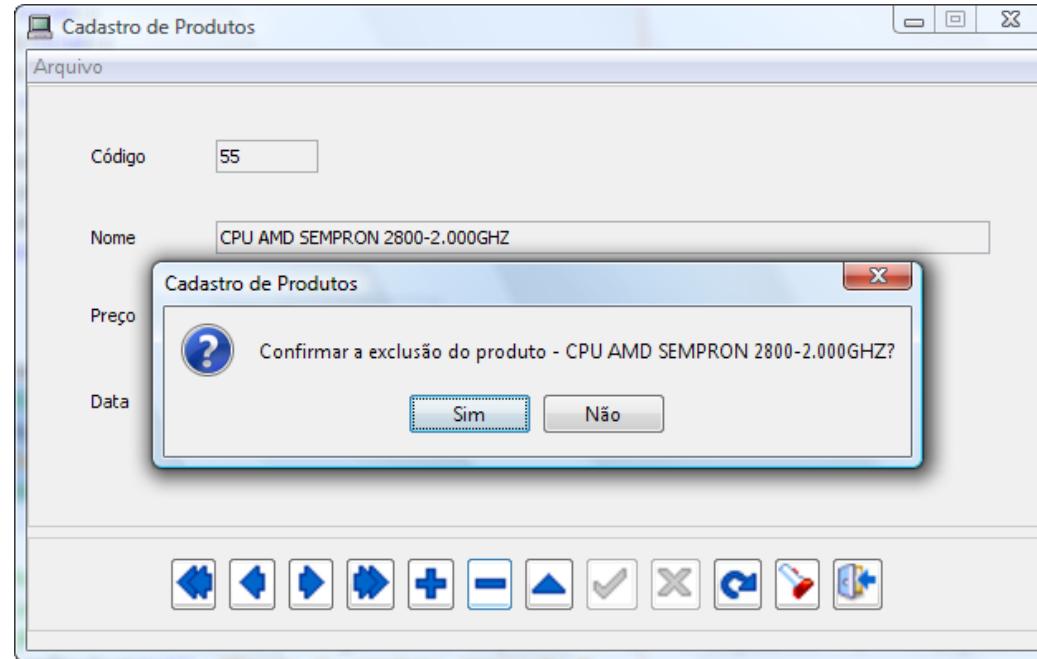
- **Componentes**

- Exemplo de Caixa de Confirmação:

```
if (JOptionPane.showConfirmDialog(null, "Confirmar a exclusão do produto"  
+ jTextFieldTexto.getText().trim() + "?", "Cadastro de Produtos",  
JOptionPane.YES_NO_OPTION) == JOptionPane.NO_OPTION ) {  
    return;  
}  
// JOptionPane.YES_NO_OPTION  
// JOptionPane.YES_OPTION  
// JOptionPane.NO_OPTION
```

- **Componentes**

- Exemplo de Caixa de Confirmação:



- **Componentes**

- Exemplo de Label:

```
// Cria um label com texto
JLabel label1 = new JLabel("Label1: Apenas texto");

// Cria um label com texto e imagem
JLabel label2 = new JLabel("Label2: Imagem e texto", new ImageIcon("javalogo.gif"), JLabel.CENTER);
label2.setVerticalTextPosition(JLabel.BOTTOM);
label2.setHorizontalTextPosition(JLabel.CENTER);
```

- **Componentes**

- Exemplo de TextField:

```
// Cria um campo de nome
JTextField campoNome = new JTextField(10);
JLabel labelNome = new JLabel("Nome: ");
labelNome.setLabelFor(campoNome);
labelNome.setDisplayedMnemonic('n'); // Alt-n

// Cria um campo de email
JTextField campoEmail = new JTextField(10);
JLabel labelEmail = new JLabel("Email: ");
labelEmail.setLabelFor(campoEmail);
labelEmail.setDisplayedMnemonic('E'); // Alt-e
```

- Componentes

- Exemplo de Button:

```
// Cria um botão com texto
JButton botao1 = new JButton("Botão Desabilitado");
botao1.setEnabled(false);
botao1.setToolTipText("Exemplo de um botão de texto");
botao1.setMnemonic(KeyEvent.VK_D); // Alt-D

// Cria um botão com texto e imagem
JButton botao2 = new JButton("Botão Habilitado", new ImageIcon("javalogo.gif"));
botao2.setToolTipText("Botão de texto e imagem");
botao2.setMnemonic(KeyEvent.VK_H); // Alt-H
botao2.setPressedIcon(new ImageIcon("javalogo2.gif"));
```

- Componentes

- Exemplo de Checkbox:

```
// Exemplo de Checkbox
JFrame f = new JFrame("Teste");
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
JCheckBox ci = new JCheckBox("Inglês", true);
JCheckBox ce = new JCheckBox("Espanhol", true);
JCheckBox cf = new JCheckBox("Francês");
Container cp = f.getContentPane();
cp.setLayout(new FlowLayout());
cp.add(ci);
cp.add(ce);
cp.add(cf);
f.pack();
f.setVisible(true);
```

- Componentes

- Exemplo de RadioButton:

```
// Exemplo de RadioButton
JFrame f2 = new JFrame("Teste");
f2.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
JRadioButton bm = new JRadioButton("Masculino", true);
JRadioButton bf = new JRadioButton("Feminino");
ButtonGroup bg = new ButtonGroup();
bg.add(bm);
bg.add(bf);
Container cp2 = f.getContentPane();
cp2.setLayout(new FlowLayout());
cp2.add(bm);
cp2.add(bf);
f2.pack();
f2.setVisible(true);
```

- Componentes

- Exemplo de JFrame:

```
// Exemplo de JFrame
JFrame janela = new JFrame("Exemplo de Janela");
janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

JLabel mensagem = new JLabel("Olá Mundo");
janela.getContentPane().add(mensagem);
janela.setLocationRelativeTo(null); // centraliza
janela.setIconImage(new ImageIcon("javalogo2.gif").getImage());
JMenuBar menuBar = new JMenuBar();
menuBar.add(new JMenu("Menu"));
janela.setJMenuBar(menuBar);
janela.pack();
janela.setVisible(true);
```

- **Componentes**

- Exemplo de Grid:

```
// Exemplo de Grid
Container contentPane = janela.getContentPane();

contentPane.setLayout(new GridLayout(0,2));

contentPane.add(new JButton("Button 1"));
contentPane.add(new JButton("2"));
contentPane.add(new JButton("Button 3"));
contentPane.add(new JButton("Long-Named Button 4"));
contentPane.add(new JButton("Button 5"));
```

- **Componentes**

- Exemplo de Flow:

```
// Exemplo de Flow
Container contentPane2 = janela.getContentPane();
contentPane2.setLayout(new FlowLayout());

contentPane2.add(new JButton("Button 1"));
contentPane2.add(new JButton("2"));
contentPane2.add(new JButton("Button 3"));
contentPane2.add(new JButton("Long-Named Button 4"));
contentPane2.add(new JButton("Button 5"));
```

- **Componentes**

- Exemplo de Border:

```
// Exemplo de Border
Container contentPane3 = janela.getContentPane();
//contentPane3.setLayout(new BorderLayout()); // Desnecessário

contentPane3.add(new JButton("Button 1 (NORTH)"), BorderLayout.NORTH);
contentPane3.add(new JButton("2 (CENTER)"), BorderLayout.CENTER);
contentPane3.add(new JButton("Button 3 (WEST)"), BorderLayout.WEST);
contentPane3.add(new JButton("Long-Named Button 4 (SOUTH)"), BorderLayout.SOUTH);
contentPane3.add(new JButton("Button 5 (EAST)"), BorderLayout.EAST);
```

- **Componentes**

- **Alguns atributos de componentes:**

- posição (x,y): posição do objeto em relação ao seu container;
- nome do componente (myWindow.setName("Teste"));
- tamanho: altura e largura;
- cor do objeto e cor de fundo;
- fonte;
- aparência do cursor;
- objeto habilitado ou não (isEnabled(), myWindow.setEnabled());
- objeto visível ou não (isVisible(), myWindow.setVisible());
- objeto válido ou não.

- **Componentes**

- **Alguns métodos de componentes:**

- void setEnabled(boolean enabled);
- void setToolTipText(String text);
- void setBounds(int x, int y, int width, int height);
- void setBounds(Rectangle rect);
- Rectangle getBounds();
- void setSize(Dimension d);
- Dimension getSize();
- setLocation(int x, int y);
- setLocation(Point p);
- Point getLocation();

- **Componentes**

- Orientação por eventos
 - Um modelo de programação que tornou-se bastante difundido com o uso de interfaces gráficas foi a **programação orientada por eventos**.
 - Segundo esse modelo, o programa deixa de ter o controle do fluxo de execução, que passa a ser um sistema encarregado de gerenciar a interface.
 - Assim, o programa passa a ser chamado pelo sistema quando algum **evento** é gerado na interface.

```
15
16 public class TesteJanela extends JFrame{
17
18     // Atributos
19     private JPanel painel = new JPanel();
20     private JButton jButtonLimpar = new JButton("Limpar");
21     private JTextField jTextFieldTexto = new JTextField("Teste", 20);
22     private JLabel jLabelMensagem = new JLabel("Exemplo de Simples Janela");
23
24     // Construtor
25     public TesteJanela(){
26         this.setTitle("Exemplo de Interface Gráfica");
27         this.setSize(400,200);
28         configurarComponentes();
29         this.setLocationRelativeTo(null); // Centralizar janela
30         this.setVisible(true); // Exibir janela
31     }
32
33
34     private void configurarComponentes() {
35         jButtonLimpar.setToolTipText("Limpar formulário");
36         jButtonLimpar.setCursor(new Cursor(Cursor.HAND_CURSOR));
37         jButtonLimpar.setMnemonic('L');
38         jButtonLimpar.addActionListener(new ActionListener() {
39
40             public void actionPerformed(ActionEvent e) {
41                 limparTela();
42             }
43         });
44         painel.setLayout(new FlowLayout(FlowLayout.CENTER, 100, 20));
45         painel.setBackground(new Color(255,255,255));
46         jTextFieldTexto.setFont(new Font("Arial", Font.BOLD, 16));
47         painel.add(jTextFieldTexto);
48         painel.add(jButtonLimpar);
49         painel.add(jLabelMensagem);
50         this.getContentPane().add(painel);
51     }
52
53
54
55     private void limparTela() {
56         jTextFieldTexto.setText("");
57         jTextFieldTexto.requestFocus();
58     }

```

Evento de click no botão



Tabela de ações para eventos:
(Pacote `java.awt.event`)

Ação que dispara o evento	Tipo de listener (que escuta a ação)
Usuário clica em um botão, pressiona return em uma caixa de texto, ou seleciona um item de menu.	ActionListener
Usuário fecha um frame (janela principal da aplicação).	WindowListener
Usuário pressiona o botão do mouse enquanto o cursor está sobre um componente.	MouseListener
Usuário move o mouse sobre um componente.	MouseMotionListener
Usuário move o wheel sobre um componente.	MouseWheelListener
Componente se torna visível.	ComponentListener
Componente obtém o foco do teclado.	FocusListener
Usuário pressiona alguma tecla.	KeyListener
Item selecionado muda em uma tabela ou lista	ListSelectionListener

Sumário

- Java
 - Interface Gráfica com Usuário
 - SWING
 - Componentes
 - Ambiente de desenvolvimento

- **Ambiente de desenvolvimento**

- Seria extremamente complexo desenvolver interfaces gráficas utilizando somente o código-fonte textual.
- Por esse motivo, precisamos de ferramentas ou frameworks que facilitem a visualização prévia da interface gráfica durante o trabalho do programador.
- Essas ferramentas são conhecidas como ferramentas de “**drag-and-drop**” de componentes.
Ou seja, arrastar e soltar componentes para dentro da sua aplicação.

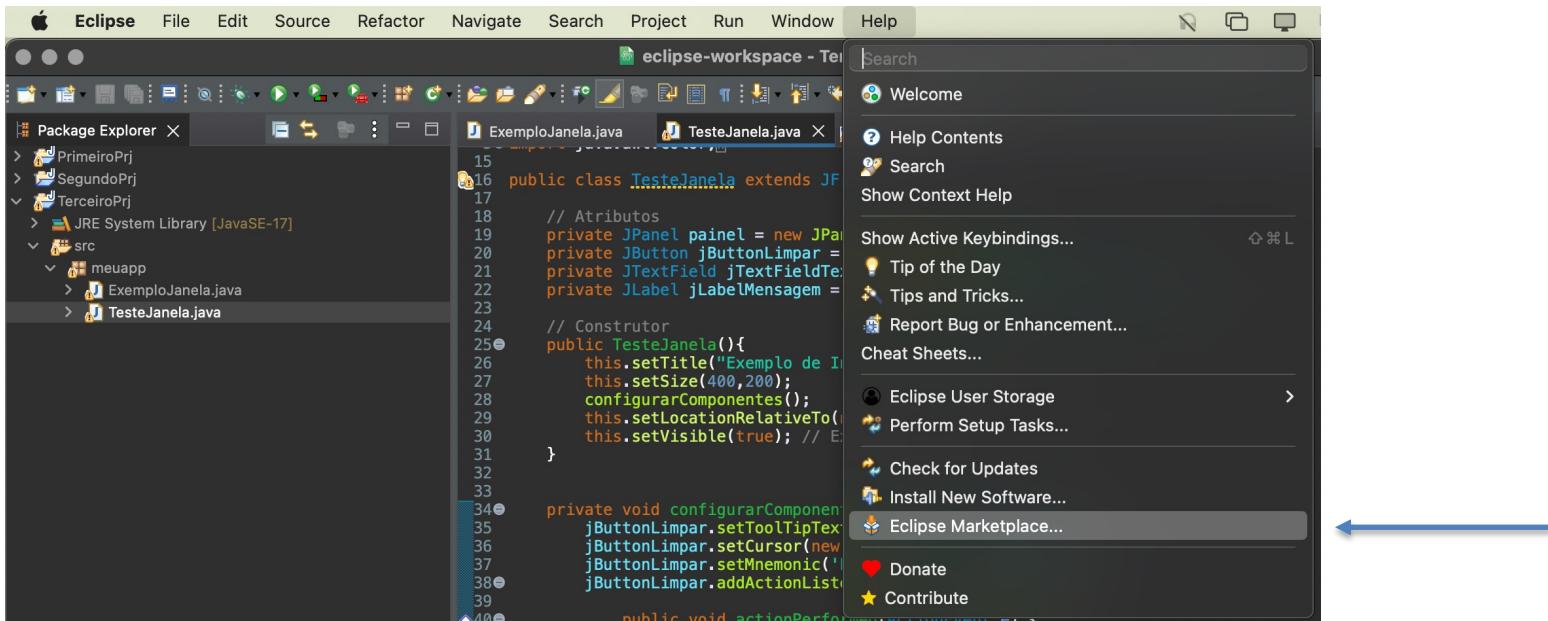
- **Ambiente de desenvolvimento**

- No caso deste curso, iremos utilizar o Eclipse **WindowBuilder**.
- O WindowBuilder possui acesso fácil aos componentes SWT e Swing.
- Link:
<https://projects.eclipse.org/projects/tools.windowbuilder/downloads>



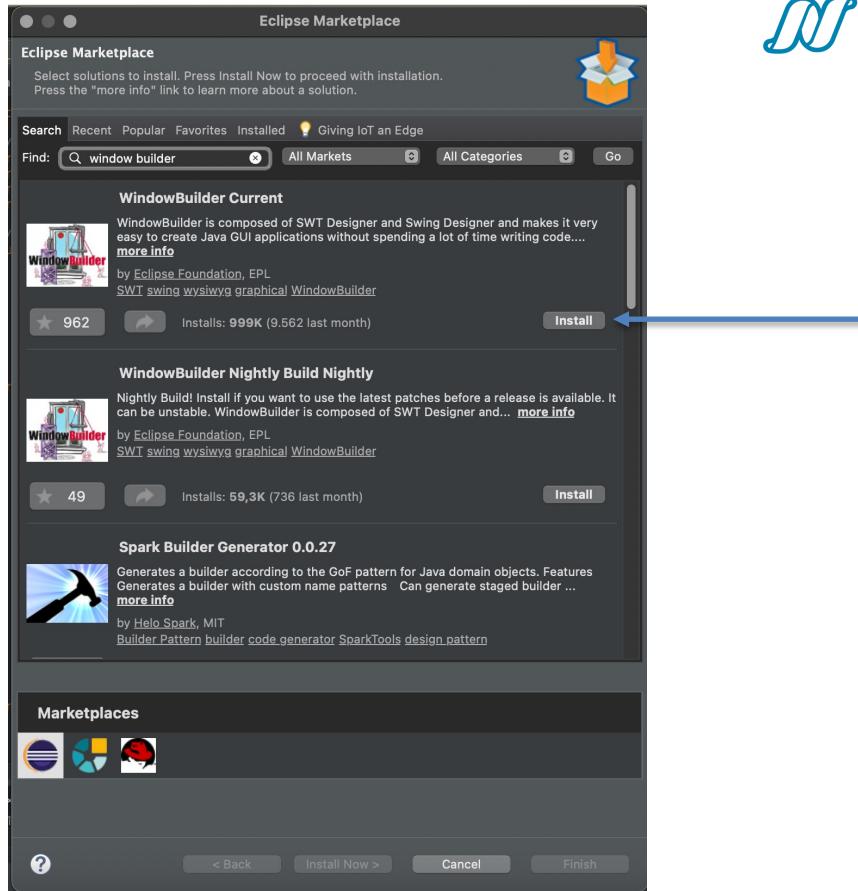
- Ambiente de desenvolvimento

- Como instalar:



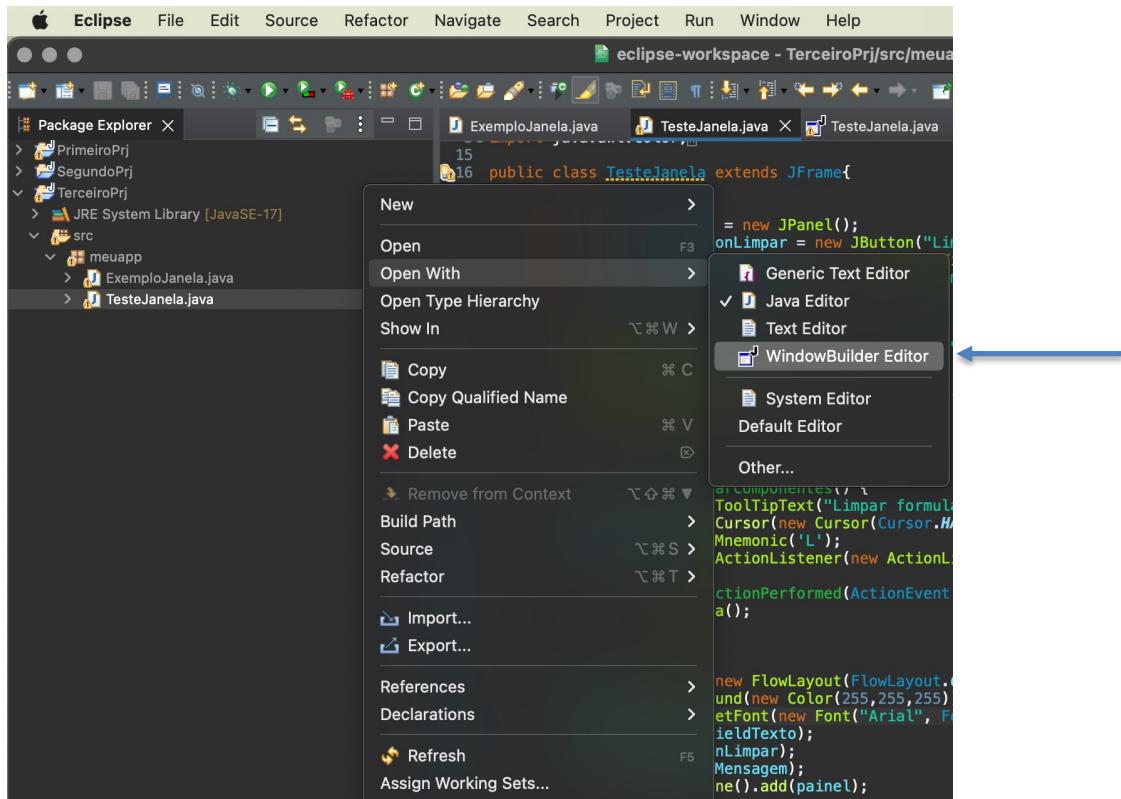
Java

- Ambiente de desenvolvimento
 - Como instalar:



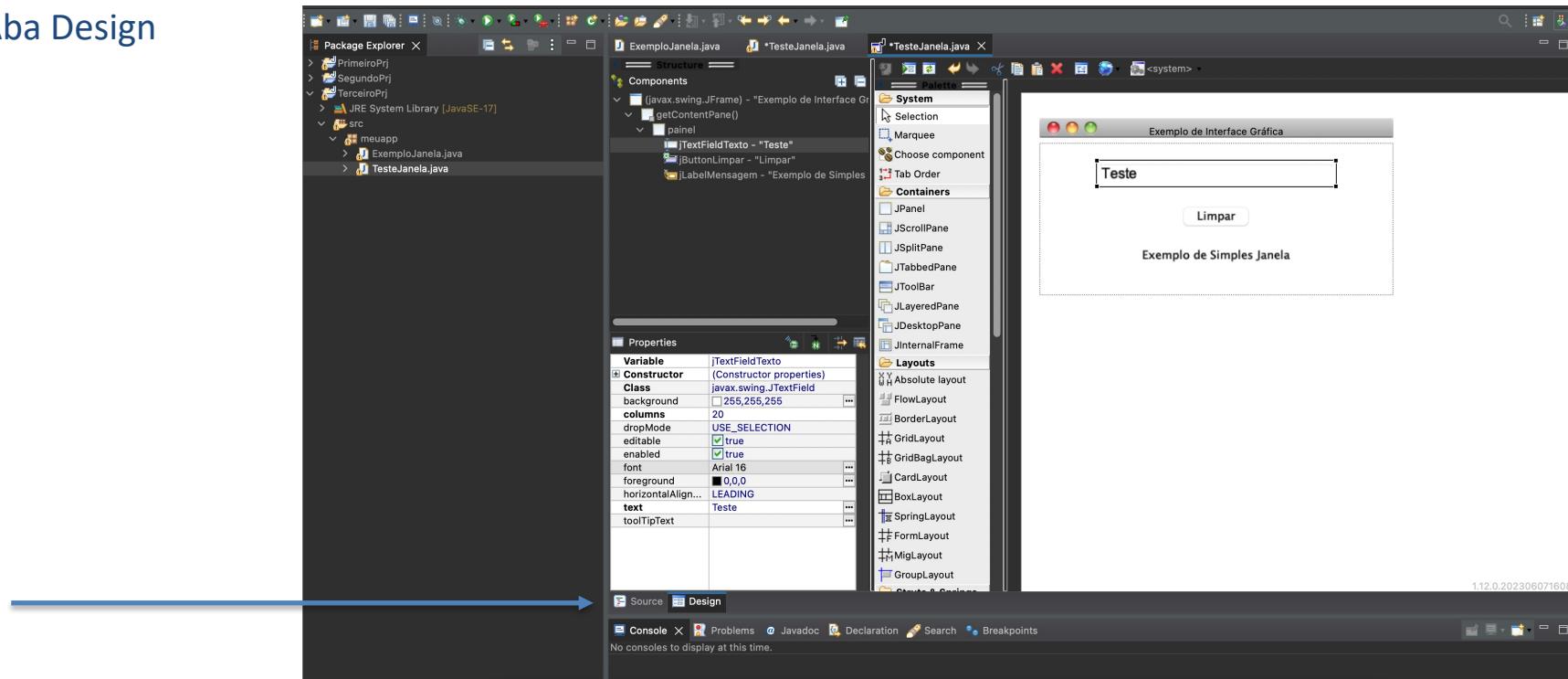
• Ambiente de desenvolvimento

- Como instalar:
- Após a instalação,
reinicie o Eclipse

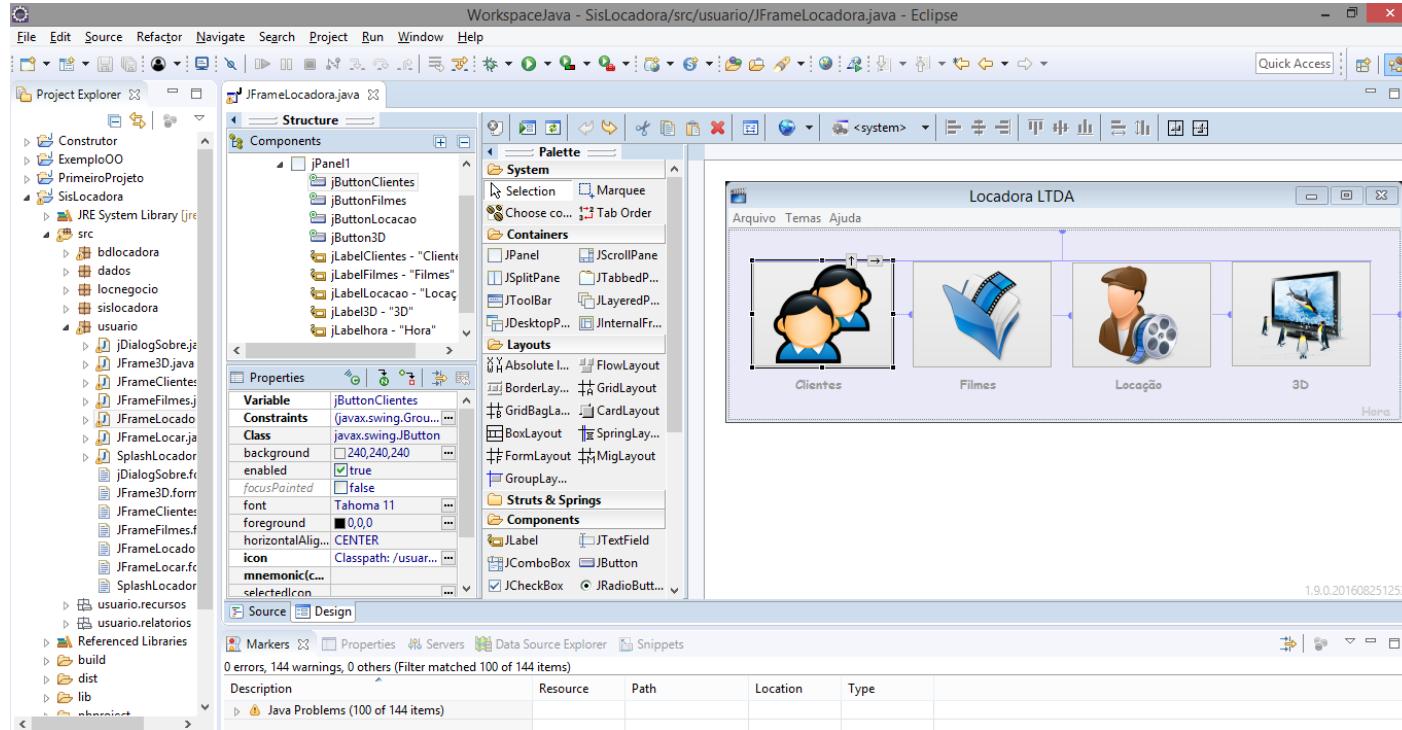


- Ambiente de desenvolvimento

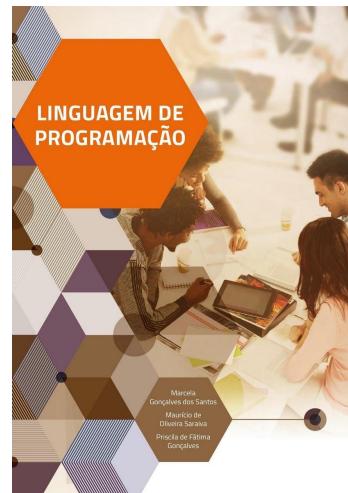
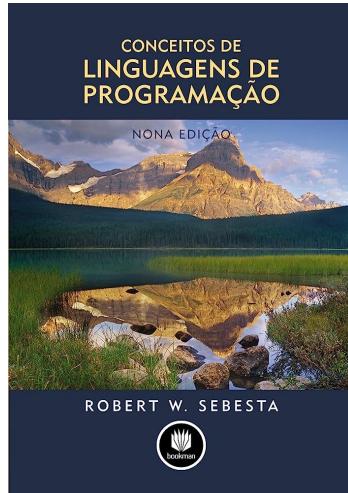
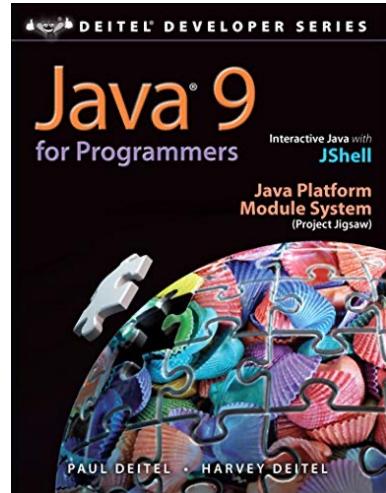
- Aba Design



- Exemplo de Sistema para Locadora construído no **WindowBuilder**.



Referências





Obrigado!

joaoaramuni@newtonpaiva.br