

# **Java - Swing**

# Parte I

# Java *Abstract Window Toolkit* (AWT)

- *Abstract Windowing Toolkit* - AWT

➔ Em 1995, a Sun criou a API *Abstract Window Toolkit* (AWT) para J2SE 1.0;

➔ Conjunto básico de componentes gráficos de Java para uso em GUI.

- Nas primeiras versões de Java era a única forma de desenvolver GUI
- Muito limitados.
- “Write once, test everywhere”

➔ Características

- Fina camada de abstração sob e GUI nativa;
- Alta fidelidade ao toolkit nativo;
- Maior integração com aplicações nativas;
- Interfaces desenvolvidas em uma plataforma não ficavam bonitas em outras;



# Java Foundation Classes - JFC

- Java Foundation Classes

➡ É um conjunto de pacotes (15) usados para criação de interfaces gráficas com o usuário (GUI).

- Framework oficial provido pela plataforma Java SE para construção de GUIs portáteis.

➡ União das tecnologias AWT, Swing e Java2D;

- Swing é um kit de ferramentas GUI e faz parte da JFC.
- Java2D: criação de desenhos em duas dimensões em Java;

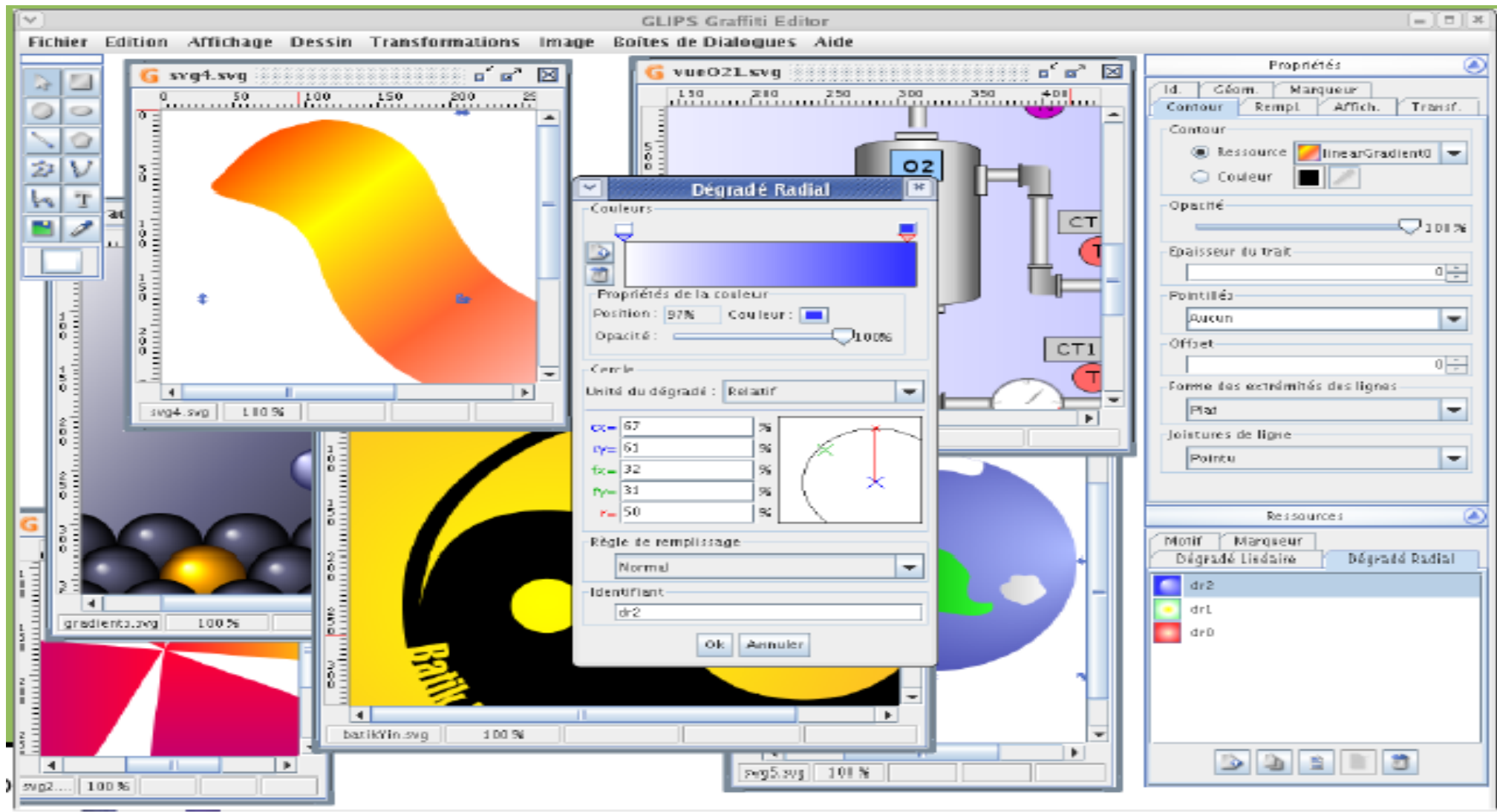
➡ Características da JFC:

- Componentes gráficos Swing (escritos em Java, fornecem um alto nível de portabilidade e flexibilidade)
- *Look&Feel* adaptável
- Recursos de arrastar e soltar
- Java2D (gráficos em 2D)
- Atalhos de teclas
- Tratamento de eventos
- Tooltips (breve descrição do componente, acionada quando o mouse é posicionado sobre o componente), ..... Entre outras.



# Exemplos de aplicações em Swing

- GLIPS (<http://glipssvgedito.sourceforge.net>):



# Exemplos de aplicações em Swing

- Jake2 (<http://bytonic.de>):

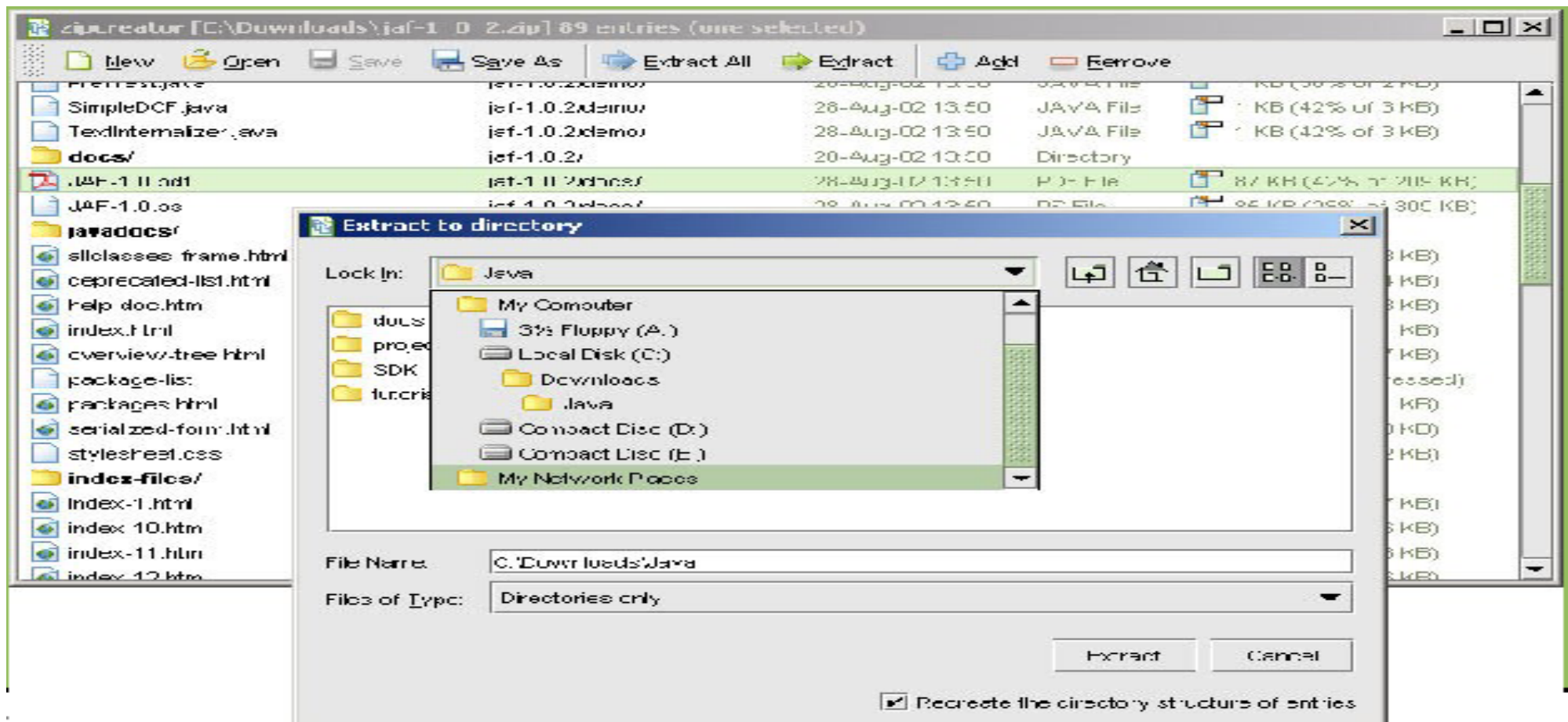
**bytonic** is a software development team from Chemnitz (Germany). We are mainly focussed on the development of Java applications. Recently we have been successful in porting the Quake2 engine to Java which you can play online using webstart.





# Exemplos de aplicações em Swing

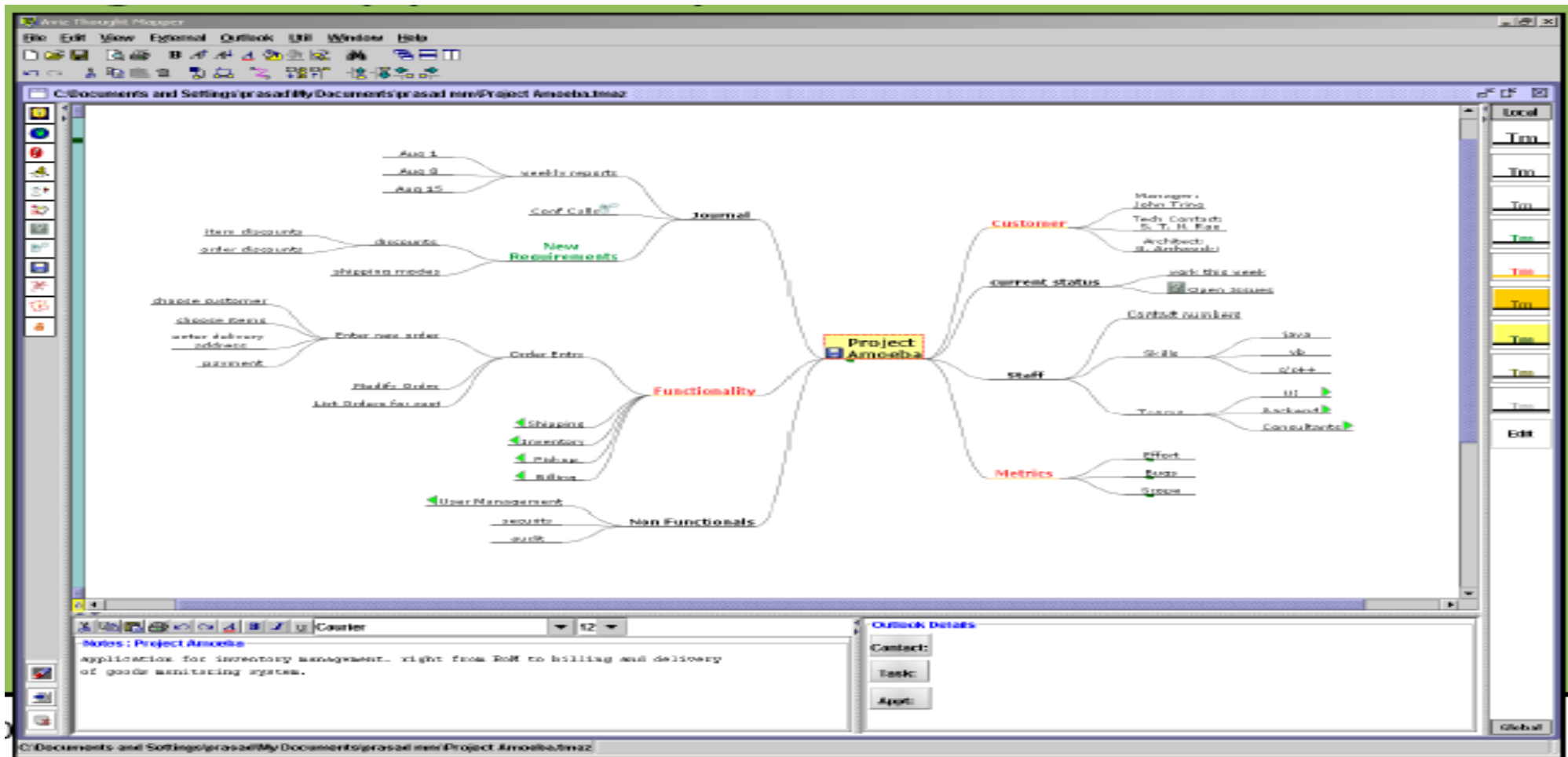
- ZipCreator (<http://www.zipcreator.com>):
  - Cross-platform zip utility: Use zipcreator to prepare zip files and extract their contents. The application is easy to use and runs on Windows, Mac OS X and Linux operating systems.





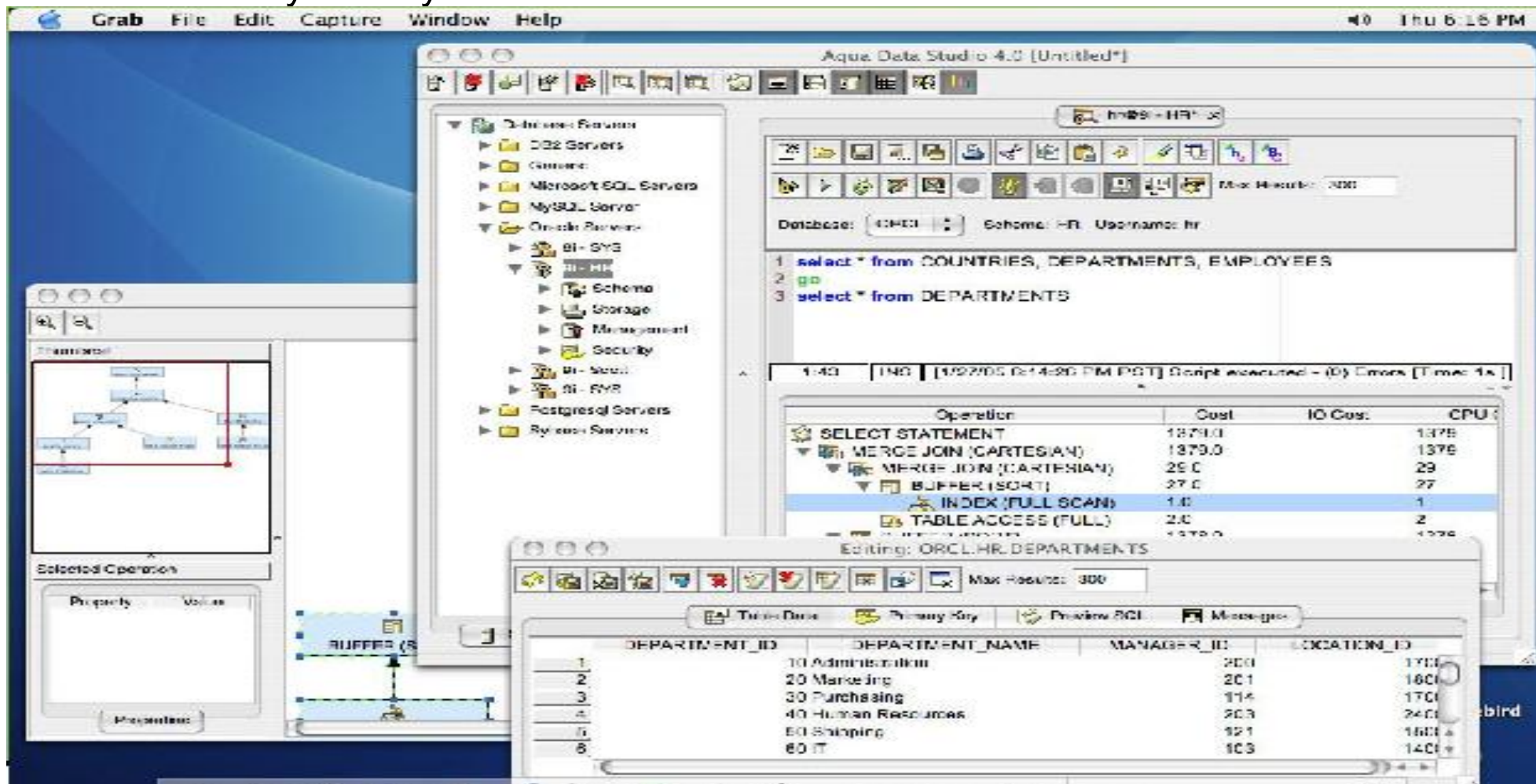
# Exemplos de aplicações em Swing

- Thought Mapper (<http://www.avizsoft.com>):
  - Thought Mapper is a powerful Mind Mapping tool that helps you Organize Thoughts Visually and ideas around a central theme by creating a Mind Map.



# Exemplos de aplicações em Swing

- Aqua Data Studio (<http://www.aquafold.com>):
  - Aqua Data Studio is a database query tool and administration tool that allows developers to easily create, edit, and execute SQL scripts, as well as browse and visually modify database structures.



# Exemplos de aplicações em Swing

- E muitas outras...

- Azureus (BitTorrent);

- Programa de declaração de ajuste anual do IR;

- NetBeans, JasperReports, Java Web Start,...;

- Etc.

# Swing e Ferramentas

➡ Como podemos trabalhar com Swing???

➡ Temos 2 formas:

- Usamos alguma ferramenta de desenho
- Ou fazemos “na mão”

# Ferramentas de desenho

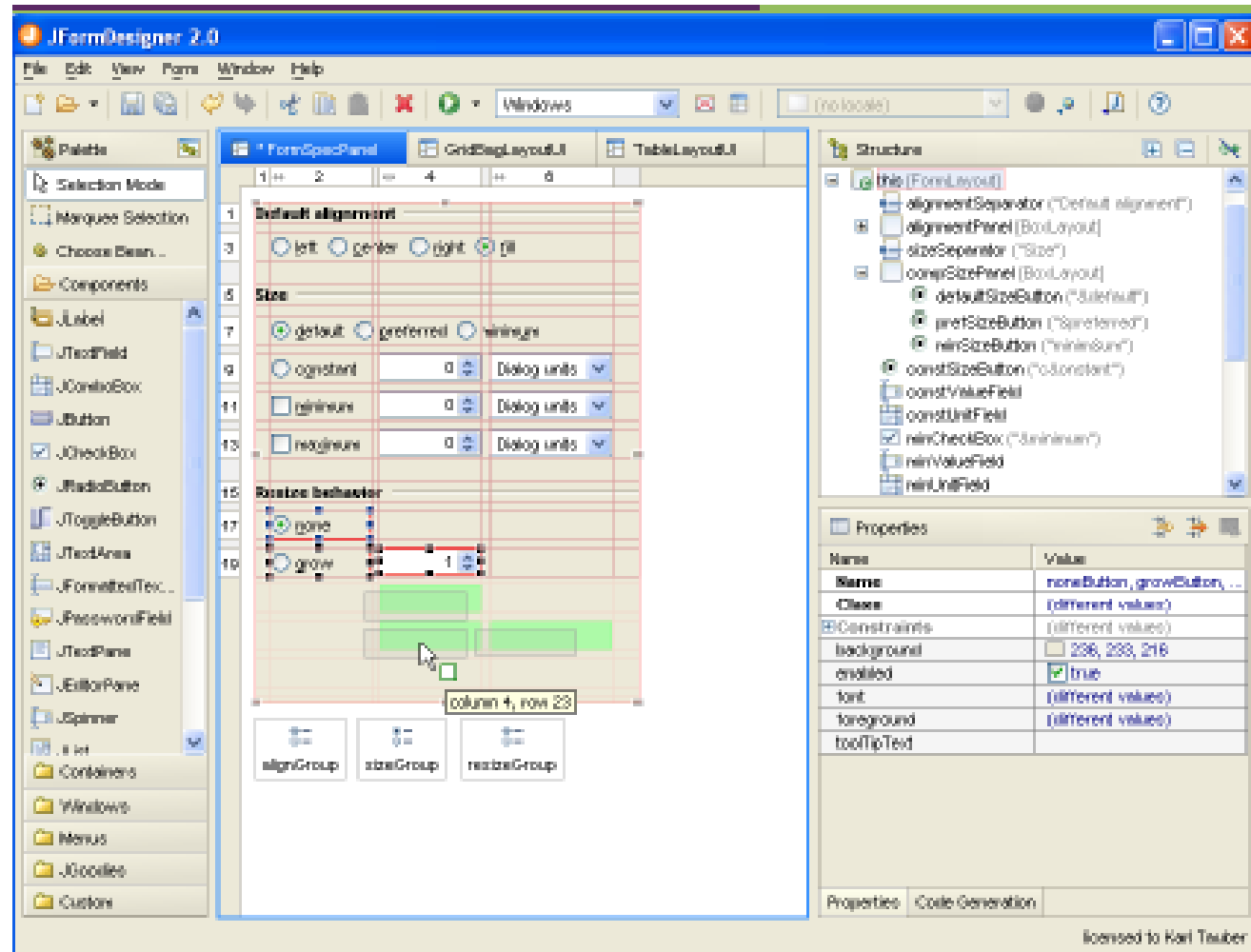
- No caso das ferramentas podemos utilizar as ferramentas

## ➔ Integradas às IDEs:

- Eclipse (Visual Edito ,Matisse4Eclipse);
- Netbeans (Matisse);
- Outras...

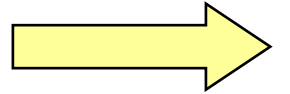
## ➔ Independentes:

- Swing Designe ➔
- JFormDesigne
- FormLayoutMake
- Abeille;
- Outras...



# Conceitos Básicos

- Blz, vamos agora começar e entender como construir “*essa parada*” de GUI em Java 😊



# Conceitos Básicos

→ Uma interface é composta por três elementos básicos

## → Componentes:

- Define um componente de interface.
  - Botões, *labels*, *listbox*, *menus*, etc.
  - Métodos como *paint()* e *repaint()*.
  - **Representado pela classe *JComponent***
    - Superclasse da maioria dos componentes *Swing*.

## → *Container* (janela, painel, etc):

- Local onde são adicionados os componentes ou outros *containers*.
  - Define um componente que pode conter outros componentes.
- Define métodos como *add()* para adicionar componentes em seu interior.
  - Possui um gerenciador de *layout*.

## → Gerenciador de *Layout*:

- Objeto responsável pelo posicionamento dos componentes inseridos num determinado *container*

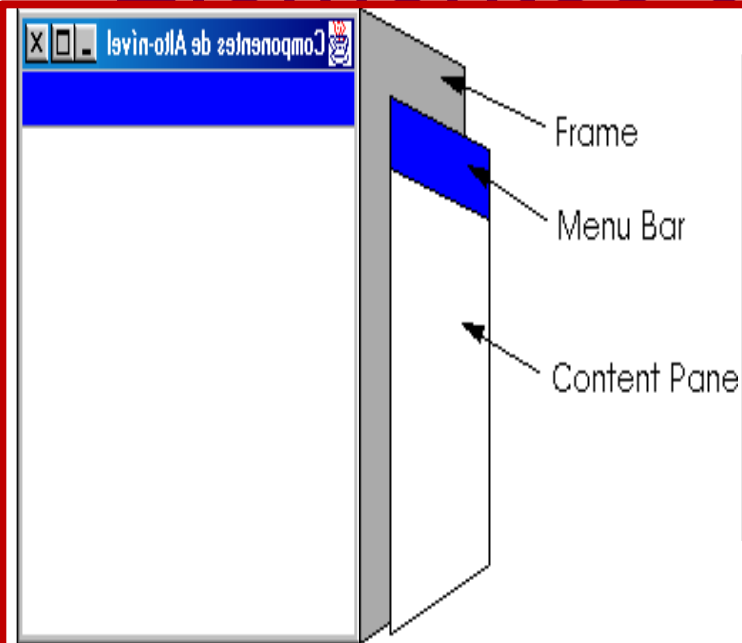


# Conceitos Básicos

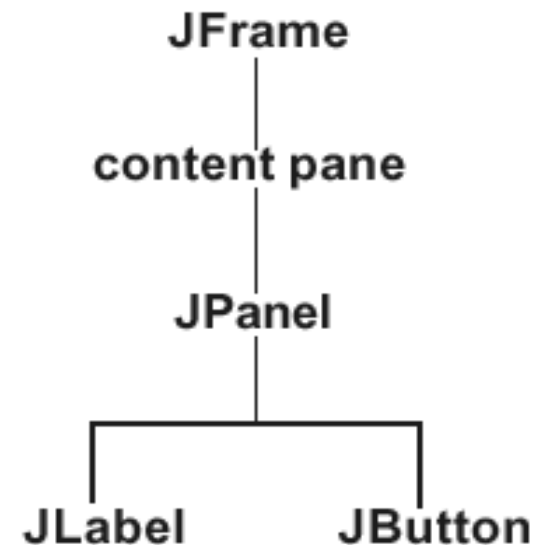
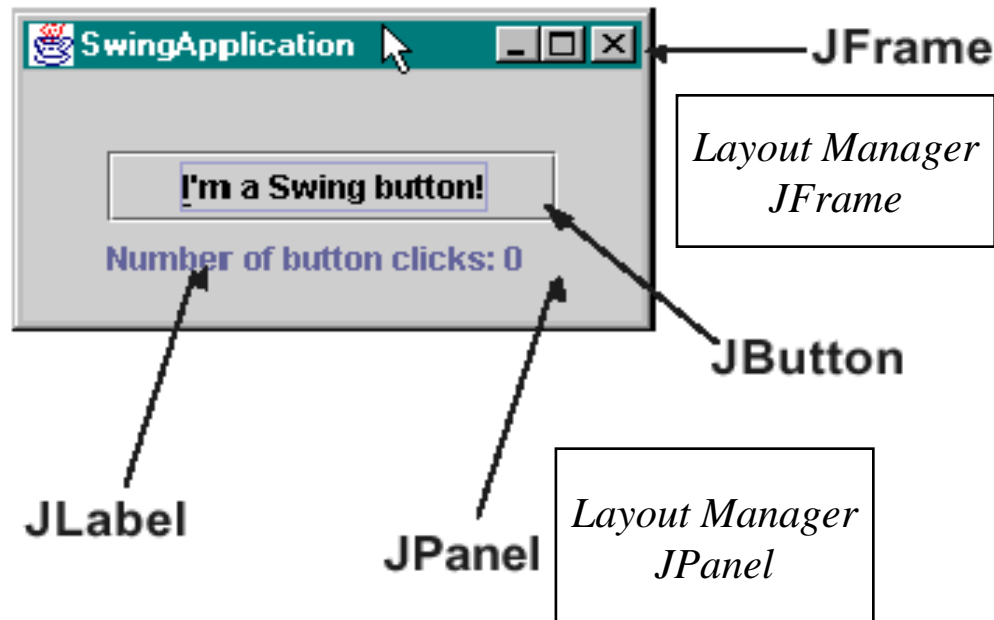
- Conceitos da Interface *Swing*

- ➔ Entre os containers, temos os Top Level Containers
  - Todo programa que utiliza componentes Swing tem pelo menos um TLC
    - Raiz de uma “*containment hierarchy*”
- ➔ Todo TLC possui um “*content pane*” que contém os elementos visíveis
  - opcionalmente, uma “*menu bar*”
- ➔ Um programa gráfico em Java pode conter 3 tipos diferentes de *containers base* (TLC):
  - Um *JFrame*,
    - Representando uma janela "principal" de aplicação;
  - Um *JDialog*
    - Para representar uma janela de diálogo (janela OK/Cancela);
  - Um *Applet*
    - Para ser usado em *Browsers*.

# Elementos de uma interface gráfica










- **JFrame** é o *top level container*.
  - Outros tipos são *applets* (JApplet) e janelas de diálogo (JDialog).
- Todo *top-level container* possui um **content pane** onde os componentes são adicionados.
  - **JPanel** é um container intermediário usado para agrupar e posicionar outros componentes.



# Java e Swing

- Componentes de interface de usuario – JFC

## ➡ Básicos

	<b>JFrame</b>	Janela principal
	<b>JPanel</b>	Panel
	<b>JButton</b>	Botão de comando e botão de toolbar
	<b>JLabel</b>	Label
	<b>JTextField</b>	Campo de texto editável e não editável (uma linha)
	<b>JCheckBox</b>	Botão de seleção includente
	<b>JRadioButton</b>	Botão de seleção excludente

# Java e Swing

- Componentes de interface de usuario – JFC

➔ Básicos



**JList**

Lista



**JComboBox**

Lista de opções com edição e sem edição



**JPasswordField**

Campo de edição oculta, para senhas



**JTextArea**

Área de texto editável



**JScrollBar**

Barras de deslizar



**JSlider**

Sliders



**JProgressBar**

Barra de progresso

# Java e Swing

- Componentes de interface de usuario – JFC

➔ Compostos



**JTextPane**

Panel editável com estilo de texto



**JEditorPane**

Panel de edição de textos



**JTable**

Tabela



**JTree**

Vista em forma de árvore

# Java e Swing

- Componentes de interface de usuario – JFC

➔ Container, Painéis com divisores, janelas



**JScrollPane**

Panel com barras de deslizamento



**JSplitPane**

Divisor de panel divisão



**JTabbedPane**

Divisor de panel em abas



**JWindow**

Janela plana



**JInternalFrame**

Janela interna em múltiplas janelas



**JDesktopPane**

Panel del escritório



**JApplet**

Container de *applets*

# Java e Swing

- Componentes de interface de usuario – JFC

➔ Caixas de diálogos



**JOptionPane**

Caixa de alerta



**JDialog**

Caixa de diálogo, janela secundária



**JColorChooser**








Seletor de cor



# Java e Swing

- Componentes de interface de usuario – JFC

➡ Barra de menus

	<b>JMenuBar</b>	Barra de menu
	<b>JMenuItem</b>	Elemento do menu
	<b>JRadioButtonMenuItem</b>	Item do menu em forma de RadioButton
	<b>JCheckBoxMenuItem</b>	Item do menu em forma de CheckBox
	<b>JSeparator</b>	Elemento separador no menu
	<b>JPopupMenu</b>	Menu contextual
	<b>JMenu</b>	Menú <i>Drop-down</i> e sub-menu

# Java e Swing

- Componentes de interface de usuario – JFC

➔ Toolbars



**JToolBar**

*Ttoolbar*



**JToggleButton**

Botão da *toolbar*



**JToolTip**

Mensagem de ajuda - hints

# Java e Swing

- Criação aplicativos Swing – Opção 2:

O que o programa está fazendo?

```
1  import javax.swing.*;
2
3  public class HelloWorldSwing {
4      public static void main(String args[])
5      {
6          JFrame frame = new JFrame("Hello");
7          JLabel label = new JLabel(" Hello, Swing World ");
8          frame.getContentPane().add(label);
9          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
10
11          //      frame.pack();
12          frame.setSize(500, 500);
13
14
15          frame.setVisible(true);
16      }
17  }
```

Diz ao Java que o *label* que criamos é para aparecer no *frame* criado anteriormente.

**Pack() => instruem o Java a mostrar o frame (a construir a interface de acordo com o tamanho dos elementos dentro do container)**  
**-> frame.pack();**

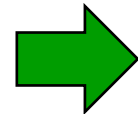


# Java e Swing

- Criação aplicativos *Swing* – Opção 2:

O que o programa está fazendo?

```
1 import java.awt.*;
2 import javax.swing.*;
3
4 public class VariosJuntos extends JFrame {
5     private JLabel label1, label2;
6     private JButton jButton1;
7     private JTextField text1;
8     private JTextArea comments;
9     private JScrollPane scroll;
10    private JCheckBox t1, t2;
11
12    private String nomeArquivo[] = { "computador.gif", "luz.gif" };
13    private JComboBox nomes;
14    private JComboBox mesesBox;
15    private ImageIcon imagem;
16 }
```

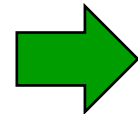


# Java e Swing

- Criação aplicativos Swing – Opção 2:

O que o programa está fazendo?

```
19 public VariosJuntos() {
20     super( "Usando Varios Componentes" );
21
22     JPanel pane = new JPanel();
23
24     // Criando JLabel com argumento String no construtor
25     label1 = new JLabel( "JLabel com texto" );
26     label1.setToolTipText( "Passe o mouse em cima do componente para ver a dica." );
27     pane.add( label1 );
28
29
30     // Criando JLabel sem argumentos no construtor
31     label2 = new JLabel();
32     label2.setText( "JLabel com texto no label via setText" );
33     label2.setToolTipText( "JLabel label2" );
34     pane.add( label2 );
35
36
37     // Criando botoes
38     jButton1 = new JButton( "Botão para Cadastrar Algo.." );
39     pane.add( jButton1 );
40
41
42     // Criando textfield
43     text1 = new JTextField();
44     text1.setText( "JTextField: Entre com texto aqui....." );
45     pane.add( text1 );
```

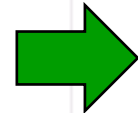


# Java e Swing

- Criação aplicativos Swing – Opção 2:

O que o programa está fazendo?

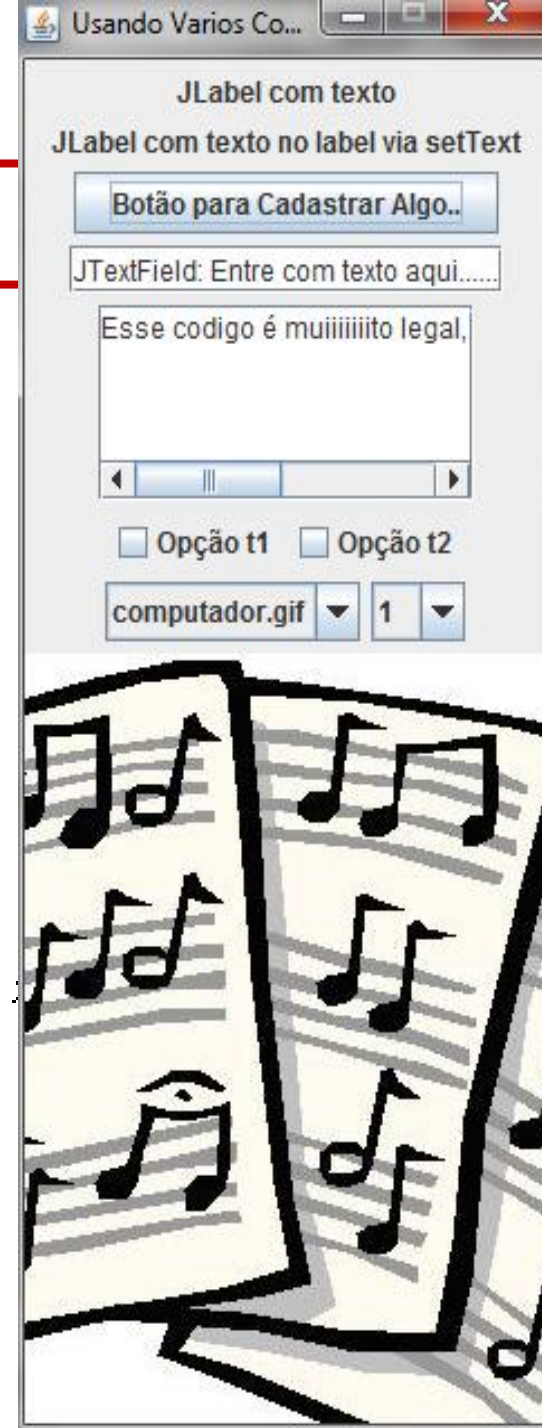
```
48 // Criando textArea
49 comments = new JTextArea(4, 15);
50 comments.setText("Esse codigo é muiiiiiiito legal, olha como funciona direitinho..");
51 scroll = new JScrollPane(comments);
52 pane.add( scroll );
53
54
55 // cria um objeto JCheckBox
56 t1 = new JCheckBox( "Opção t1" );
57 t2 = new JCheckBox( "Opção t2");
58 pane.add( t1 );
59 pane.add( t2 );
60
61
62 // Criando JComboBox
63 nomes = new JComboBox(nomeArquivo);
64 pane.add(nomes);
65
66 // Criando JComboBox
67 mesesBox = new JComboBox();
68 for (int i = 1; i < 13; i++) { mesesBox.addItem("" + i); }
69 pane.add(mesesBox);
70
71
72 imagem = new ImageIcon("desenho.jpg", "notas musicais");
73 JLabel label = new JLabel(imagem);
74 pane.add(label);
75
```



# Java e Swing

- Criação aplicativos *Swing* – Opção 2:

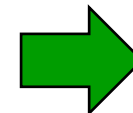
```
70
71
72
73
74
75
76
77
78     this.getContentPane().add( pane );
79     setSize( 240, 600 ); // largura x altura
80
81     setVisible(true);
82 }
83
84
85 public static void main( String args[] ) {
86
87     VariosJuntos app = new VariosJuntos();
88     app.setResizable(false);
89
90     // Evento associado a fechar a janela
91     app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
92
93 }
94
95 }
```





# Gerenciadores de *Layout*

- ➔ Os gerenciadores de *layout* são fornecidos para organizar componentes GUI em um contêiner para propósitos de apresentação.
  - Fornecem capacidades básicas de *layout* que são mais fáceis de utilizar do que determinar a posição e o tamanho exatos de cada componente GUI.
- ➔ No *Swing* há vários gerenciadores de *layout* disponíveis.
  - *FlowLayout*
  - *BorderLayout*
  - *GridLayout*
  - Vamos conversar um pouco sobre eles.....



# Gerenciadores de *Layout*

- *FlowLayout*

➔ É o gerenciador de *layout* mais básico. Os componentes GUI são colocados em um contêiner da esquerda para a direita na ordem em que são adicionados ao contêiner.

➔ Quando a borda do contêiner é alcançada, os componentes continuam na próxima linha.

➔ Se o usuário diminuir a largura da janela, alguns componentes podem passar para a linha de baixo, o que o torna um pouco instável para aplicações profissionais.

– O mesmo ocorre se o programa determinar a largura de cada componente explicitamente.

– Alinhamentos possíveis:

- Esquerda
- Direita
- Centralizados (padrão)

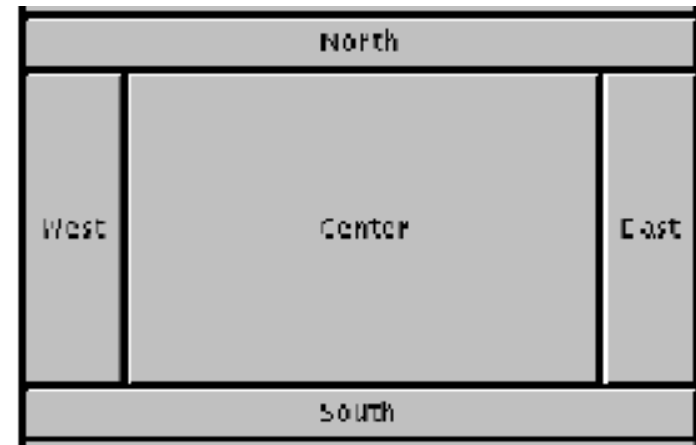


# Gerenciadores de *Layout*

- *BorderLayout*

➡ Este gerenciador de *layout* organiza componentes em cinco regiões:

- Norte
- Sul
- Leste
- Oeste
- Centro



- ➡ Se as regiões norte ou sul não forem utilizadas, as demais ocupam a área correspondente.
- ➡ Se as regiões leste ou oeste não forem utilizadas, a central ocupa a respectiva área.

# Gerenciadores de *Layout*

- *GridLayout*

- ➔ É o gerenciador de *layout* que divide o contêiner em uma grade de modo que os componentes podem ser colocados nas linhas e colunas.
- ➔ Cada componente de um *GridLayout* tem a mesma largura e altura.
- ➔ Os componentes são adicionados iniciando a célula na parte superior esquerda da grade e prosseguindo da esquerda para a direita até a linha estar cheia.

- Há dois construtores disponíveis:

- `GridLayout (int linhas, int colunas)`
  - indica apenas o tamanho da matriz
- `GridLayout (int linhas, int colunas, int espaçohorizontal, int espaçovertical)`
  - indica um espaçamento entre as regiões.



# Gerenciadores de *Layout*

- *CardLayout*

➡ É um grupo de contêineres ou componentes que são exibidos um de cada vez.

➡ Imagine um baralho num jogo de vinte-e-um. Somente uma carta é revelada por vez

- Somente a primeira “carta” é visível

➡ O modo mais comum de usar esse *layout* é usar um painel por “carta”

➡ Primeiro os componentes são acrescentados aos painéis

➡ Depois os painéis são acrescentados ao contêiner definido com o *CardLayout*.



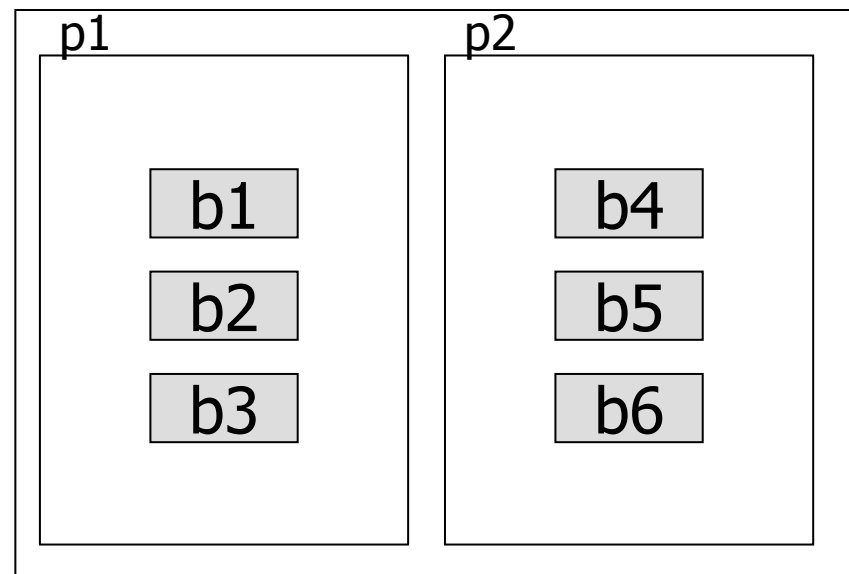
# Gerenciadores de *Layout*

- *BoxLayout*

- ➔ Gerenciador que facilita **arrumação** de componentes em **'caixas' verticais ou horizontais**
- ➔ Dimensão da 'caixa' será função do tamanho e tipo do maior componente
- ➔ Componentes são colocados em linha ou coluna única
  - **Arrumação dos componentes**
    - eixo X: esquerda para direita
    - eixo Y: cima para baixo

- **Construtor:**

- `BoxLayout (Container target, int axis)`



# Gerenciadores de *Layout*

## → *GridBagLayout*

→ É uma extensão do gerenciador de grade

– Difere desse *layout* anterior pelas seguintes características

→ Um componente pode ocupar mais de uma célula na grade

→ As proporções entre diferentes linhas e colunas não precisam ser iguais

→ Os componentes dentro das células da grade podem ser arrumados de diferentes maneiras.

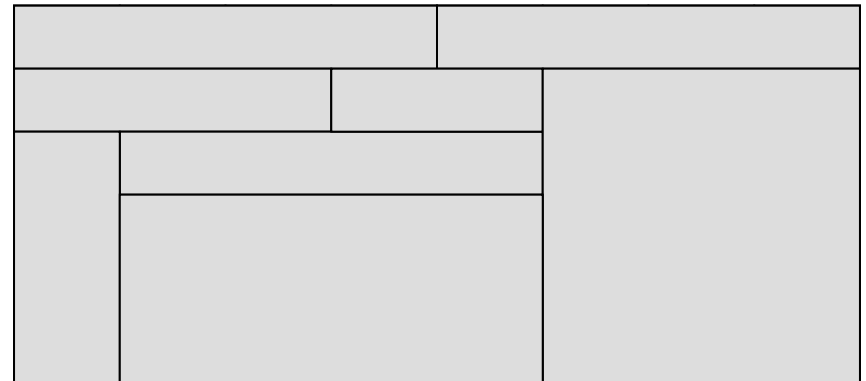
→ De forma genérica, a criação do *layout grid bag* envolve em:

→ Criar um objeto *GridBagLayout* e defini-lo como o gerenciador de layout atual

→ Criar uma nova instância de *GridBagConstraints*

→ Informar ao gerenciador de *layout* sobre o componente e suas restrições

→ Acrescentar o componente ao contêiner





# Gerenciadores de *Layout*

- *GridBagLayout*

➡ A localização de cada componente é definida por uma instância de *GridBagConstraints*, que possui os seguintes parâmetros

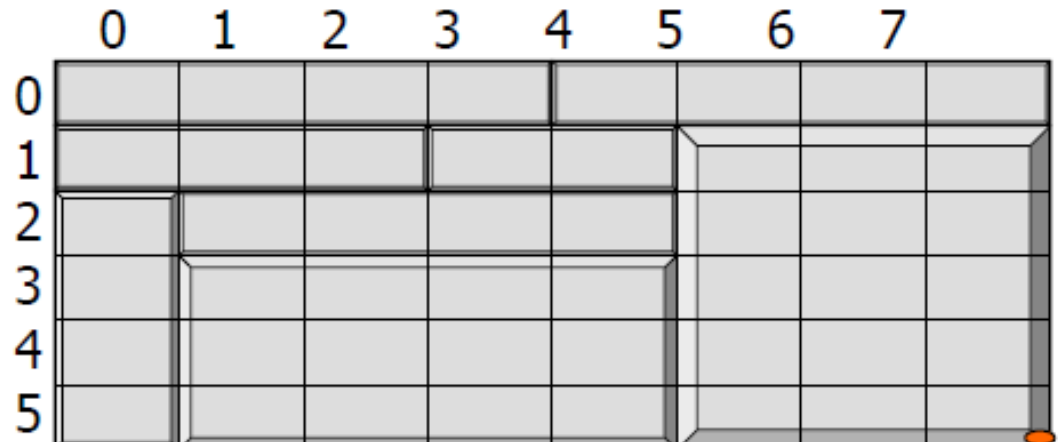
➡ *gridx*, *gridy*:

- São as coordenadas da células contendo o componente.
  - Ou seja, são as posições da matriz que representam o *grid*
  - Se o componente ocupar mais de uma célula, as coordenadas informadas deverão ser a da célula no canto superior esquerdo

➡ *gridwidth*, *gridheight*:

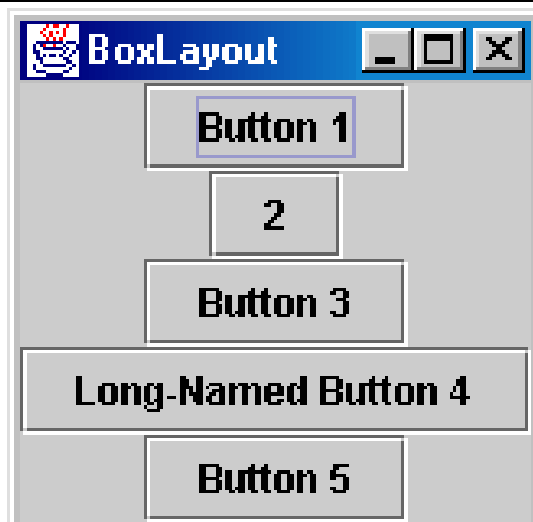
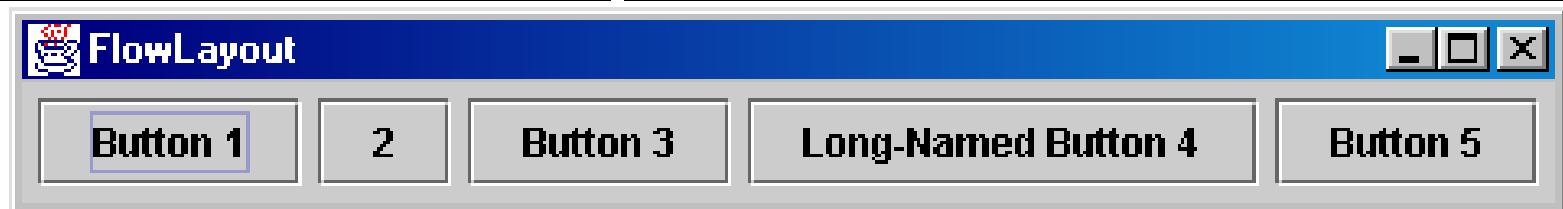
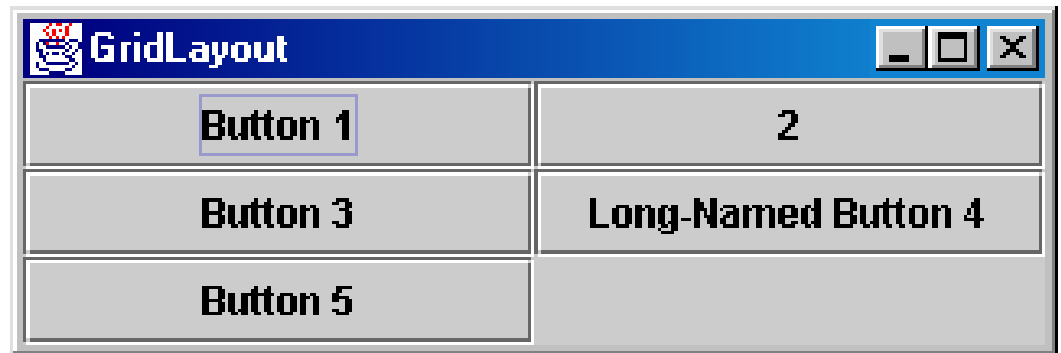
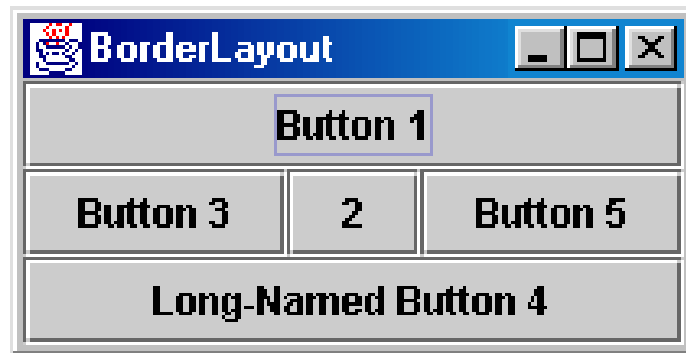
- Não são a altura ou largura da célula, mas sim o número de células em que o componente se espalha. O primeiro para colunas e o segundo para linhas

Depois que as restrições tiverem sido preparadas, basta usar o método `setConstraints(Componente, Restrição)`



# Gerenciadores de *Layout*

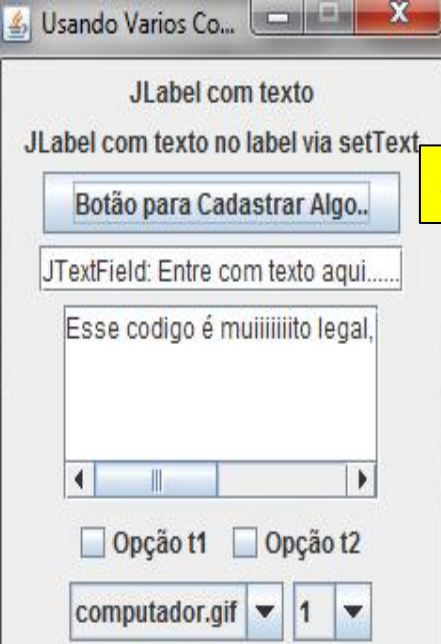
➔ Resumo Simplificado...



# Gerenciadores de *Layout*

antes

depois



```
1 import java.awt.*;  
2 import javax.swing.*;  
3  
4 public class VariosJuntos2 extends JFrame {  
5     private JLabel label1, label2;  
6     private JButton jButton1;  
7     private JTextField text1;  
8     private JTextArea comments;  
9     private JScrollPane scroll;  
10    private JCheckBox t1, t2;  
11  
12    private String nomeArquivo[] = { "computador.gif", "luz.gif" };  
13    private JComboBox nomes;  
14    private JComboBox mesesBox;  
15    private ImageIcon imagem;  
16  
17    private BoxLayout box;  
18  
19  
20    public VariosJuntos2() {  
21        super( "Usando Varios Componentes" );  
22  
23        JPanel pane = new JPanel();  
24  
25        box = new BoxLayout(pane, BoxLayout.Y_AXIS);  
26        pane.setLayout( box );  
27
```



# IMPORTANTE (Projeto)

## Misturando os gerenciadores de *layout*

- ➔ GUIs complexas exigem que cada componente seja colocado em uma localização exata.
- ➔ Estas GUIs consistem frequentemente em:
  - Múltiplos painéis com os componentes de cada painel organizados em um *layouts* específico.
  - ➔ Ou seja, vários contêineres dentro de um contêiner maior, sendo que todos tem o seu próprio gerenciador de *layout*
- ➔ Diferentes *layouts* podem ser especificados
  - Durante a construção ou
  - Através do método *setLayout( )*:
    - *BorderLayout*, *FlowLayout*, *GridLayout*.

# Caixas de Diálogo Padrão

➡ Pra fechar.. A classe *JOptionPane* oferece métodos que criam caixas de diálogos padrão – Janelas Modais “prontas”:

– As quatro caixas de diálogos padrão são as seguintes:

➡ Janela de Confirmação

– *showConfirmDialog (...)*

- Exibe a mensagem e busca confirmação com respostas Yes/No/Cancel

➡ Janela de Entrada de texto

– *showInputDialog (...)*

- Exibe a mensagem e busca uma linha de texto do usuário

➡ Janela de Mensagem

– *showMessageDialog (...)*

- Exibe a mensagem e aguarda o OK

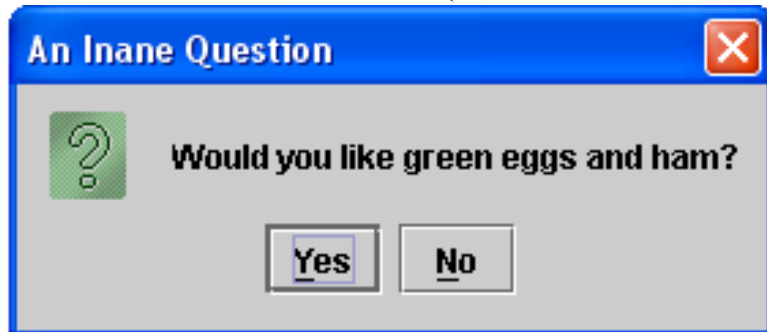
➡ Janela Genérica de Diálogo ou de Opção

– *showOptionDialog (...)*

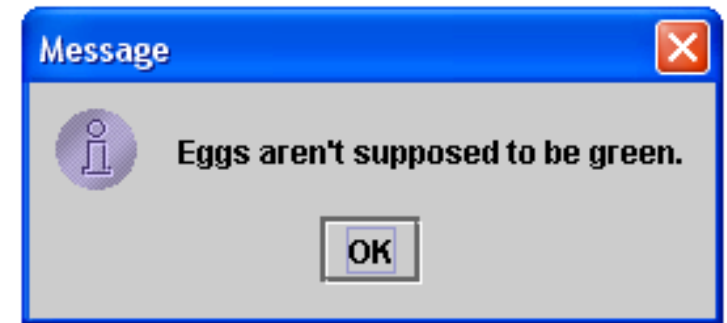
- Exibe a mensagem e busca a opção do usuário, dentre um conjunto de possibilidade

# Caixas de Diálogo Padrão

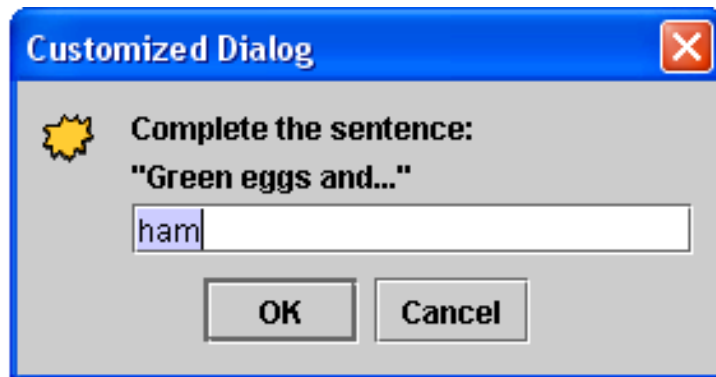
Janela de  
Confirmação



Janela de  
Mensagem



Janela de  
Entrada de  
texto



Janela Genérica de  
Diálogo

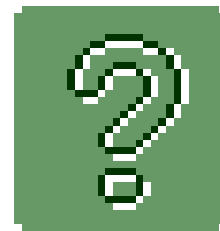
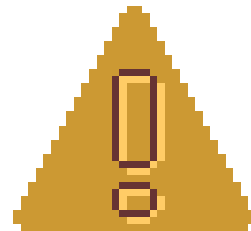
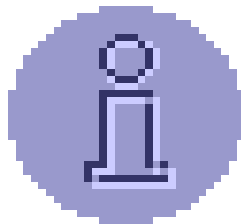
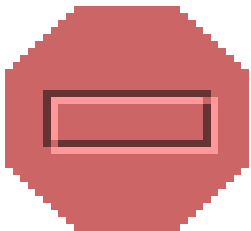


# Caixas de Diálogo Padrão

## → Características

### → O tipo de mensagens determinam o ícone

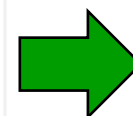
- **ERROR\_MESSAGE** - erro
- **INFORMATION\_MESSAGE** - informação
- **WARNING\_MESSAGE** - advertência
- **QUESTION\_MESSAGE** - confirmação
- **PLAIN\_MESSAGE** - esclarecimento



# Caixas de Diálogo Padrão

- O que o código abaixo está fazendo?

```
1 import java.awt.GridLayout;
2 import javax.swing.*;
3
4 public class Informacoes extends JFrame {
5     private JLabel titleLabel = new JLabel("Titulo: ", SwingConstants.RIGHT);
6     private JTextField title;
7     private JLabel addressLabel = new JLabel("Endereco: ", SwingConstants.RIGHT);
8     private JTextField address;
9     private JLabel typeLabel = new JLabel("Tipo: ", SwingConstants.RIGHT);
10    private JTextField type;
11
12    public Informacoes() {
13        super("Informacao do Site");
14
15        // Nome do site
16        String resposta1 = JOptionPane.showInputDialog(null, "Entre com o titulo do Site:");
17        title = new JTextField(resposta1, 20);
18
19        // Endereço do Site
20        String resposta2 = JOptionPane.showInputDialog(null, "Entre com o endereco do Site :");
21        address = new JTextField(resposta2, 20);
22
23        // Tipo de Site
24        String[] choices = { "Pessoal", "Comercial", "Desconhecido" };
25        int response3 = JOptionPane.showOptionDialog(null,
26            "Que tipo de site eh esse?",
27            "Tipo de Site",
28            0,
29            JOptionPane.QUESTION_MESSAGE,
30            null,
31            choices,
32            choices[0]);
33        type = new JTextField(choices[response3], 20);
34    }
```

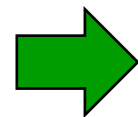




# Caixas de Diálogo Padrão

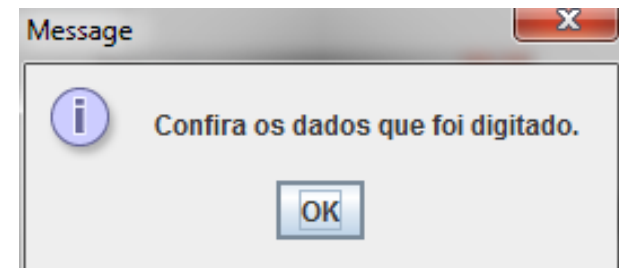
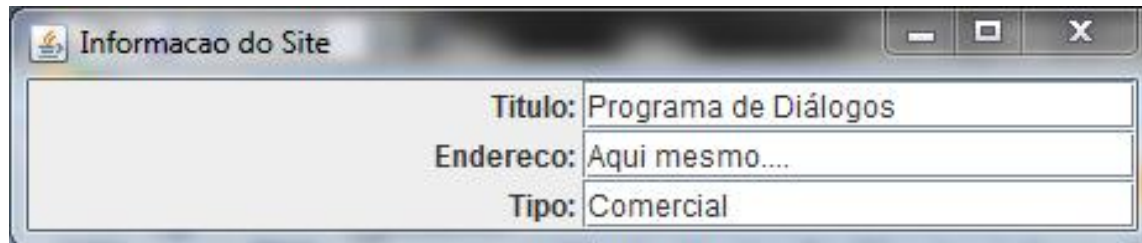
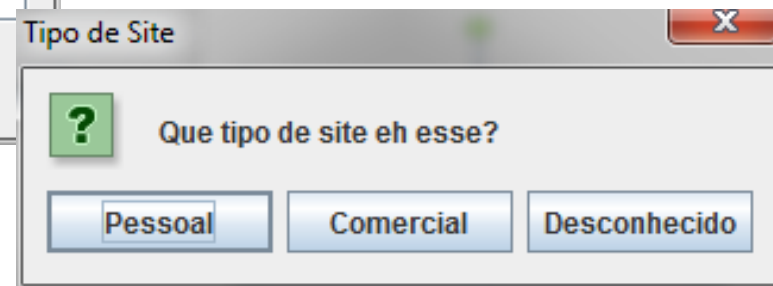
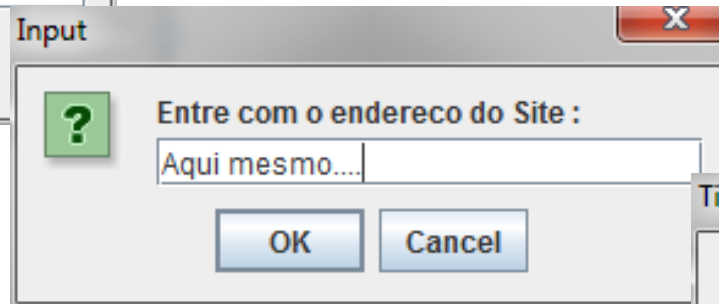
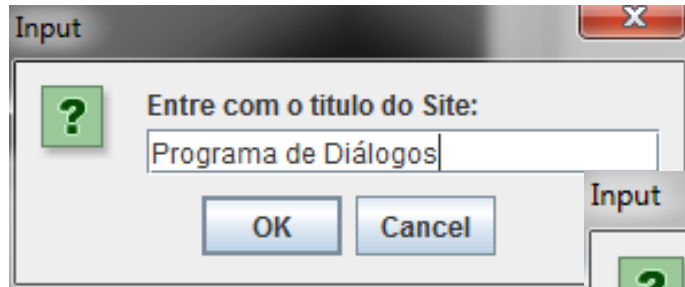
- O que o código abaixo está fazendo?

```
34
35     JPanel pane = new JPanel();
36     pane.setLayout(new GridLayout(3, 2));
37     pane.add(titleLabel);
38     pane.add(title);
39     pane.add(addressLabel);
40     pane.add(address);
41     pane.add(typeLabel);
42     pane.add(type);
43     this.getContentPane().add( pane );
44
45     pack();
46     setVisible(true);
47 }
48
49 public static void main(String[] arguments) {
50     Informacoes frame = new Informacoes();
51
52     JOptionPane.showMessageDialog( null, "Confira os dados que foi digitado.");
53     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
54
55 }
56 }
```



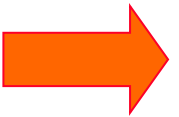
# Caixas de Diálogo Padrão

- O que o código abaixo está fazendo?



# Exercícios

- Blz... Agora é hora de exercitar.....
- Tente resolver ou analisar os seguintes problemas...
  - Em dupla
  - Apresentar ao professor no final da aula



# Exercício 1

- Vamos juntos abrir o NetBeans e criar um Projeto Java com JFrame para suporte ao Swing, usar alguns componentes e executar..

The screenshot shows the NetBeans IDE interface during the creation of a new Java project. The 'Choose File Type' dialog is open, showing the 'Project: Bingo' dropdown and a list of categories and file types. The 'JFrame Form' file type is selected. The 'NewJFrame.java' file is open in the editor, showing the following code:

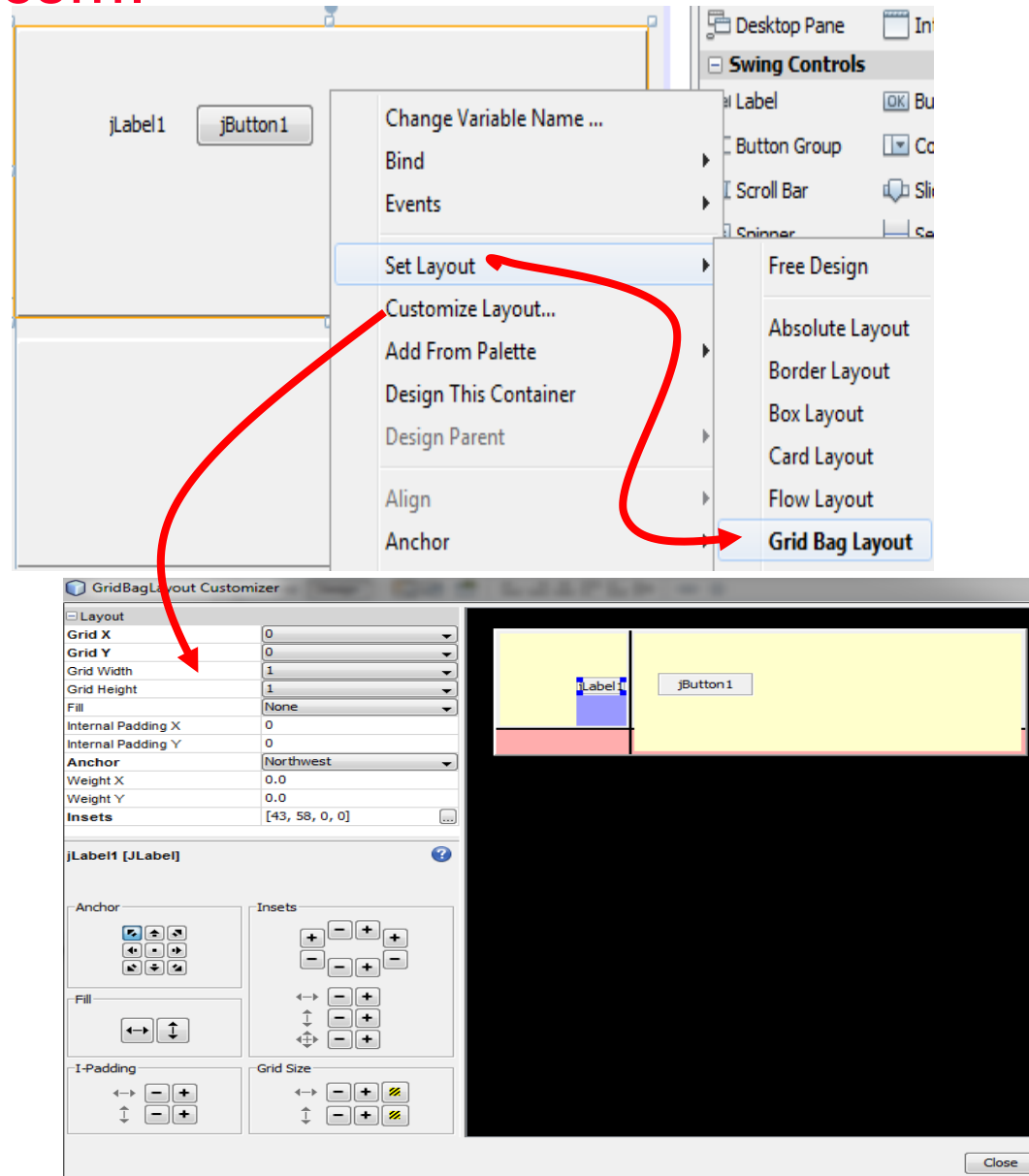
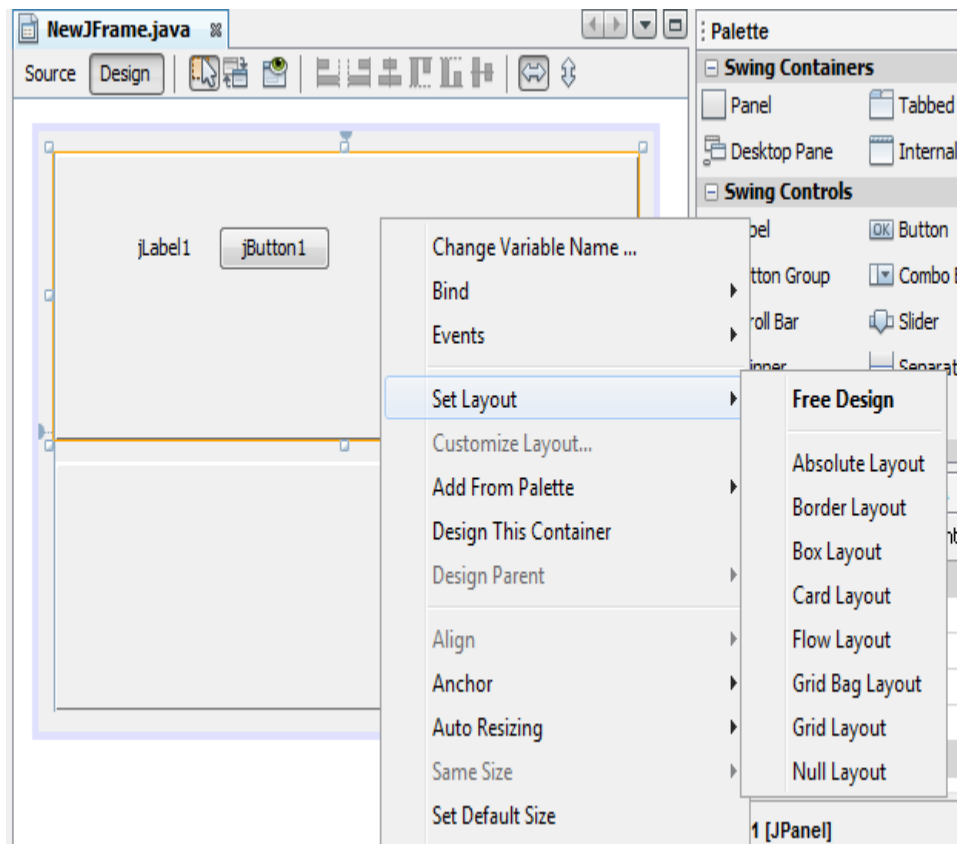
```
public class NewJFrame extends javax.swing.JFrame {  
  
    /** Creates new form NewJFrame */  
    public NewJFrame() {  
        initComponents();  
    }  
}
```

The 'Palette' window on the right shows the 'Swing Containers' and 'Swing Controls' sections. The 'Panel' component is selected in the 'Swing Containers' section. The 'Properties' window at the bottom right shows the 'defaultCloseOperation' property set to 'EXIT\_ON\_CLOSE'.

A red arrow points from the 'Panel' component in the 'Swing Containers' section of the 'Palette' window to the 'NewJFrame.java' file in the editor. A green arrow points to the 'EXIT\_ON\_CLOSE' dropdown menu in the 'Properties' window.

# Exercício 1

- “Brincando” com os Gerenciadores....



# Exercício 2

- Explore as potencialidades da interface Swing nestes exemplos...
  - Java2Demo.jar
  - laffy.jar
  - SwingSet2.jar
  - SwingSet3.jar

# Exercício 3

- Crie um projeto no NetBeans e execute os códigos presentes no .zip abaixo, em especial nos que foram apresentados em sala de aula...
  - `codigos_swing_parte_1.zip`
- “Brinque” com os códigos, modifique, veja o que acontece, etc.....

# Exercício 4

- Elabore a seguinte interface ao lado

- Neste momento pode utilizar ferramentas de desenhos

- Desafio:

- SETAR OS GERENCIADORES DE LAYOUTS PARA OS PADRÕES, EM VEZ DO DESENHO LIVRE

- Não é necessário implementar as funcionalidades

