Interface Gráfica

Prof. Dr. Lucas C. Ribas

Disciplina: Programação Orientada a Objetos

Departamento de Ciências de Computação e Estatística

unesp
UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"

IBILCE / UNESP - CÂMPUS DE SÃO JOSÉ DO RIO PRETO

Agenda

- Introdução
- Componentes Swing
- Gerenciadores de Layout
- Look and Feel
- GUI e NetBeans

unesp

Introdução

- Uma interface gráfica de usuário (GUI) é uma forma amigável do usuário interagir com as funcionalidade de um aplicativo
- A API Java oferece alguns frameworks para trabalhar com GUIs
  - AWT
  - Swing
  - JavaFX
- Trataremos nesta aula principalmente dos componentes **Swing**

unesp

Introdução

- A API do Swing é composta por 18 pacotes
- Porém, a maioria das aplicações se concentra no uso das classes e interfaces dos pacotes
  - javax.swing
  - javax.swing.event

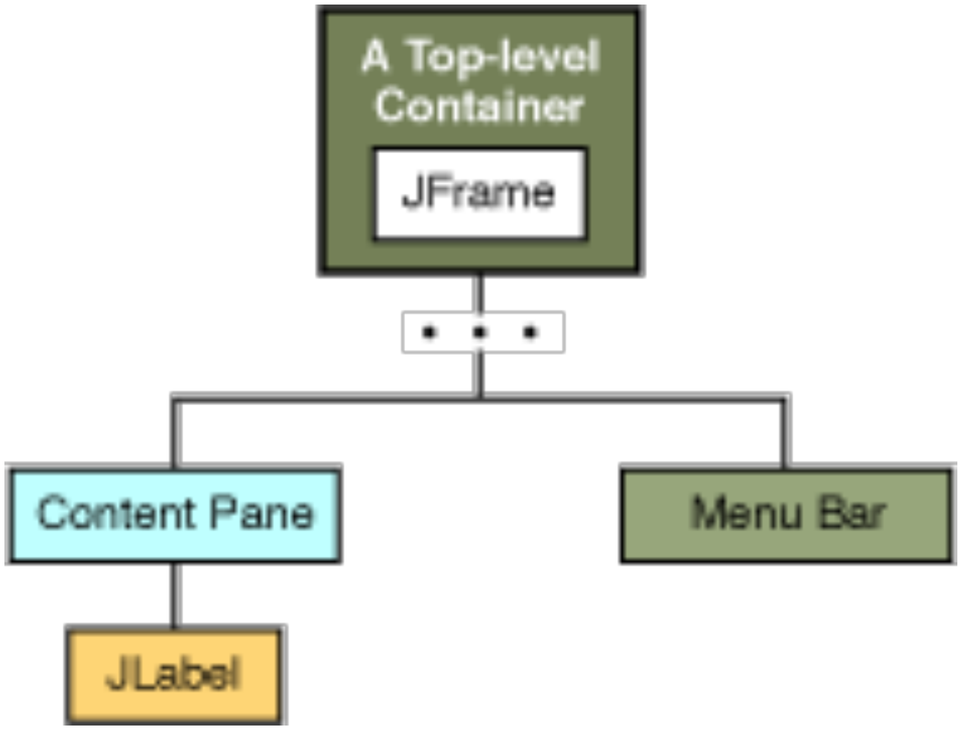
unesp

Componentes Swing

unesp

Componentes Swing

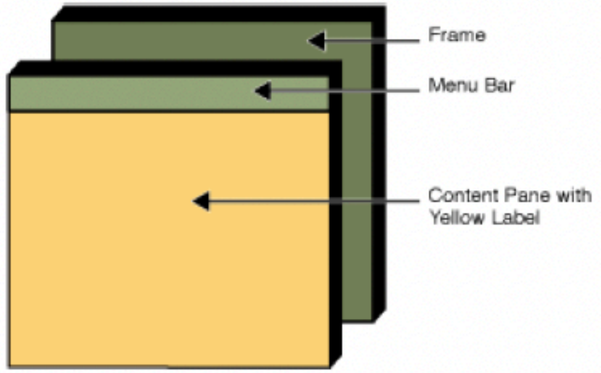
- Swing possui três componentes *top-level*, no qual todos os outros elementos devem ser inseridos
  - JFrame
  - JDialog
  - JApplet
- Uma GUI deve ter pelo menos um *top-level container*




```
frame.getContentPane().add(yellowLabel, BorderLayout.CENTER);
```

Componentes Swing

- Swing possui três componentes *top-level*, no qual todos os outros elementos devem ser inseridos
  - JFrame
  - JDialog
  - JApplet
- Uma GUI deve ter pelo menos um *top-level container*



O método **add()** do JFrame adiciona no *content pane*.

Componentes Swing - Hello World

```
import javax.swing.SwingUtilities;
import javax.swing.JFrame;

public class SwingHelloWorld {

    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                createAndShowGUI();
            }
        });
    }

    private static void createAndShowGUI() {
        JFrame f = new JFrame("Swing Hello World!");
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setSize(250,250);
        f.setVisible(true);
    }
}
```

unesp

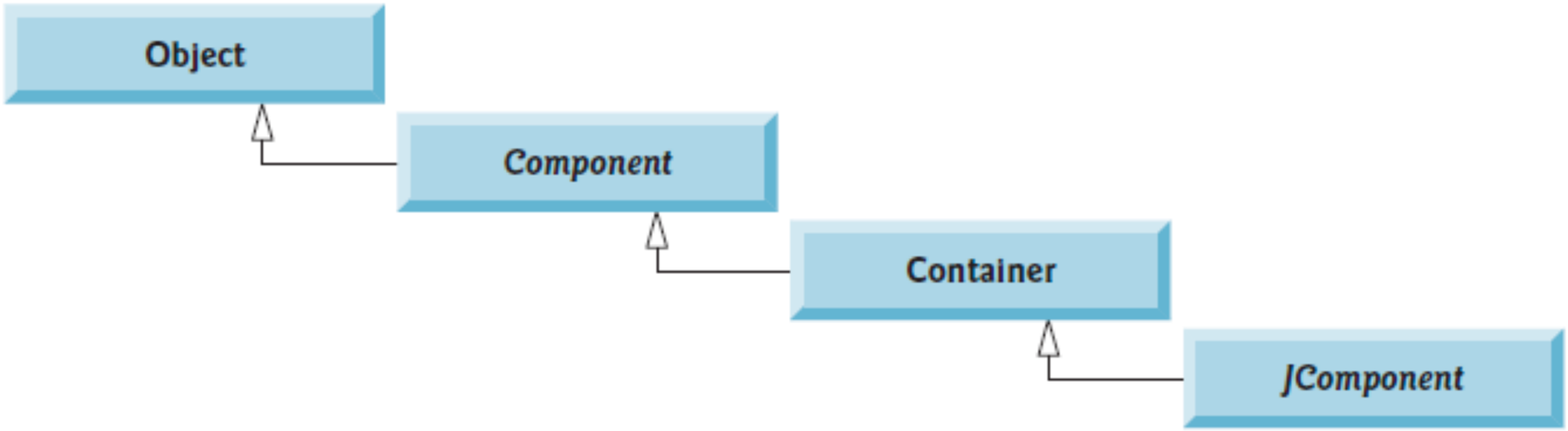
Componentes Swing - Hello World

- Em Java Swing, todas as operações relacionadas à interface gráfica devem ser executadas no thread de despacho de eventos do Swing, também conhecido como Event Dispatch Thread (EDT)
- O EDT é responsável por lidar com eventos de interface do usuário, como ações do mouse e do teclado, atualizações de componentes Swing e redimensionamento de janelas adequadamente
- A chamada **SwingUtilities.invokeLater(new Runnable() { ... })** é uma maneira de garantir que o código relacionado à criação e exibição da GUI Swing seja executado no EDT
- Se você executar o código sem usar **SwingUtilities.invokeLater(...),** pode encontrar problemas relacionados à concorrência e à atualização da interface do usuário

unesp

Componentes Swing

- Todos os componentes Swing (exceto os *top-level*) são descendentes da classe **JComponent**
- **Container** e **Component** são do pacote AWT



Componentes Swing

- Principais componentes do Swing
  - **JLabel**: exibe texto/ícones não editáveis
  - **JTextField**: campo de texto editável
    - JTextArea, JPasswordField
  - **JButton**: botão
  - **JRadioButton**: botão de seleção (uma única seleção é permitida em um grupo)
  - **JCheckBox**: caixa de seleção (múltiplas)
  - **JComboBox**: lista *drop-down* de opções

unesp

Componentes Swing

- Principais componentes do Swing
  - **JList**: lista de itens selecionáveis
  - **JTable**: tabela
  - **JPanel**: umá área em que componentes podem ser colocados e organizados
    - Existem vários tipos
  - **JFileChooser:** componente que navega pelos diretórios e retorna um arquivo (abrir ou salvar)
  - **JOptionPane:** janela de diálogo para mostrar informações ou ler entradas simples

unesp

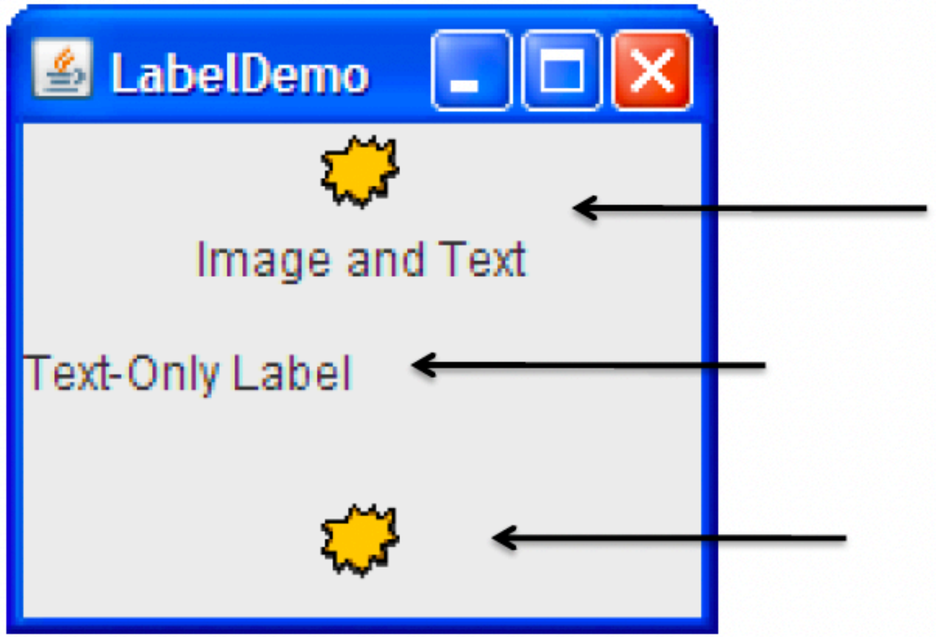
JLabel

- Permite mostrar texto e imagens
- É possível ajustar cores e fontes
  - Também aceita HTML

```
ImageIcon icon = createImageIcon("images/middle.gif");

JLabel label1, label2, label3;
label1 = new JLabel("Image and Text", icon, JLabel.CENTER);
//Set the position of the text, relative to the icon:
label1.setVerticalTextPosition(JLabel.BOTTOM);
label1.setHorizontalTextPosition(JLabel.CENTER);

label2 = new JLabel("Text-Only Label");
label3 = new JLabel(icon);

f.add(jlabel1); f.add(jlabel2); f.add(jlabel3);
```

JLabel

- Permite mostrar texto e imagens
- É possível ajustar cores e fontes
  - Também aceita HTML



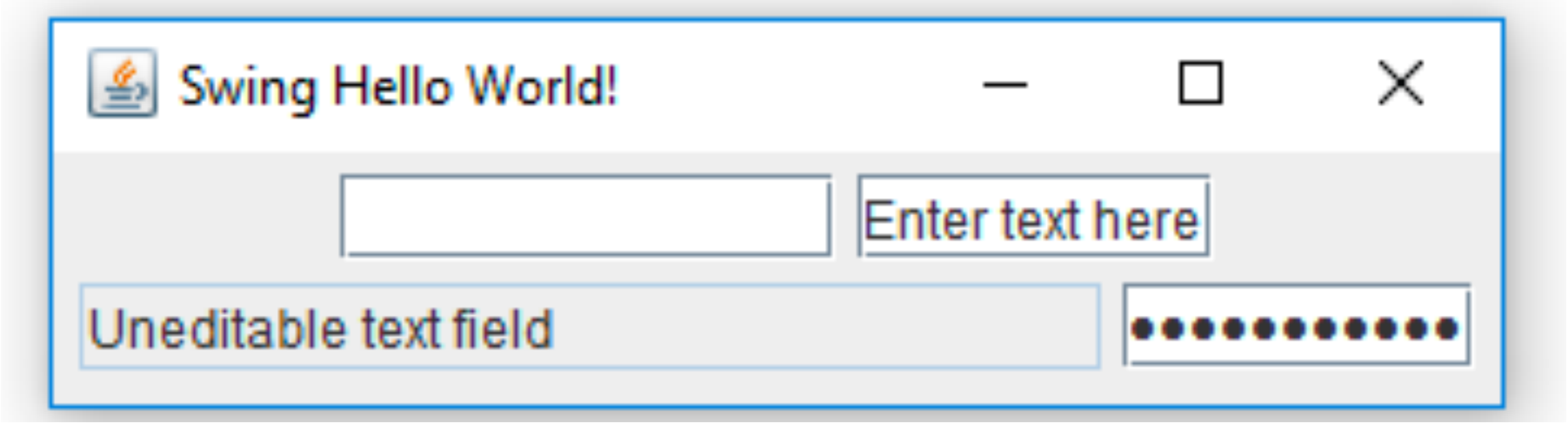
JTextField

- Entrada de texto pequena (uma linha)
  - Outros: JPasswordField, JFormattedTextField
- Use **JTextArea** para textos longos (várias linhas)

```
f.setLayout(new FlowLayout());

JTextField tField1, tField2, tField3;
tField1 = new JTextField(10);  // size (columns)
tField2 = new JTextField("Enter text here");
tField3 = new JTextField("Uneditable text field",21);
tField3.setEditable(false);

f.add(tField1); f.add(tField2); f.add(tField3);

JPasswordField passField = new JPasswordField("Hidden text");
f.add(passField);
```

JTextField

- Entrada de texto pequena (uma linha)
  - Outros: JPasswordField, JFormattedTextField
- Use **JTextArea** para textos longos (várias linhas)



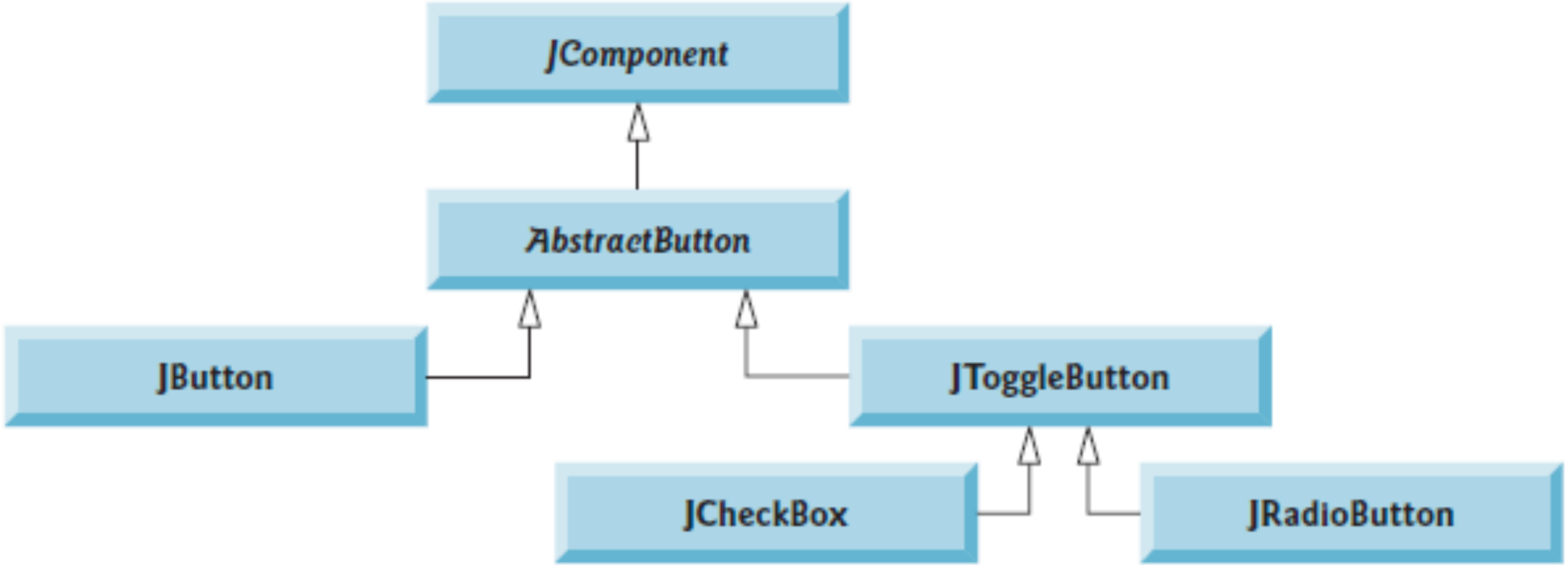
JTextField

- Leitura do texto
  - String getText()
  - char[] getPassword()

```
String name = tField1.getText();

char[] pass;
pass = passField.getPassword();
```

unesp

Botões


JComponent
AbstractButton
JButton
JToggleButton
JCheckBox
JRadioButton

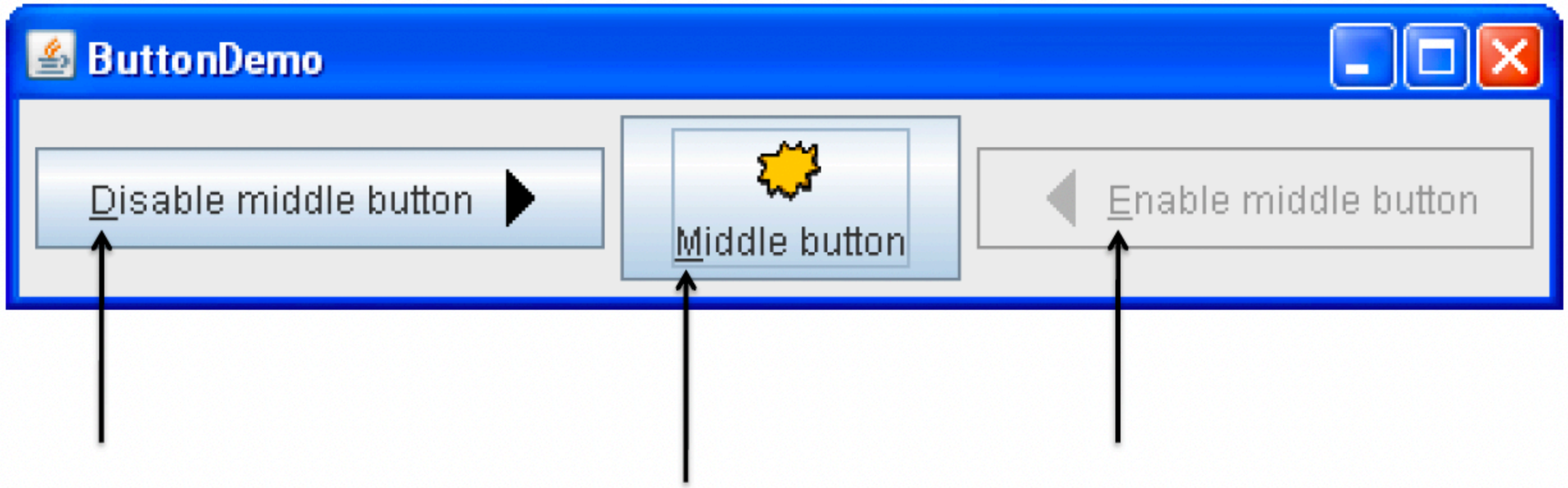
JButton


ButtonDemo
Disable middle button
Middle button
Enable middle button

JButton

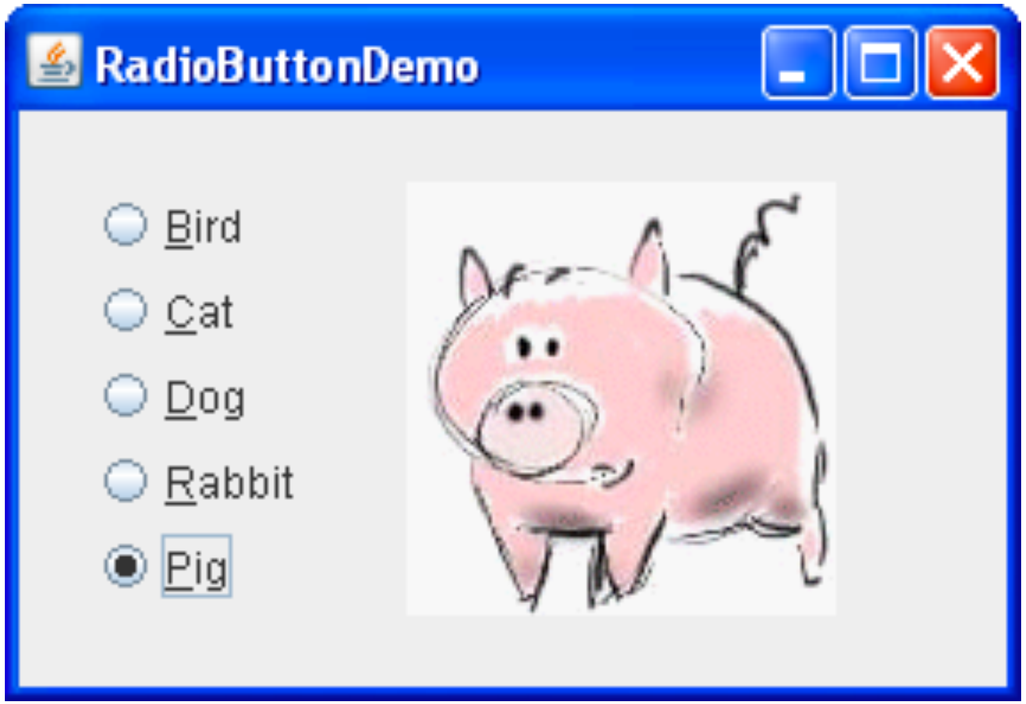
```
ImageIcon leftButtonIcon = createImageIcon("img/right.gif");
ImageIcon middleButtonIcon = createImageIcon("img/middle.gif");
ImageIcon rightButtonIcon = createImageIcon("img/left.gif");

JButton b1, b2, b3;

b1 = new JButton("Disable middle button", leftButtonIcon);
b1.setVerticalTextPosition(AbstractButton.CENTER);
b1.setHorizontalTextPosition(AbstractButton.LEADING);
b1.setMnemonic(KeyEvent.VK_D);

b2 = new JButton("Middle button", middleButtonIcon);
b2.setVerticalTextPosition(AbstractButton.BOTTOM);
b2.setHorizontalTextPosition(AbstractButton.CENTER);
b2.setMnemonic(KeyEvent.VK_M);

b3 = new JButton("Enable middle button", rightButtonIcon);
b3.setMnemonic(KeyEvent.VK_E);
b3.setEnabled(false);

f.add(b1); f.add(b2); f.add(b3);
```

JButton



Mnemonics (Alt + Tecla)

JRadioButton



JRadioButton

```
JRadioButton birdButton = new JRadioButton("Bird");
birdButton.setMnemonic(KeyEvent.VK_B);

JRadioButton catButton = new JRadioButton("Cat");
catButton.setMnemonic(KeyEvent.VK_C);

JRadioButton dogButton = new JRadioButton("Dog");
dogButton.setMnemonic(KeyEvent.VK_D);

JRadioButton rabbitButton = new JRadioButton("Rabbit");
rabbitButton.setMnemonic(KeyEvent.VK_R);

JRadioButton pigButton = new JRadioButton("Pig");
pigButton.setMnemonic(KeyEvent.VK_P);
pigButton.setSelected(true);

f.add(birdButton);
f.add(catButton);
f.add(rabbitButton);
f.add(pigButton);
```

JRadioButton

- A ideia do **JRadioButton** é permitir apenas uma seleção dentre um grupo de opções
  - Adicionamos em um **ButtonGroup**

```
//Group the radio buttons.
ButtonGroup radioGroup = new ButtonGroup();
radioGroup.add(birdButton);
radioGroup.add(catButton);
radioGroup.add(dogButton);
radioGroup.add(rabbitButton);
radioGroup.add(pigButton);
```

unesp

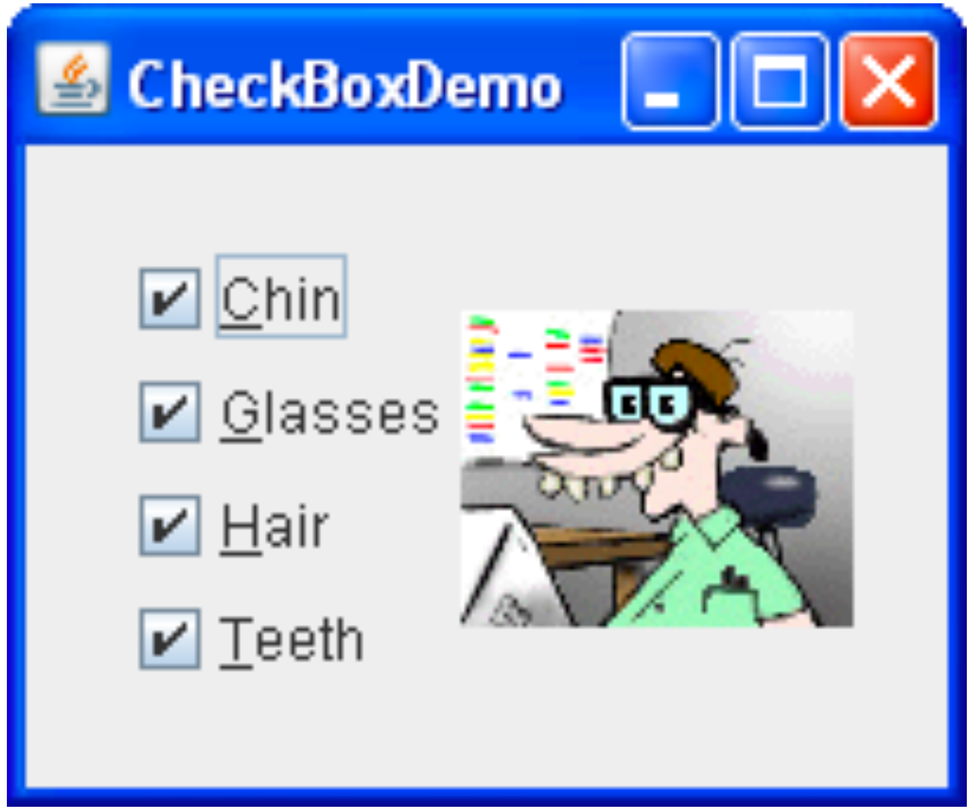
JRadioButton

- Verificando se está selecionado
  - boolean isSelected()
- Limpando seleções
  - void clearSelection() [ButtonGroup]

```
if(birdButton.isSelected())
    System.out.println("Bird button selected");
else if(catButton.isSelected())
    System.out.println("Cat button selected");
else if(dogButton.isSelected())
    System.out.println("Dog button selected");
else if(rabbitButton.isSelected())
    System.out.println("Rabbit button selected");
else if(pigButton.isSelected())
    System.out.println("Pig button selected");

radioGroup.clearSelection();
```

JCheckBox

-



JCheckBox

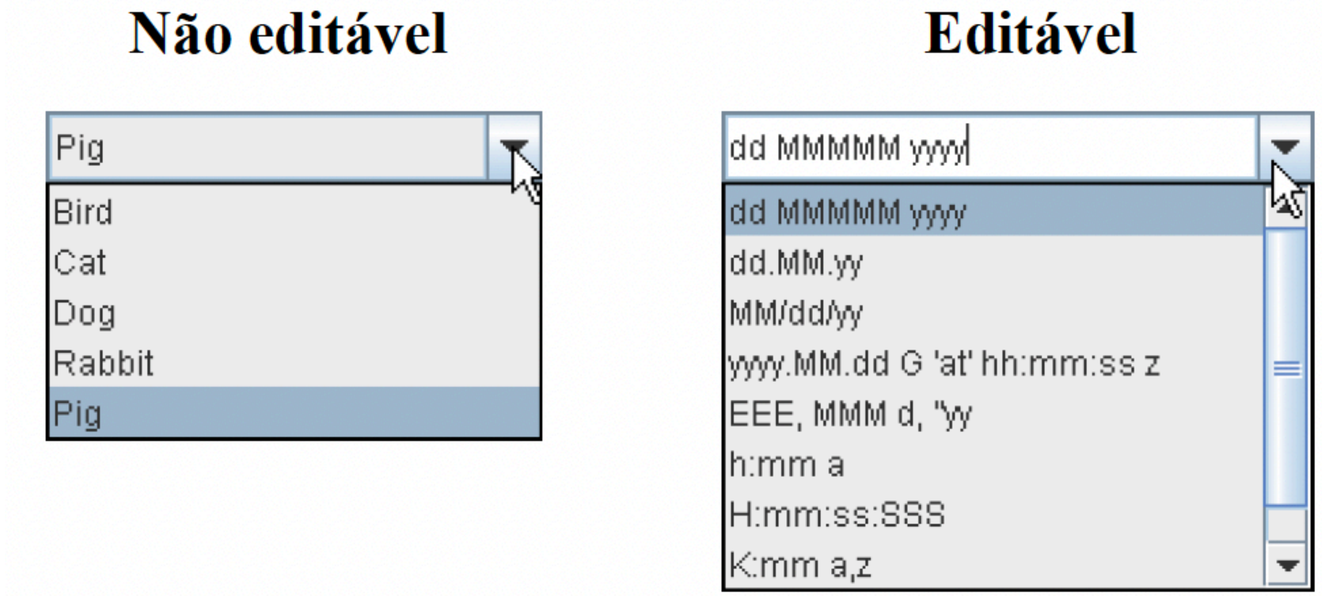
-

```
JCheckBox chinButton = new JCheckBox("Chin");
chinButton.setMnemonic(KeyEvent.VK_C);
chinButton.setSelected(true);

JCheckBox glassesButton = new JCheckBox("Glasses");
glassesButton.setMnemonic(KeyEvent.VK_G);
glassesButton.setSelected(true);

JCheckBox hairButton = new JCheckBox("Hair");
hairButton.setMnemonic(KeyEvent.VK_H);
hairButton.setSelected(true);

JCheckBox teethButton = new JCheckBox("Teeth");
teethButton.setMnemonic(KeyEvent.VK_T);
teethButton.setSelected(true);

f.add(chinButton);
f.add(glassesButton);
f.add(hairButton);
f.add(teethButton);
```

unesp

JComboBox

-


Não editável
Pig
Bird
Cat
Dog
Rabbit
Pig
Editável
dd MMMMM yyyy
dd MMMMM yyyy
dd.MM.yy
MM/dd/yy
yyyy.MM.dd G 'at' hh:mm:ss z
EEE, MMM d, 'yy
h:mm a
H:mm:ss:SSS
K:mm a,z

JComboBox

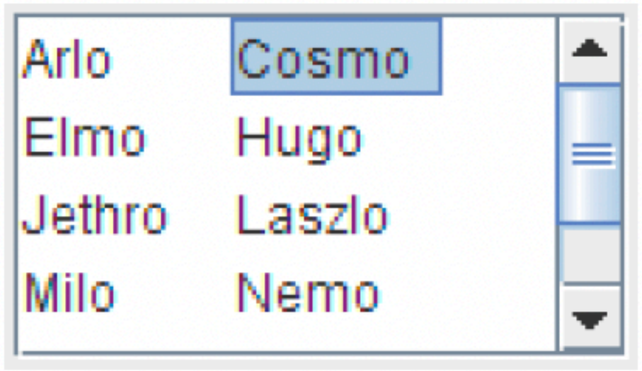
JComboBox

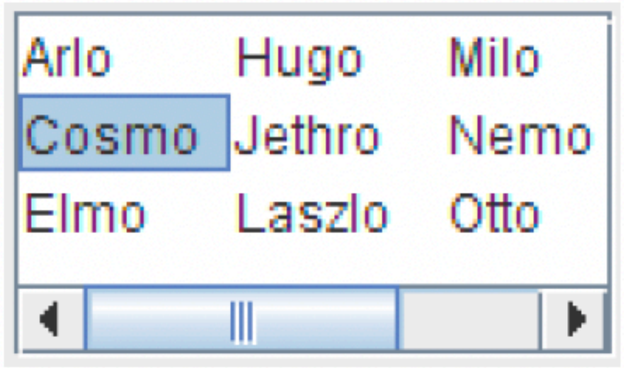
```
String[] petStrings = {"Bird", "Cat", "Dog", "Rabbit", "Pig" };

//Create the combo box, select item at index 4.
//Indices start at 0, so 4 specifies the pig.
JComboBox petCombo = new JComboBox(petStrings);
petCombo.setSelectedIndex(4);

f.add(petCombo);
```

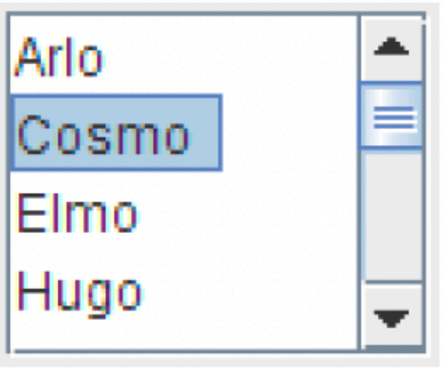
- Lendo o estado de um JComboBox

```
String petName = (String)petCombo.getSelectedItem();
```

JList


Arlo
Cosmo
Elmo
Hugo
Jethro
Laszlo
Milo
Nemo


Arlo
Hugo
Milo
Cosmo
Jethro
Nemo
Elmo
Laszlo
Otto


Arlo
Cosmo
Elmo
Hugo

JList

- Em geral, listas são colocadas dentro de painéis com rolagem (**JScrollPane**)
- Para criar listas mutáveis (inserção e remoção de ítens) é preciso trabalhar com um **ListModel**
  - **DefaultListModel**
- Através de **DefaultListModel**
  - Métodos para adição e remoção de ítens
- Através do **JList**
  - Métodos para obtenção de ítens selecionados

unesp

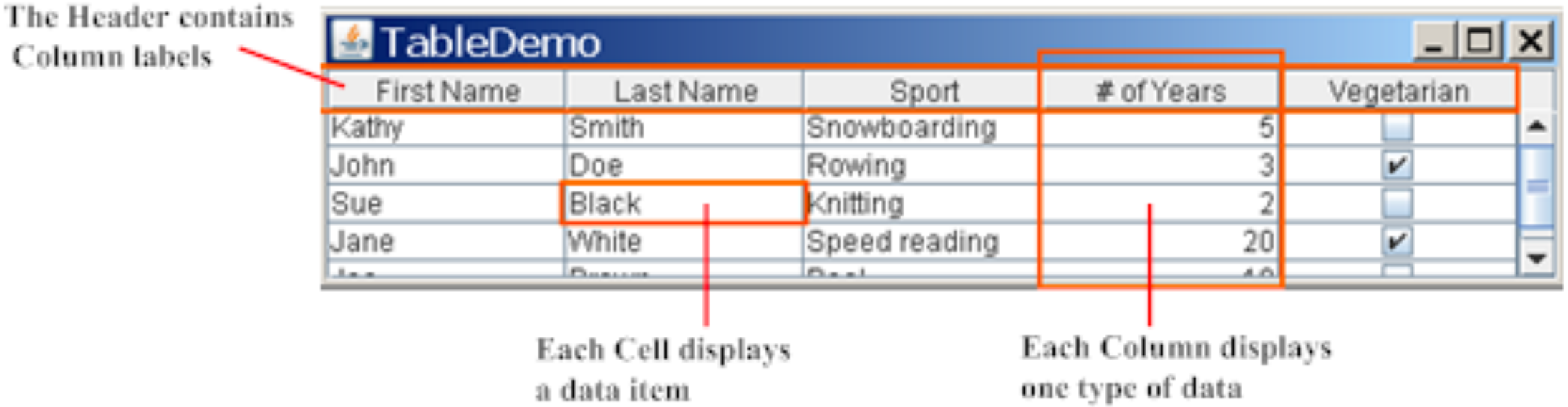
JList

```
DefaultListModel listModel = new DefaultListModel();
listModel.addElement("Jane Doe");
listModel.addElement("John Smith");
listModel.addElement("Kathy Green");

JList list = new JList(listModel);
list.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
list.setLayoutOrientation(JList.VERTICAL_WRAP);

f.add(list);
```

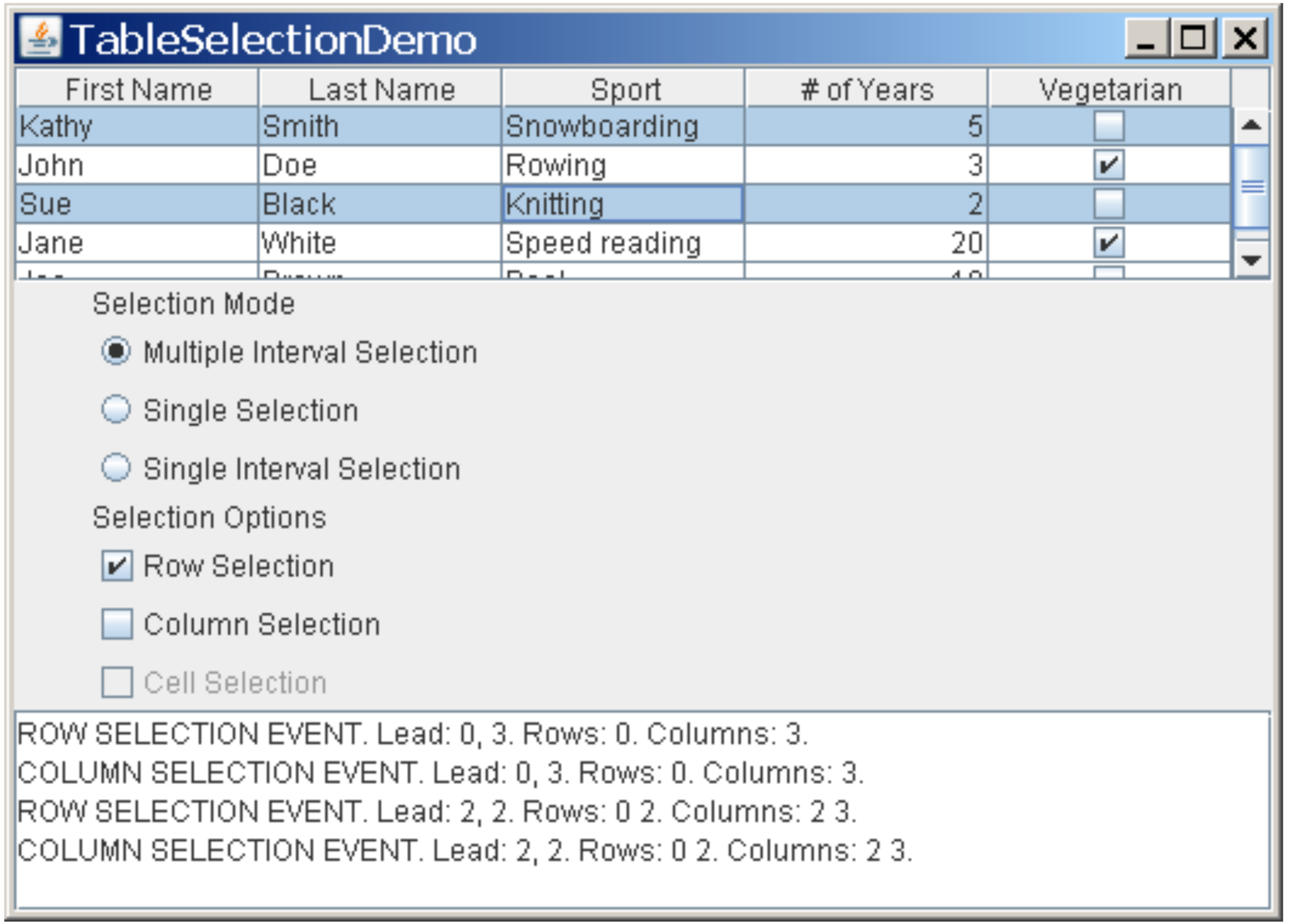
JScrollPane scrollPane = new JScrollPane(list);
f.add(scrollPane);

```
int index = list.getSelectedIndex();
listModel.remove(index);

System.out.println("List size: "+listModel.getSize());
```

JTable


The Header contains
Column labels
TableDemo
First Name
Last Name
Sport
# of Years
Vegetarian
Kathy
Smith
Snowboarding
5
John
Doe
Rowing
3
Sue
Black
Knitting
2
Jane
White
Speed reading
20
Each Cell displays
a data item
Each Column displays
one type of data

JTable


TableSelectionDemo
First Name
Last Name
Sport
# of Years
Vegetarian
Kathy
Smith
Snowboarding
5
John
Doe
Rowing
3
Sue
Black
Knitting
2
Jane
White
Speed reading
20
Selection Mode
Multiple Interval Selection
Single Selection
Single Interval Selection
Selection Options
Row Selection
Column Selection
Cell Selection
ROW SELECTION EVENT. Lead: 0, 3. Rows: 0. Columns: 3.
COLUMN SELECTION EVENT. Lead: 0, 3. Rows: 0. Columns: 3.
ROW SELECTION EVENT. Lead: 2, 2. Rows: 0 2. Columns: 2 3.
COLUMN SELECTION EVENT. Lead: 2, 2. Rows: 0 2. Columns: 2 3.

JTable

- Assim como a **JList**, uma **JTable:**
  - É geralmente colocada dentro de um painel de rolagem
    - JScrollPane
  - Possui um TableModel para manipulação dos dados
    - DefaultTableModel
  - Métodos de obtenção da seleção estão na própria **JTable**

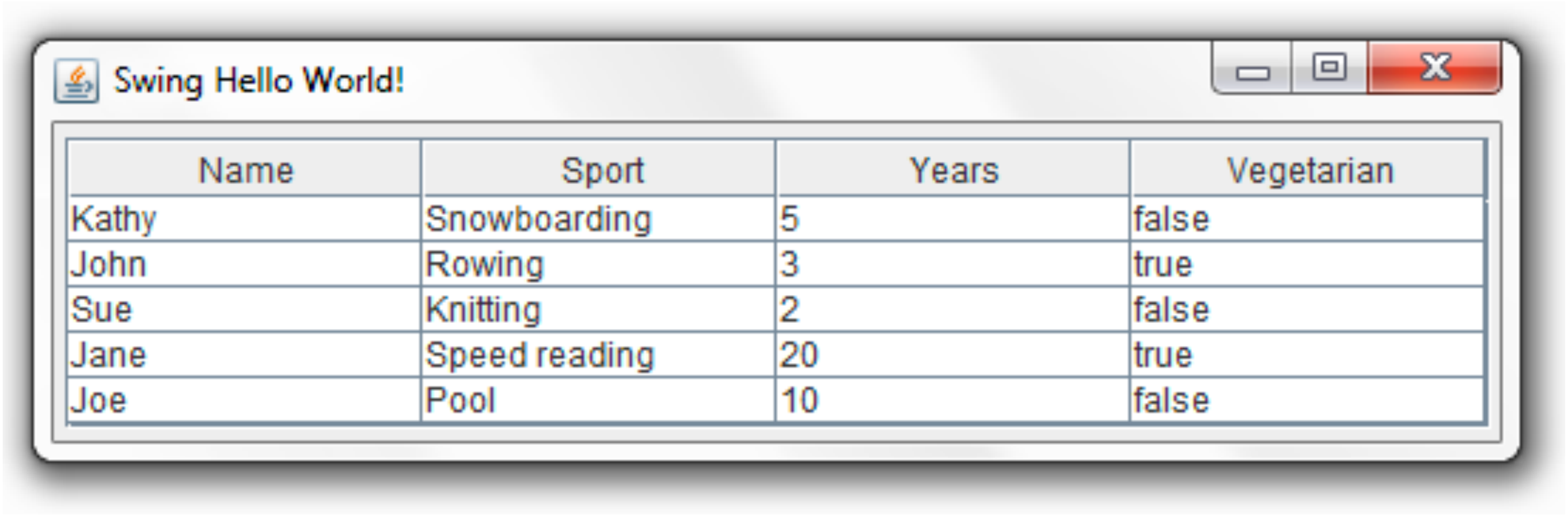
- Porém, é possível criar um **JTable** diretamente de uma matriz de objetos
  - Limitado: Ex. todos os elementos se tornam strings

unesp

JTable

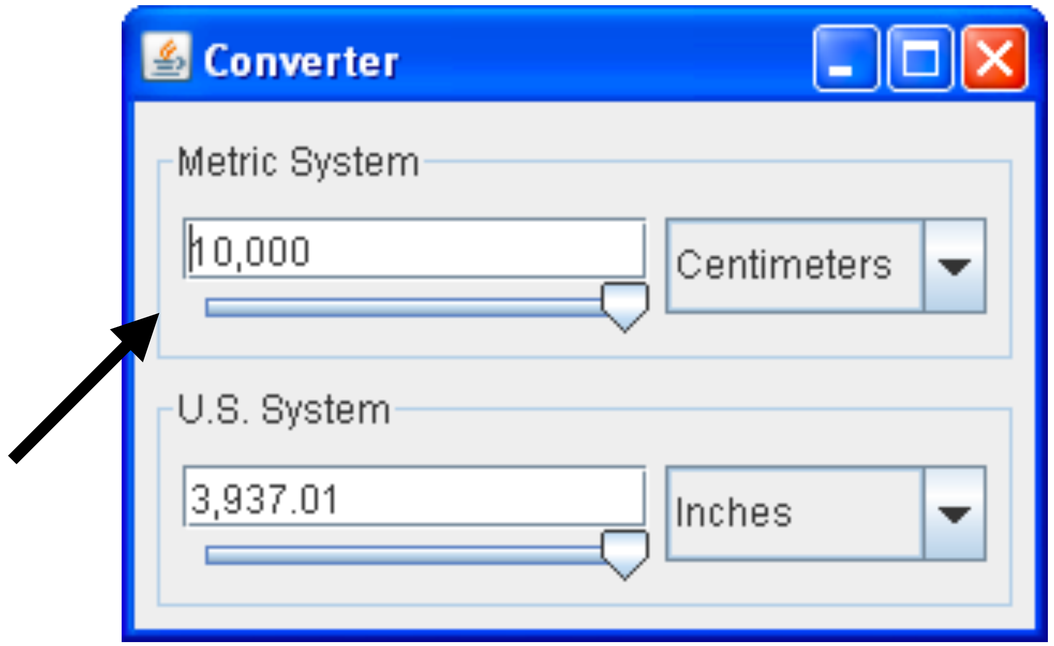
```
String[] colNames = {"Name“, "Sport“, “Years“, "Vegetarian"};

Object[][] data = {
  {"Kathy", "Snowboarding", new Integer(5), new Boolean(false)},
  {"John", "Rowing", new Integer(3), new Boolean(true)},
  {"Sue", "Knitting", new Integer(2), new Boolean(false)},
  {"Jane", "Speed reading", new Integer(20), new Boolean(true)},
  {"Joe", "Pool", new Integer(10), new Boolean(false)}};

JTable table = new JTable(data, colNames);
table.setPreferredScrollableViewportSize(new Dimension(500,80));
table.setFillsViewportHeight(true);

//Create the scroll pane and add the table to it.
JScrollPane scrollPane = new JScrollPane(table);

//Add the scroll pane to the JFrame.
f.add(scrollPane);
```

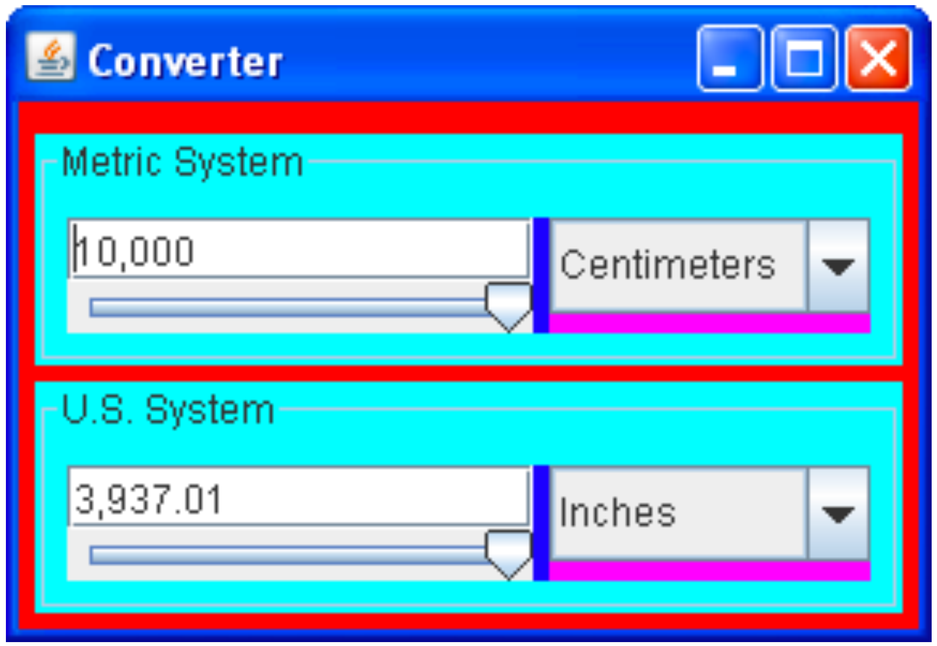
JTable


Swing Hello World!
Name
Sport
Years
Vegetarian
Kathy
Snowboarding
5
false
John
Rowing
3
true
Sue
Knitting
2
false
Jane
Speed reading
20
true
Joe
Pool
10
false

JPanel

- JPanel é um container para armazenar componentes
- Por padrão, podemos alterar sua cor de fundo e seu **tipo de borda**



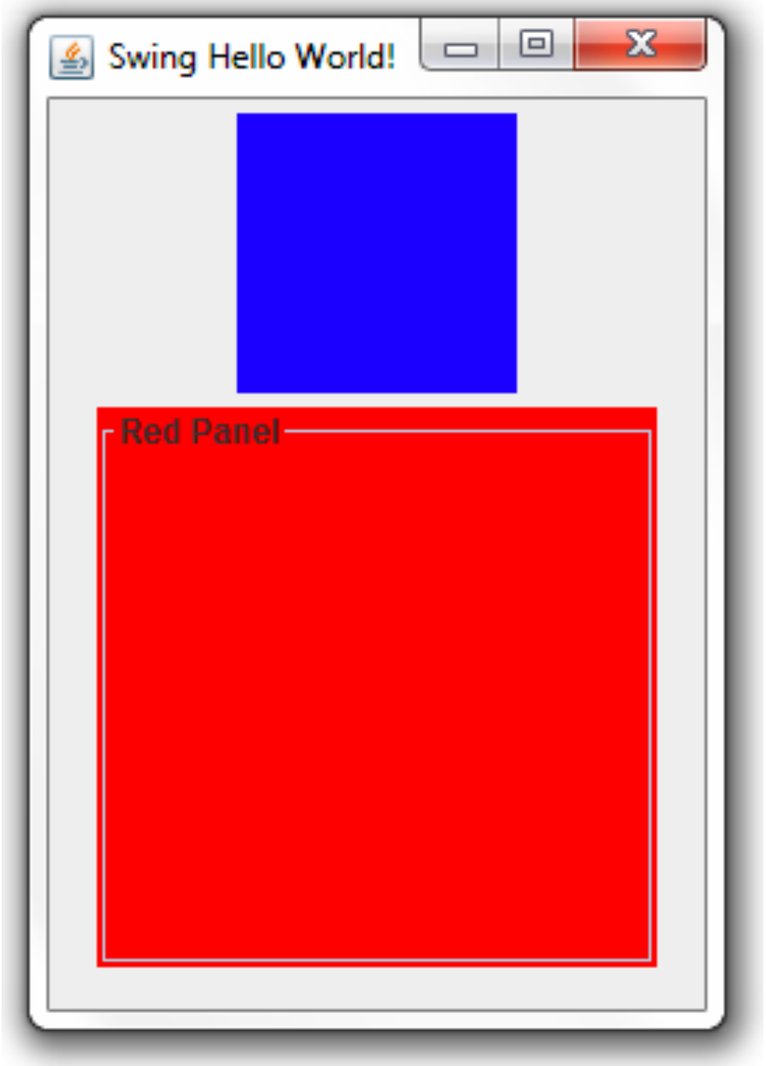


JPanel

- Outro exemplo

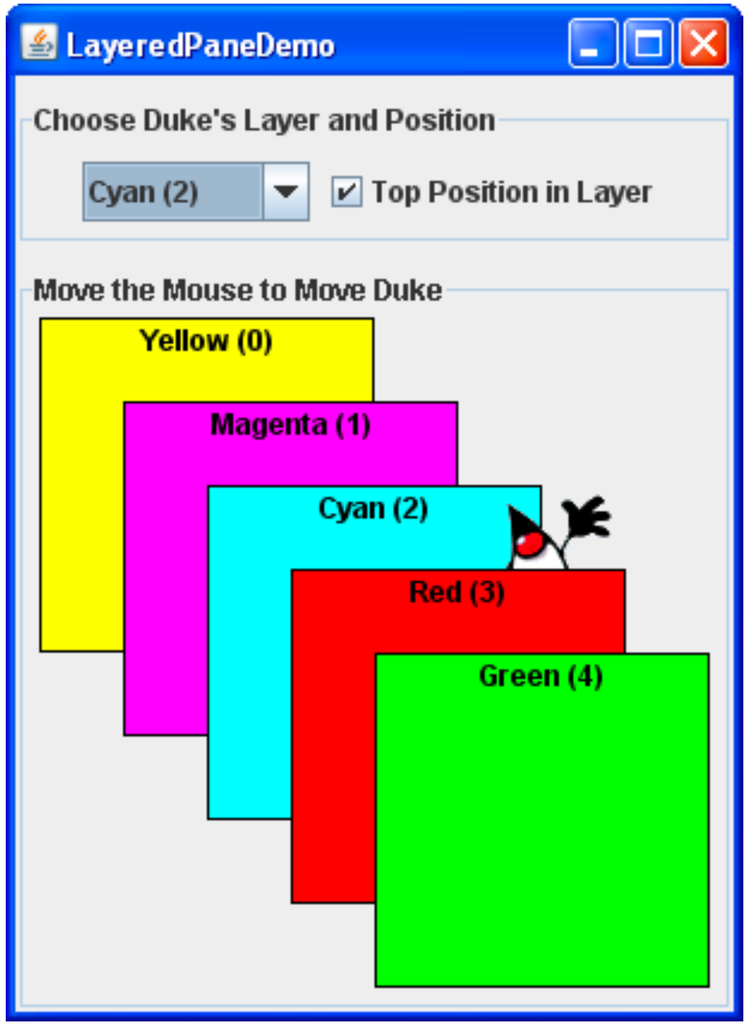
```
JPanel panel1 = new JPanel();
JPanel panel2 = new JPanel();

panel1.setBackground(Color.BLUE);
panel1.setPreferredSize(new Dimension(100,100));

panel2.setBackground(Color.RED);
panel2.setPreferredSize(new Dimension(200,200));
panel2.setBorder(BorderFactory.createTitledBorder("Red Panel"));

f.add(panel1);
f.add(panel2);
```

JPanel



JPanel

- Existem diversos tipos de painéis
  - JLayeredPane

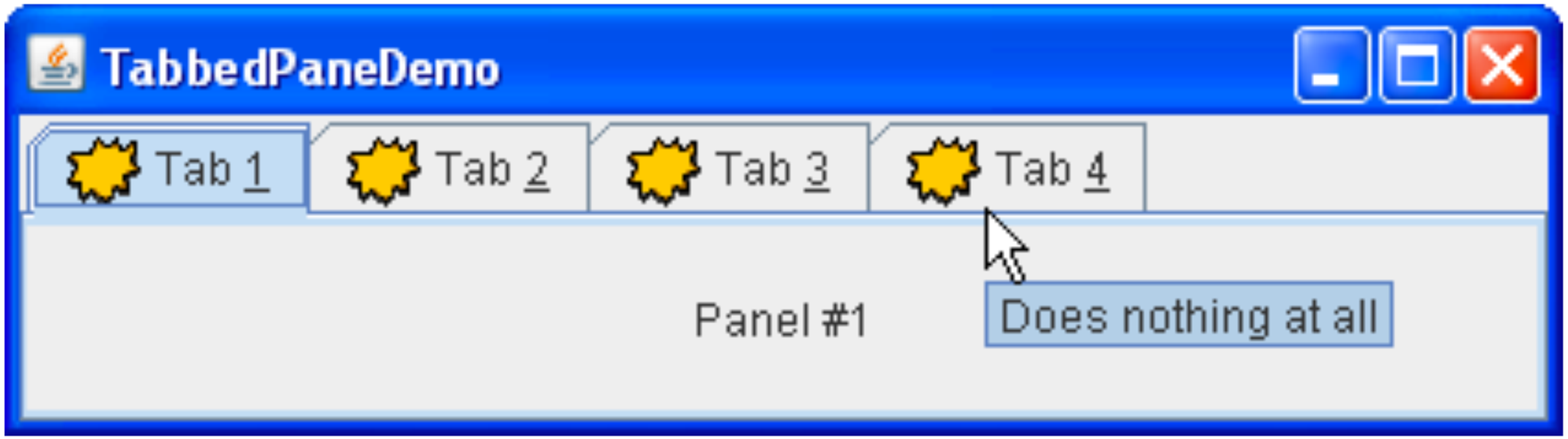


JPanel

- Existem diversos tipos de painéis
  - JScrollPane

JPanel

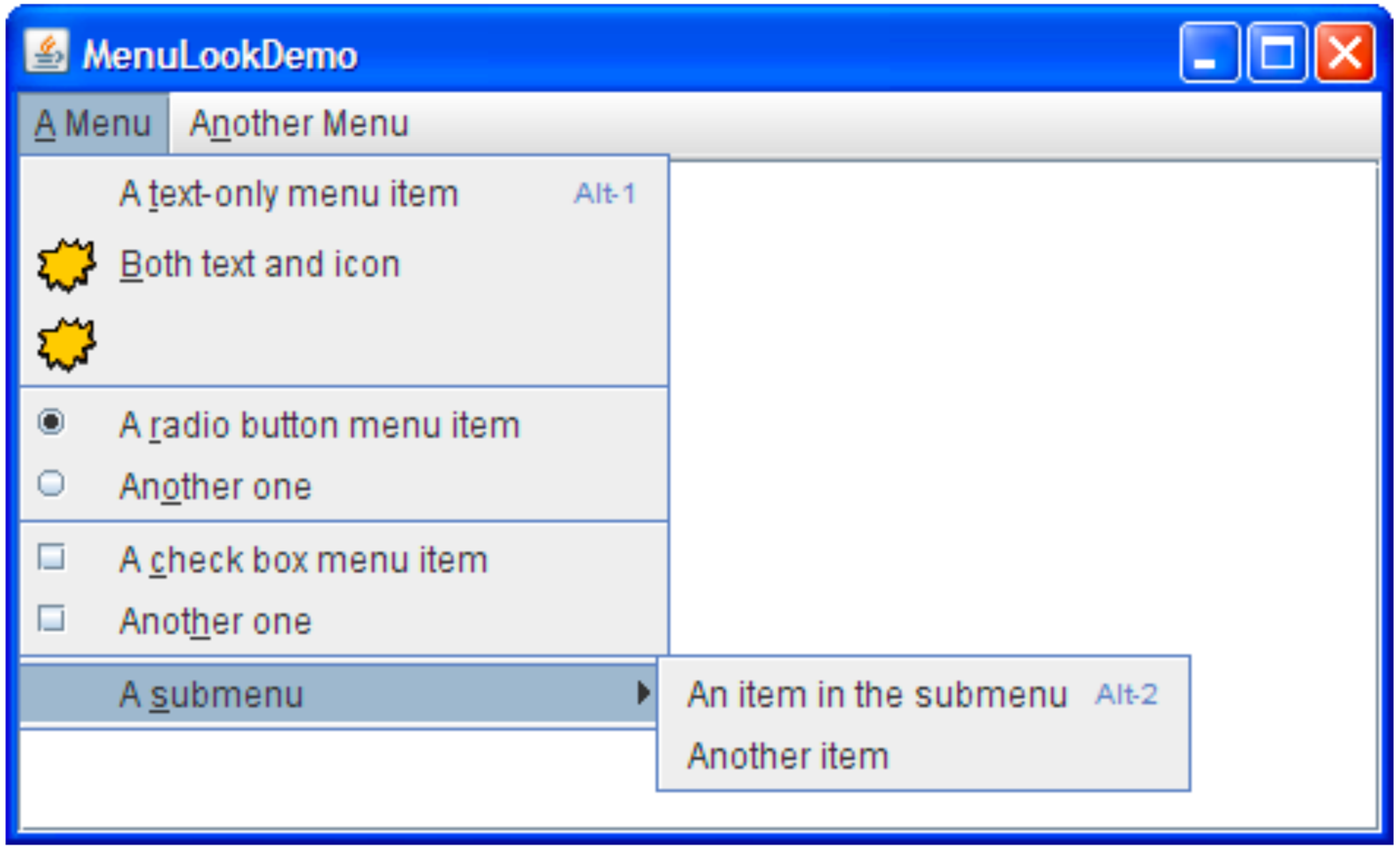
- Existem diversos tipos de painéis
  - JTabbedPane



Menus

- Um menu serve para agrupar componentes na forma de componentes *drop-down*
- Em geral, usa-se **JMenuBar**, **JMenu**, **JMenuIten**
  - Os ítens de um menu podem ser outros componentes
  - Ex: **RadioButtonMenuItem, CheckBoxMenuItem**
- É possível associar mnemônicos aos menus e ítens de menu
  - Os ítens de menu também podem ter *aceleradores*
  - São combinações de teclas que ativam o item diretamente (visível ou não)

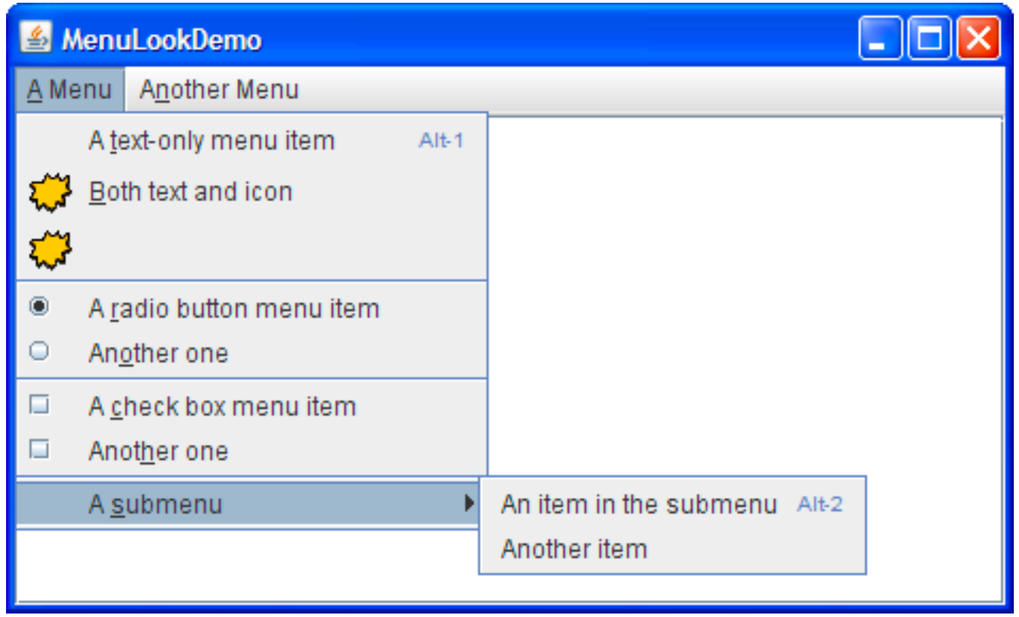
Menus


MenuLookDemo
A Menu
Another Menu
A text-only menu item
Alt-1
Both text and icon
A radio button menu item
Another one
A check box menu item
Another one
A submenu
An item in the submenu
Alt-2
Another item

Menus

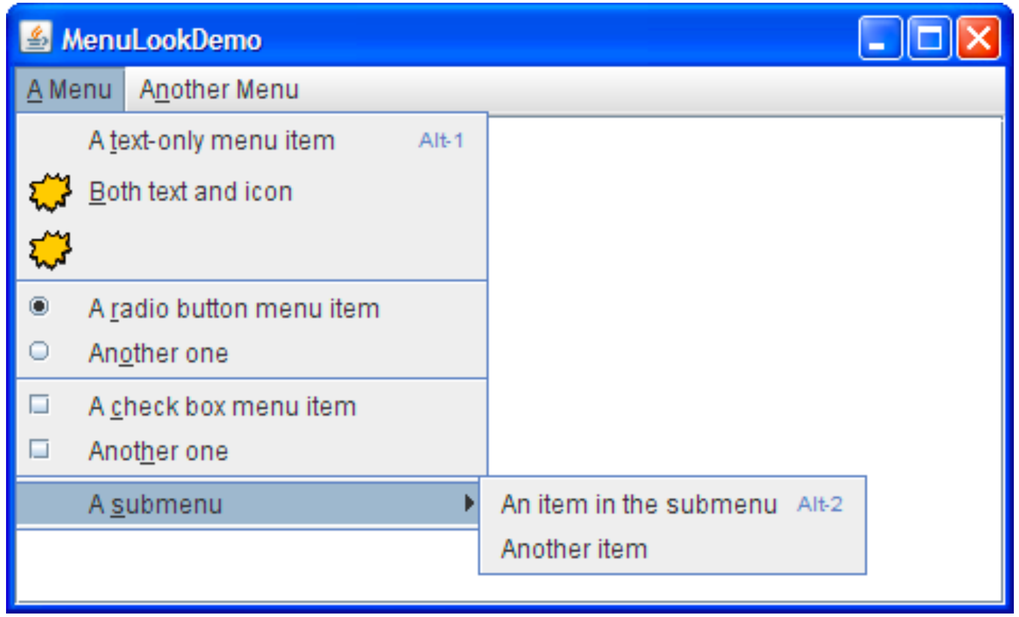
```
// Criar a barra de menu
JMenuBar menuBar = new JMenuBar();

// Construir o primeiro menu
JMenu menu1 = new JMenu("A Menu");
// Definir uma tecla de atalho para o menu
menu1.setMnemonic(KeyEvent.VK_A);
// Adicionar o menu à barra de menu
menuBar.add(menu1);

// Criar um grupo de itens de menu
JMenuItem menuItem = new JMenuItem("A text-only menu item", KeyEvent.VK_T);
// Definir um atalho de teclado para o item de menu
menuItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_1, ActionEvent.ALT_MASK)); // Alt + 1
// Adicionar o item de menu ao menu
menu1.add(menuItem);

// Criar um item de menu com texto e ícone
menuItem = new JMenuItem("Both text and icon", new ImageIcon("images/middle.gif"));
// Definir uma tecla de atalho para o item de menu
menuItem.setMnemonic(KeyEvent.VK_B);
// Adicionar o item de menu ao menu1
menu1.add(menuItem);
```

Menus



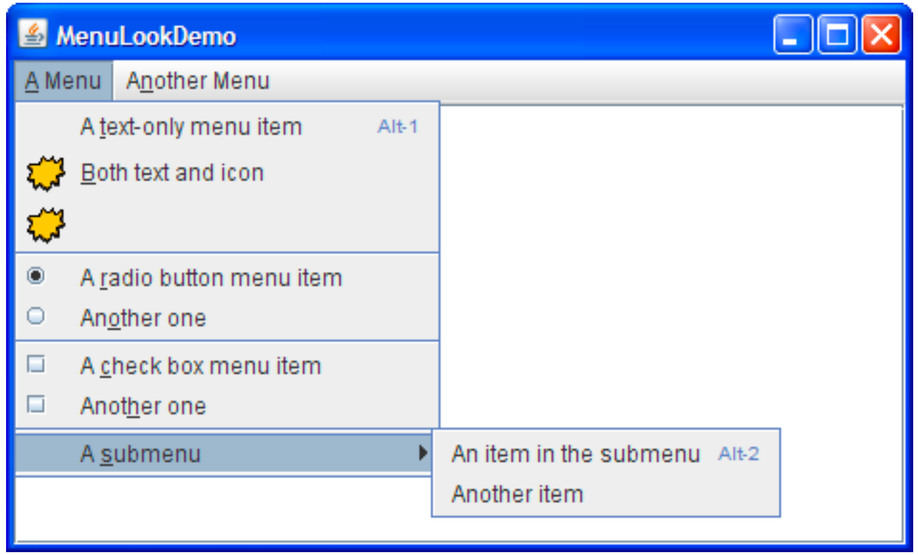

```java
// Cria um novo item de menu com um ícone
menuItem = new JMenuItem(new ImageIcon("images/middle.gif"));
// Define a tecla de atalho para o item de menu
menuItem.setMnemonic(KeyEvent.VK_D);
// Adiciona o item de menu ao menu1
menu1.add(menuItem);

// Adiciona um separador entre os itens de menu
menu1.addSeparator();

// Cria um grupo de botões de rádio para os itens de menu
ButtonGroup group = new ButtonGroup();
// Cria um item de menu do tipo botão de rádio e define seu texto
JRadioButtonMenuItem rbMenuItem =
new JRadioButtonMenuItem("A radio button menu item");
// Define o botão de rádio como selecionado por padrão
rbMenuItem.setSelected(true);
// Adiciona o item de menu do tipo botão de rádio ao grupo
group.add(rbMenuItem);
// Adiciona o item de menu ao menu1
menu1.add(rbMenuItem);
// Cria outro item de menu do tipo botão de rádio
rbMenuItem = new JRadioButtonMenuItem("Another one");
// Adiciona o item de menu do tipo botão de rádio ao grupo
group.add(rbMenuItem);
// Adiciona o item de menu ao menu1
menu1.add(rbMenuItem);
```

Menus



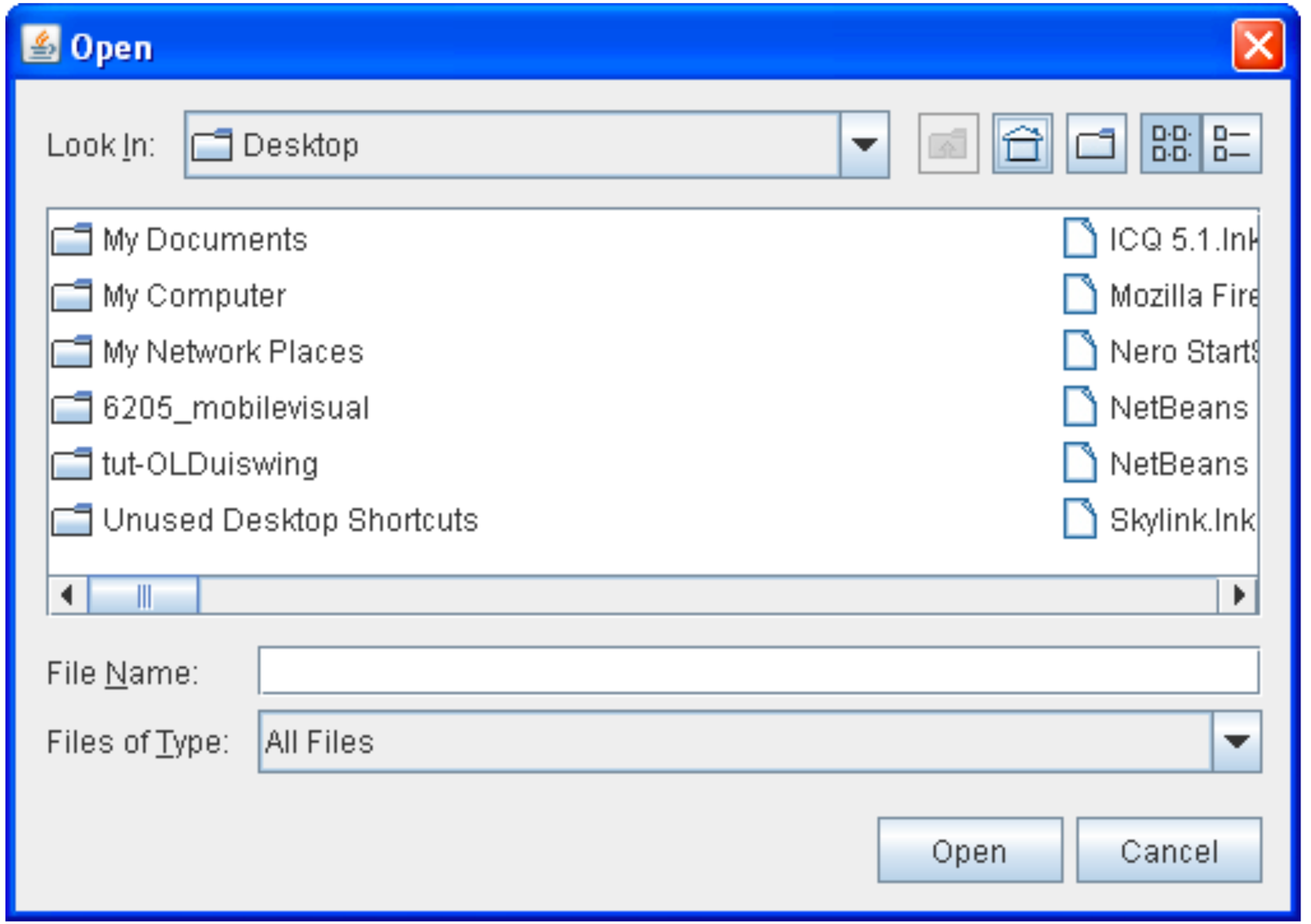

```
// Adiciona um separador entre os itens de menu
menu1.addSeparator();

// Cria um item de menu do tipo checkbox e define seu texto
JCheckBoxMenuItem cbMenuItem =
new JCheckBoxMenuItem("A check box menu item");
// Define a tecla de atalho para o item de menu
cbMenuItem.setMnemonic(KeyEvent.VK_C);
// Adiciona o item de menu do tipo checkbox ao menu1
menu1.add(cbMenuItem);

// Cria outro item de menu do tipo checkbox
cbMenuItem = new JCheckBoxMenuItem("Another one");
// Define a tecla de atalho para o item de menu
cbMenuItem.setMnemonic(KeyEvent.VK_H);
// Adiciona o item de menu do tipo checkbox ao menu1
menu1.add(cbMenuItem);

// Adiciona um separador entre os itens de menu
menu1.addSeparator();

// Cria um submenu dentro do menu1
JMenu submenu = new JMenu("A submenu");
// Define a tecla de atalho para o submenu
submenu.setMnemonic(KeyEvent.VK_S);
```

Menus




```
 // Cria um item de menu dentro do submenu
menuItem = new JMenuItem("An item in the submenu");
// Define um atalho para o item de menu (Alt + 2)
menuItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_2,
ActionEvent.ALT_MASK));
// Adiciona o item de menu ao submenu
submenu.add(menuItem);

// Cria outro item de menu dentro do submenu
menuItem = new JMenuItem("Another item");
// Adiciona o item de menu ao submenu
submenu.add(menuItem);
// Adiciona o submenu ao menu1
menu1.add(submenu);

// Cria o segundo menu na barra de menus
JMenu menu2 = new JMenu("Another Menu");
// Define a tecla de atalho para o segundo menu
menu2.setMnemonic(KeyEvent.VK_N);
// Adiciona o segundo menu à barra de menus
menuBar.add(menu2);

// Define a barra de menus no JFrame
f.setJMenuBar(menuBar);
```

JFileChooser



JFileChooser

- Abrindo um arquivo
- Possíveis retornos
  - JFileChooser.CANCEL_OPTION
  - JFileChooser.APPROVE_OPTION
  - JFileChooser.ERROR_OPTION

```
JFileChooser chooser = new JFileChooser();

FileNameExtensionFilter filter = new FileNameExtensionFilter(
                                 "JPG & GIF Images", "jpg“, "gif");
chooser.setFileFilter(filter);

int returnVal = chooser.showOpenDialog(parent);
if(returnVal == JFileChooser.APPROVE_OPTION) {
    File selectedFile = chooser.getSelectedFile();
    System.out.println("You chose : “ + selectedFile.getName());
}
```

JFileChooser

- Salvando um arquivo
- Possíveis retornos
  - JFileChooser.CANCEL_OPTION
  - JFileChooser.APPROVE_OPTION
  - JFileChooser.ERROR_OPTION

```
JFileChooser chooser = new JFileChooser();
int returnVal = chooser.showSaveDialog(parent);

if(returnVal == JFileChooser.APPROVE_OPTION) {
    File selectedFile = chooser.getSelectedFile();
    System.out.println("You chose to save this file as: " +
                       selectedFile.getName());
}
```

unesp

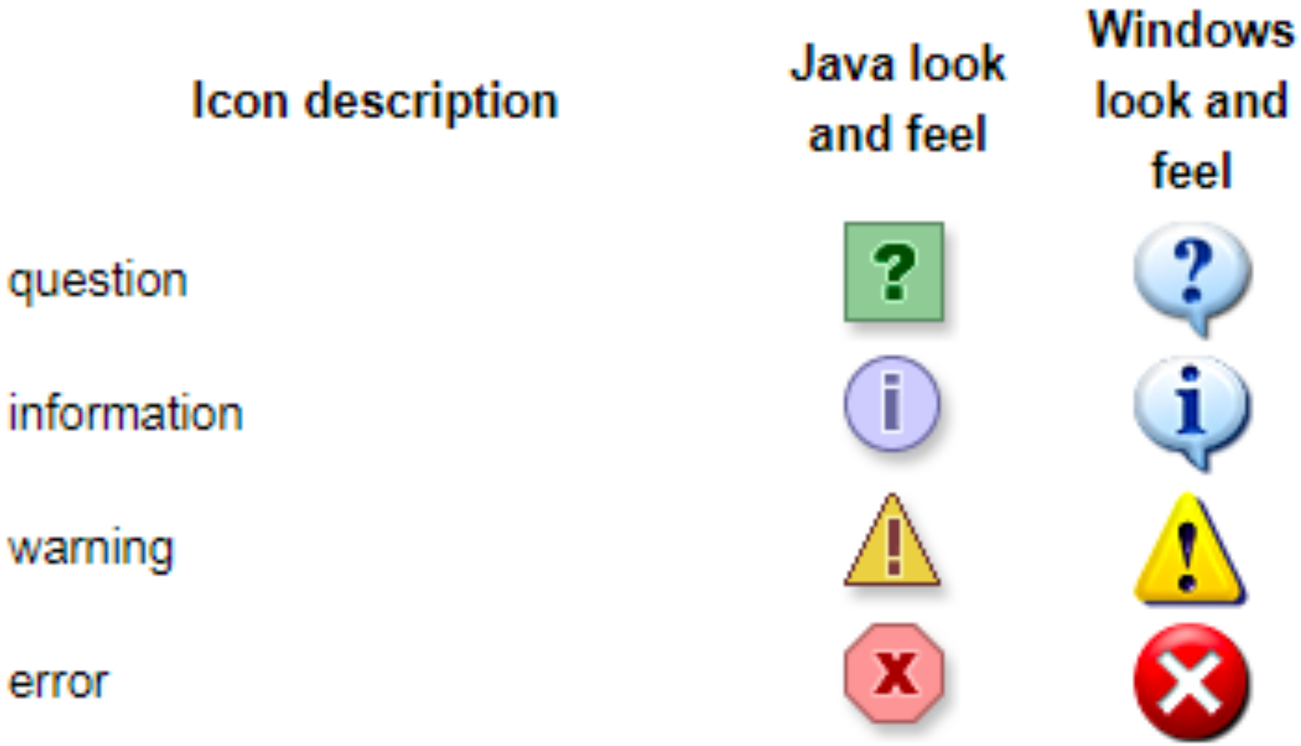
E/S com JOptionPane

- Uma forma rápida e simples de criar caixas de diálogo para
  - Mostrar uma mensagem na tela (informação)
    - **showMessageDialog**
  - Requisitar uma confirmação do usuário (sim/não)
    - **showConfirmDialog**
  - Ler uma informação do usuário (texto, combo)
    - **showInputDialog**
  - Todos os tipos podem ser criados usando a forma genérica
    - **showOptionDialog**

unesp

E/S com JOptionPane

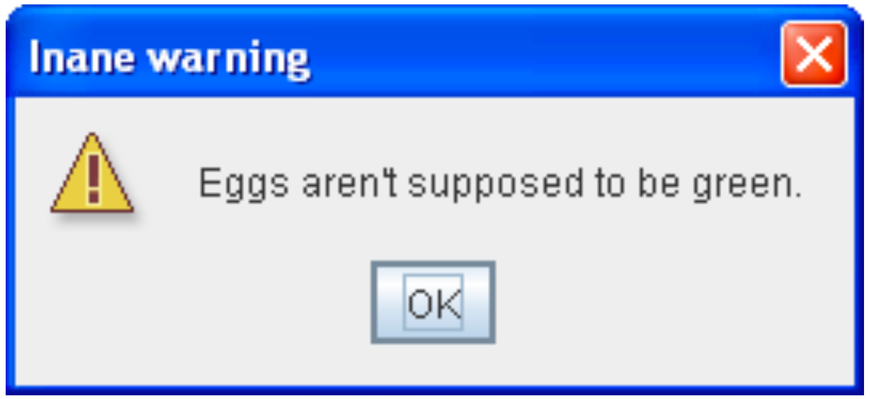
- Exibir informações
  - Para mostrar uma mensagem na tela, em geral passamos um título para a janela, a mensagem e um ícone



E/S com JOptionPane

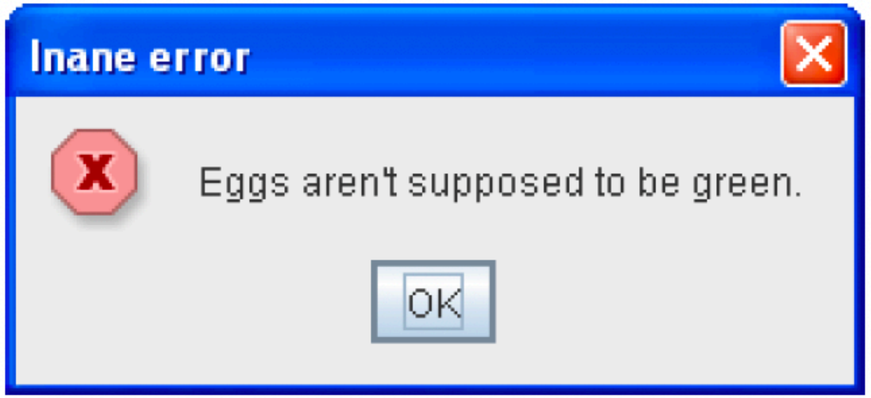
```
JOptionPane.showMessageDialog(f, "Eggs are not supposed to be
                                  green.");
```


```
JOptionPane.showMessageDialog(f, "Eggs are not supposed to be
        green.“, "Inane warning", JOptionPane.WARNING_MESSAGE);
```

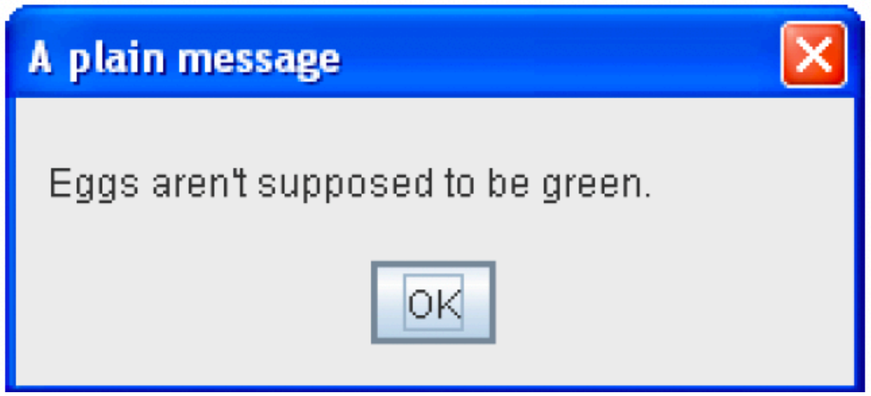


E/S com JOptionPane

```
JOptionPane.showMessageDialog(f, "Eggs are not supposed to be
        green.“, "Inane error", JOptionPane.ERROR_MESSAGE);
```




```
JOptionPane.showMessageDialog(f, "Eggs are not supposed to be
        green.“, “A plain message", JOptionPane.PLAIN_MESSAGE);
```



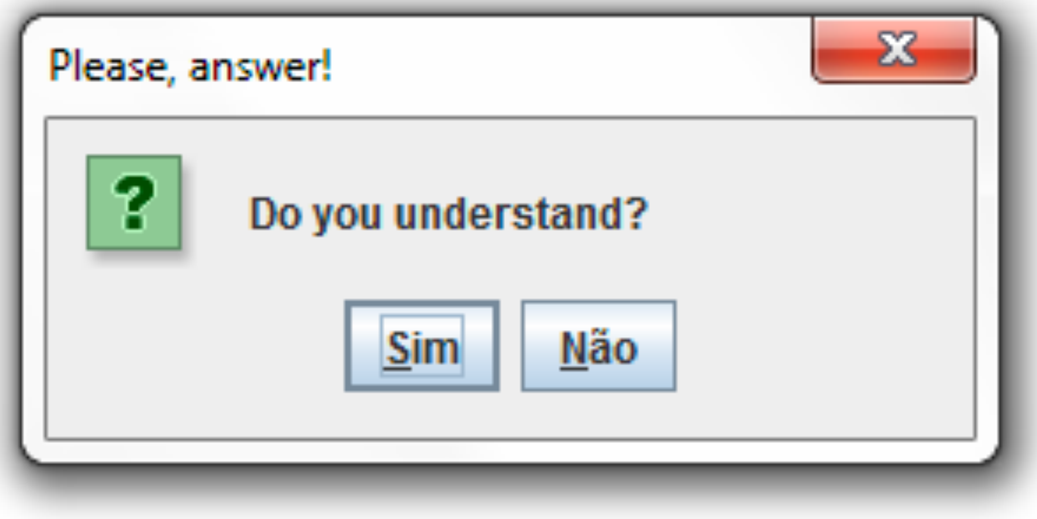
E/S com JOptionPane

- Pedir uma confirmação do usuário
  - Similar ao anterior
  - Porém, a caixa de diálogo terá dois ou três botões de opção
  - SIM / NÃO
  - SIM / NÃO / CANCELA
  - Personalizado
  - O botão clicado retorna um inteiro que identifica a opção escolhida

unesp

E/S com JOptionPane

```
int opt = JOptionPane.showConfirmDialog(f,
                                "Do you understand?",
                                "Please, answer!",
                                JOptionPane.YES_NO_OPTION,
                                JOptionPane.QUESTION_MESSAGE);
```



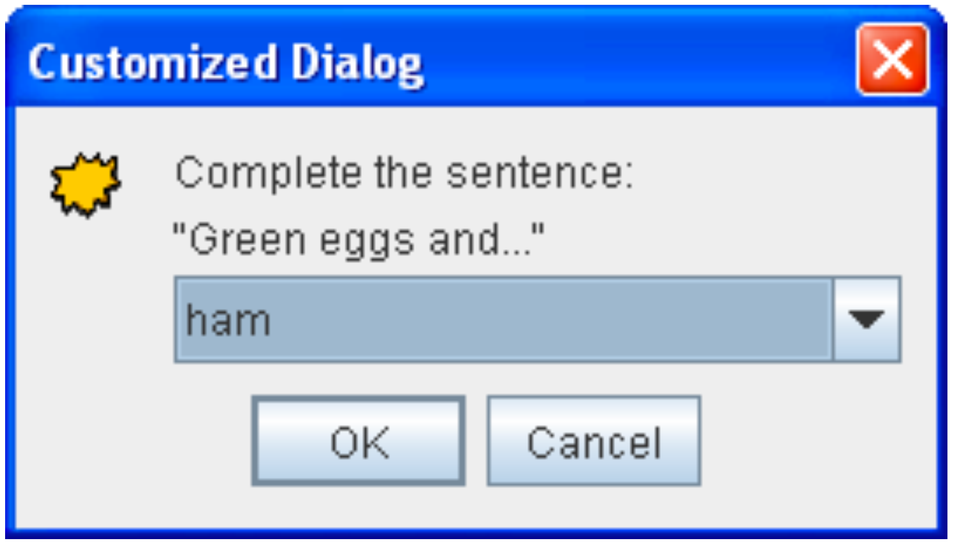
E/S com JOptionPane

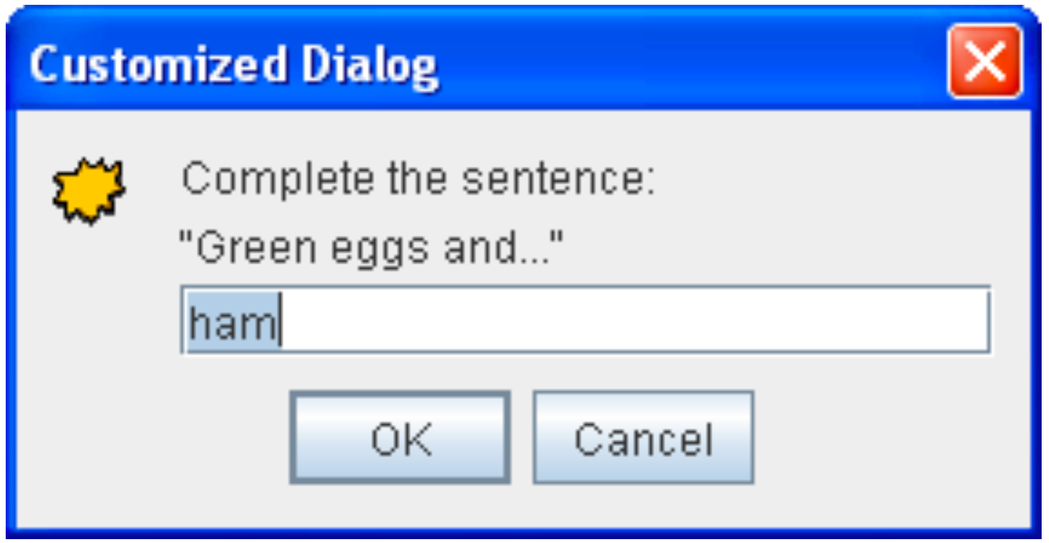
- Personalizado
  - Usa **showOptionDialog**

```
String[] options = {"Not sure", "Maybe", "What?"};
int opt = JOptionPane.showOptionDialog(f,
                    "Do you understand?",
                     "Please, answer!",
                     JOptionPane.YES_NO_CANCEL_OPTION,
                     JOptionPane.QUESTION_MESSAGE,
                     null,
                     options,
                     options[0]);
```

E/S com JOptionPane

- Ler uma informação do usuário
  - Similar aos anteriores
  - Neste caso, a caixa de diálogo exibirá um campo de texto ou um *combo box* para ler a entrada





E/S com JOptionPane

- Usando *combo box*

```
Object[] possibilities = {"ham", "spam", "yam"};

String s = (String)JOptionPane.showInputDialog(f,
            "Complete the sentence:\n" + "\"Green eggs and...\"",
            "Customized Dialog",
            JOptionPane.PLAIN_MESSAGE,
            icon,
            possibilities,
            "ham");
```

E/S com JOptionPane

- Usando caixa de texto

```
Object[] possibilities = {"ham", "spam", "yam"};

String s = (String)JOptionPane.showInputDialog(f,
            "Complete the sentence:\n" + "\"Green eggs and...\"",
            "Customized Dialog",
            JOptionPane.PLAIN_MESSAGE,
            icon,
            null,
            "ham");
```

Gerenciadores de Layout

Gerenciadores de Layout

- Os gerenciadores de layout organizam a posição e tamanho dos componentes dentro de um container
- Sem um gerenciador de layout, seria preciso
  - Especificar a posição absoluta de cada componente no container (em função do canto superior esquerdo)
  - Não haveria controle automático de tamanho e posicionamento em caso de redimensionamento
- Todo *conteiner* tem um método **setLayout**
  - Após definir o layout, inserções são organizadas pelo objeto de layout
  - Em alguns casos, inserção já informa algo sobre o layout (posição, restrições)

unesp

Gerenciadores de Layout

- Existem vários controladores de layout
  - FlowLayout
  - BorderLayout
  - CardLayout
  - BoxLayout
  - GroupLayout
  - GridLayout
  - GridBagLayout
- Comentaremos sobre alguns
- Mais detalhes: https://docs.oracle.com/javase/tutorial/uiswing/layout/using.html

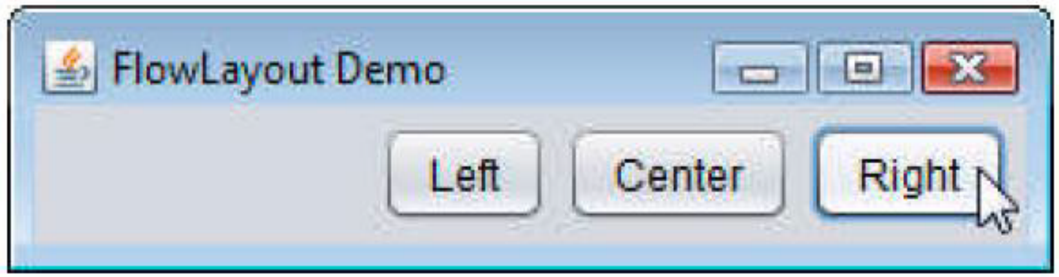
Gerenciadores de Layout

- FlowLayout
  - Padrão para JPanel
  - Insere os componentes da esquerda para a direita
    - Ordem de inserção
  - Se não couber em uma linha, continua na próxima
  - É possível alinhar à esquerda, centro e direita

Gerenciadores de Layout

- FlowLayout

```
FlowLayout flowLayout = new FlowLayout();
flowLayout.setAlignment(FlowLayout.RIGHT);
f.setLayout(flowLayout);

JButton leftButton = new JButton("Left");
JButton centerButton = new JButton("Center");
Jbutton rightButton = new JButton("Right");

f.add(leftButton); f.add(centerButton); f.add(RightButton);
```



Gerenciadores de Layout

- BorderLayout
  - Padrão para JFrame
  - Organiza os componentes em cinco regiões:
    - NORTH, SOUTH, EAST, WEAST, CENTER
  - Limita o container a ter no máximo 5 componentes
  - Porém, cada componente pode ser um container
  - NORTH → Topo (linha de cima)
  - SOUTH → Base (linha de baixo)
  - EAST, CENTER, WEAST → linha do meio
  - Ao adicionar componentes, região deve ser informada

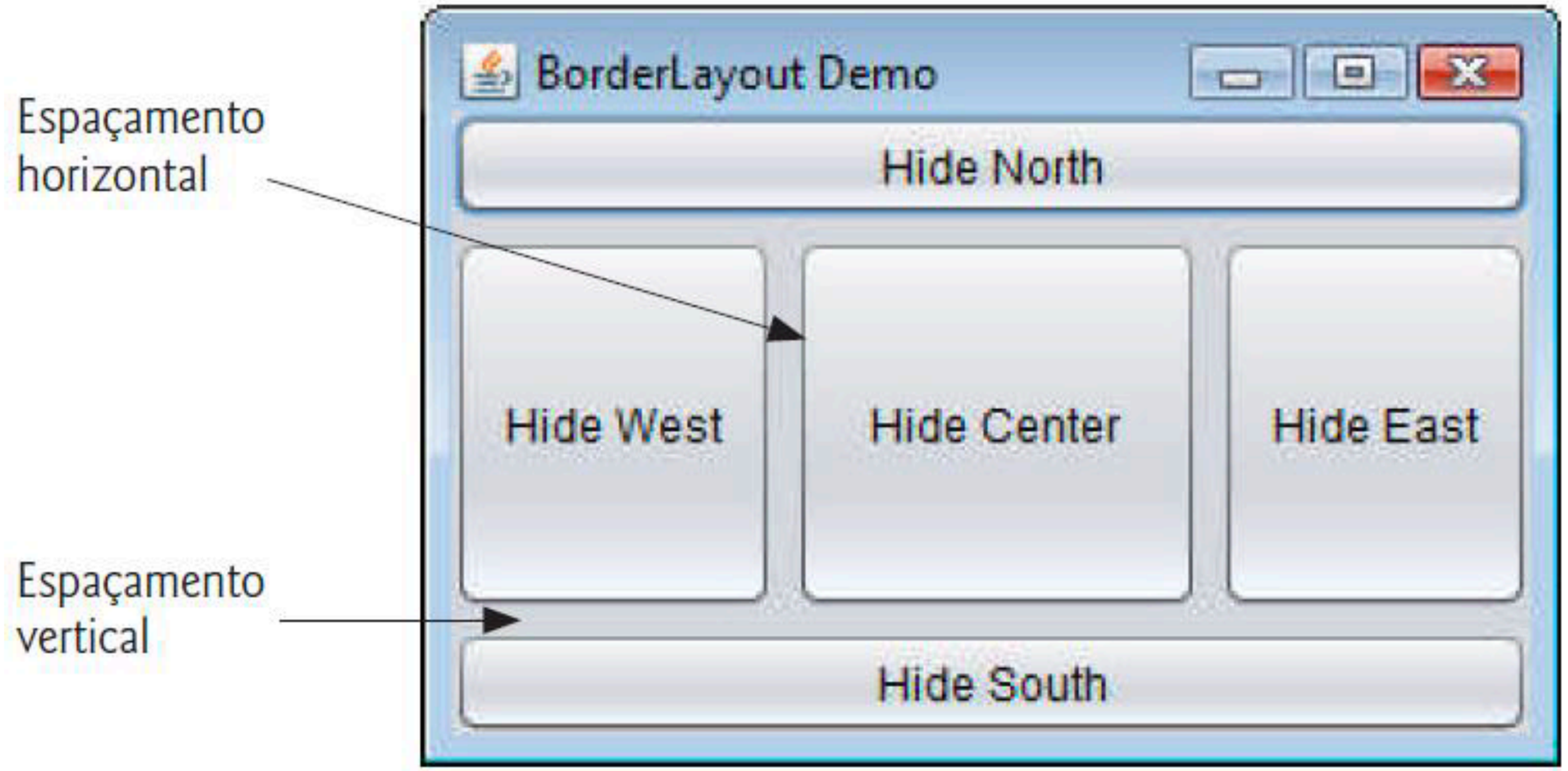
Gerenciadores de Layout

- BorderLayout

```
BorderLayout borderLayout = new BorderLayout(5,5); //spacing 5px
f.setLayout(borderLayout);

String[] names = {“Hide North”, “Hide South”, “Hide East”,
                  “Hide West”, “Hide Center”};
JButton[] buttons = new JButton[names.length];

for (int count = 0; count < names.length; count++) {
    buttons[count] = new JButton(names[count]);
    // add listener
}

f.add(buttons[0], BorderLayout.NORTH);
f.add(buttons[1], BorderLayout.SOUTH);
f.add(buttons[2], BorderLayout.EAST);
f.add(buttons[3], BorderLayout.WEST);
f.add(buttons[4], BorderLayout.CENTER);
```

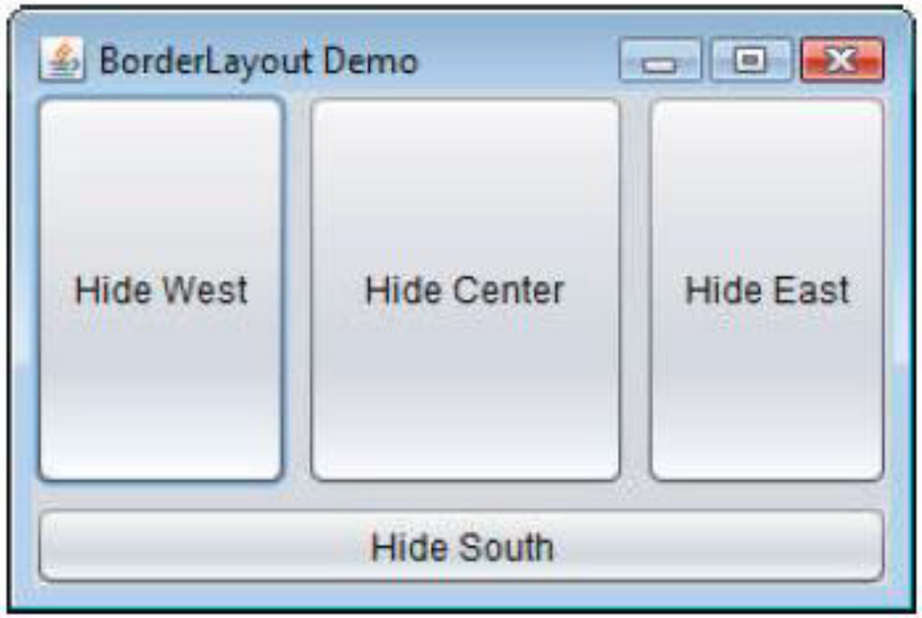
Gerenciadores de Layout

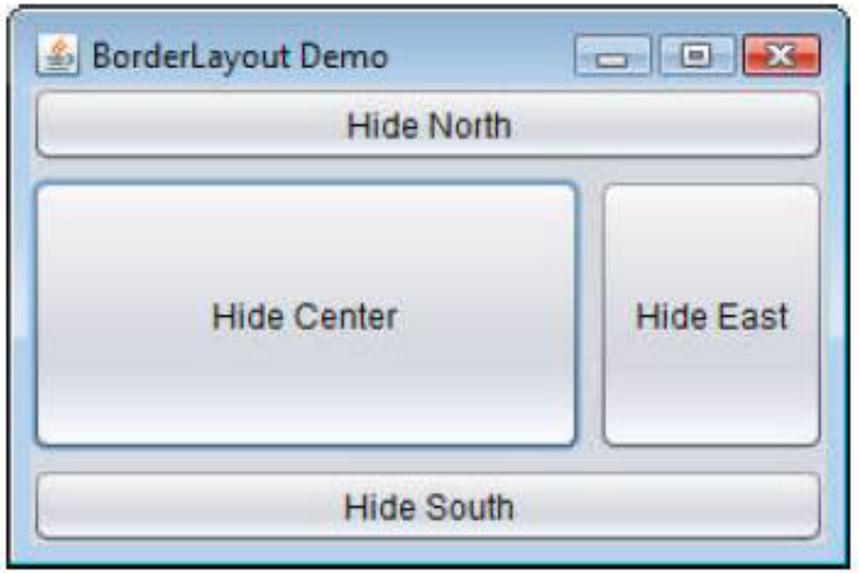
- BorderLayout

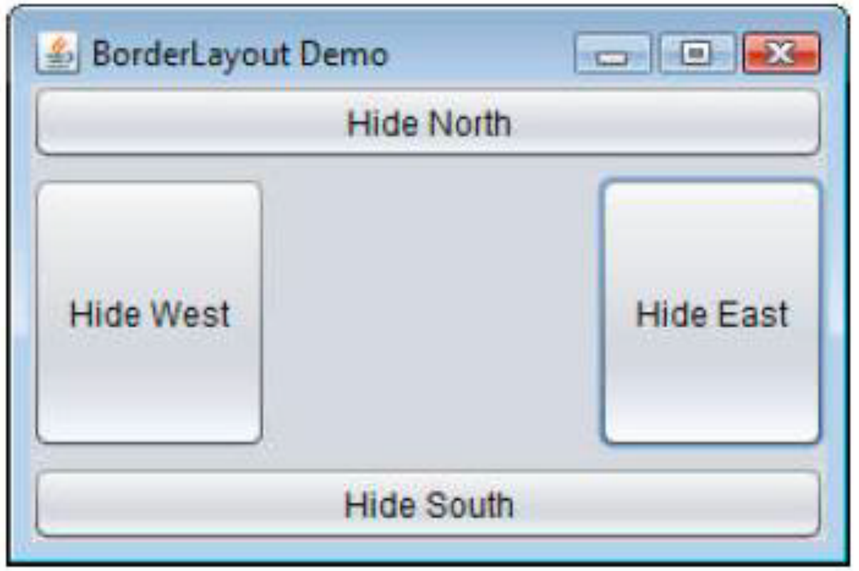


Gerenciadores de Layout

- BorderLayout







Gerenciadores de Layout

- GridLayout
  - Divide o container em uma grade (linhas e colunas)
  - Cada elemento é adicionado em uma posição dessa grade
    - Começando de cima para baixo, esquerda para direita
  - Todos os elementos da grade tem mesma altura e largura

unesp

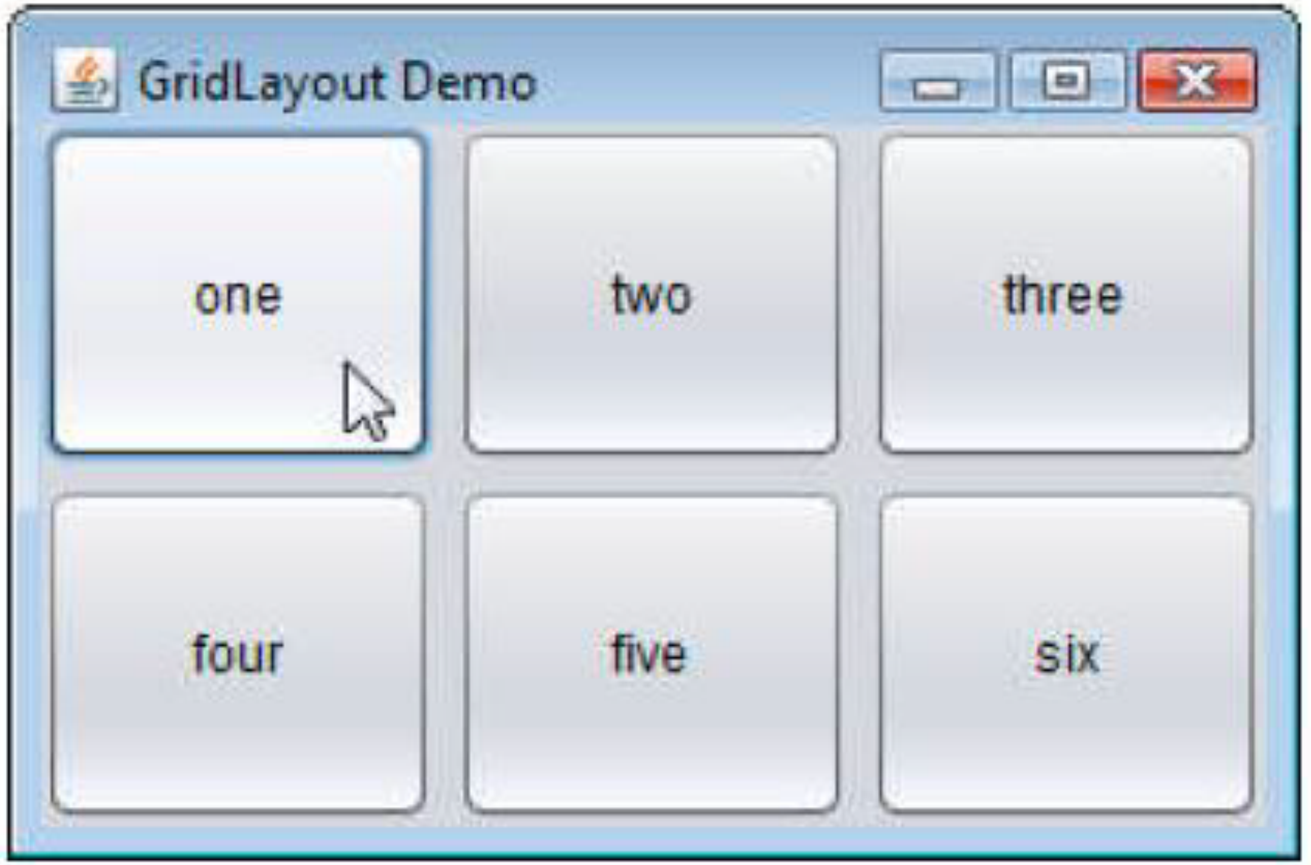
Gerenciadores de Layout

- GridLayout

```
// Grid 2x3 with spacing 5px
GridLayout gridLayout = new GridLayout(2,3,5,5);

f.setLayout(gridLayout);

String[] names = {“one”, “two”, “three”, “four”, “five”, “six”};
JButton[] buttons = new JButton[names.length];

for (int count = 0; count < names.length; count++) {
    buttons[count] = new JButton(names[count]);
    f.add(buttons[count]);
    // add listener
}
```

Gerenciadores de Layout

- GridLayout



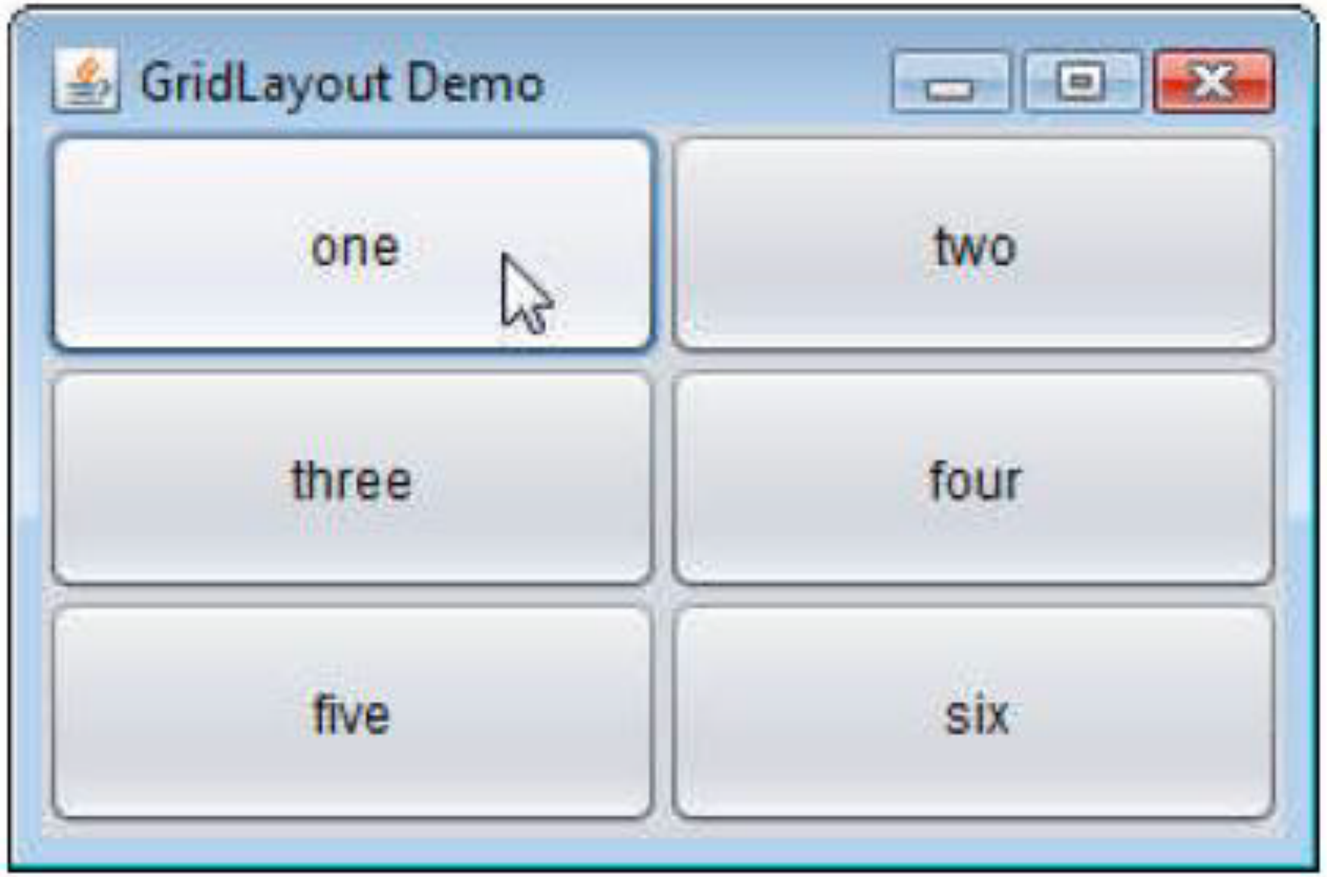
Gerenciadores de Layout

- GridLayout

```
// Grid 3x2 with no spacing
GridLayout gridLayout = new GridLayout(3,2);

f.setLayout(gridLayout);

String[] names = {“one”, “two”, “three”, “four”, “five”, “six”};
JButton[] buttons = new JButton[names.length];

for (int count = 0; count < names.length; count++) {
    buttons[count] = new JButton(names[count]);
    f.add(buttons[count]);
    // add listener
}
```

Gerenciadores de Layout

- GridLayout



Gerenciadores de Layout

- GridBagLayout
  - Permite definir layouts mais flexíveis
  - Porém, o trabalho se torna mais complexo
  - NetBeans possui uma ferramenta para ajuste do controlador de layout que facilita bastante o trabalho

Gerenciadores de Layout

- GridBagLayout
  - Assim como GridLayout, é um layout de grades
    - Linhas e colunas
  - Porém, os componentes podem ocupar mais de uma linha ou coluna
  - Cada linha ou coluna pode ter tamanhos diferentes
  - Posição e espaçamento são ajustados para cada célula
  - É possível especificar quais células (e em qual proporção) deverão ser redimensionadas junto com o redimensionamento do container
  - Cada componente adicionado ao container vai associado a um objeto **GridBagConstraints**

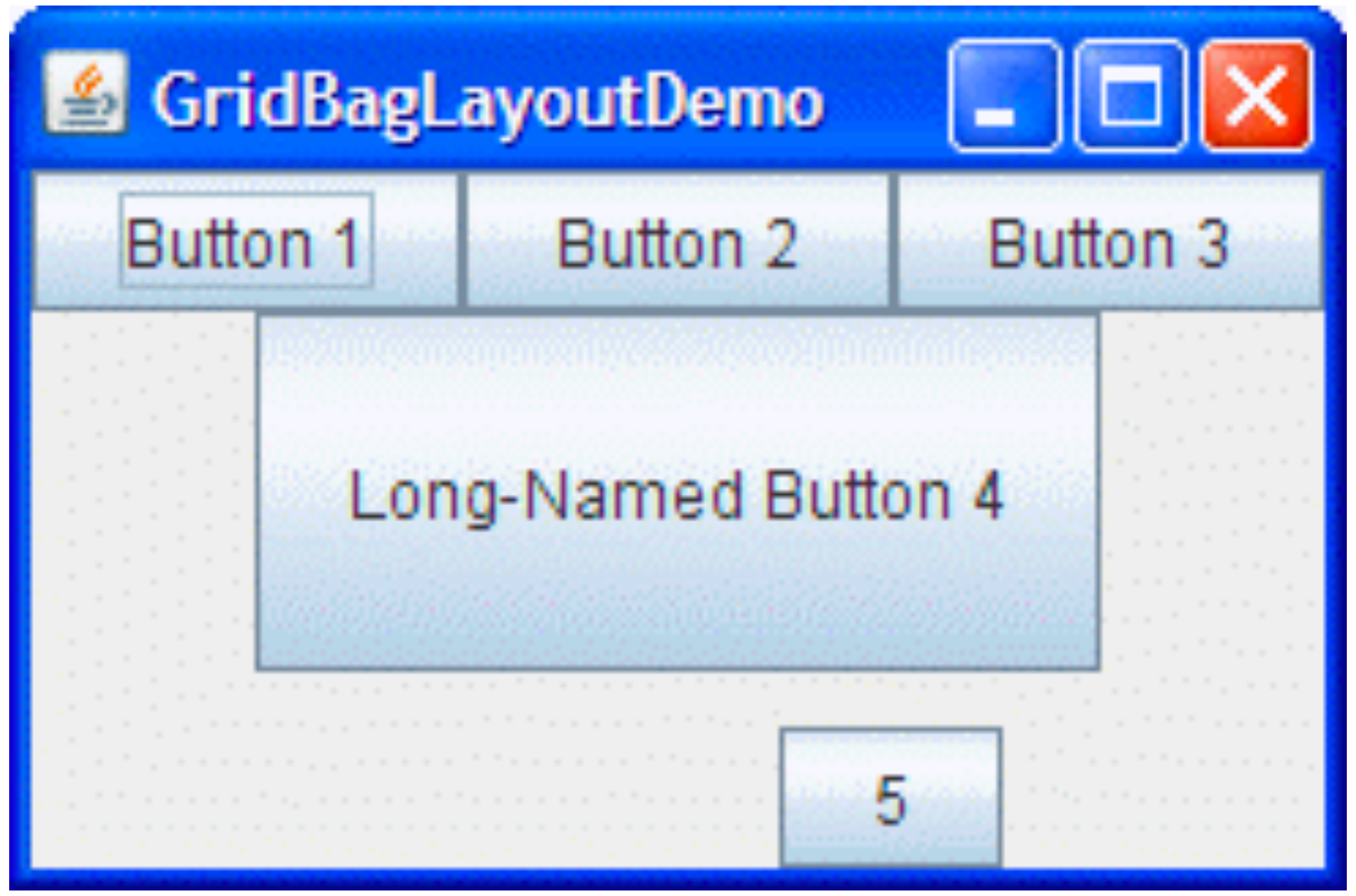
unesp

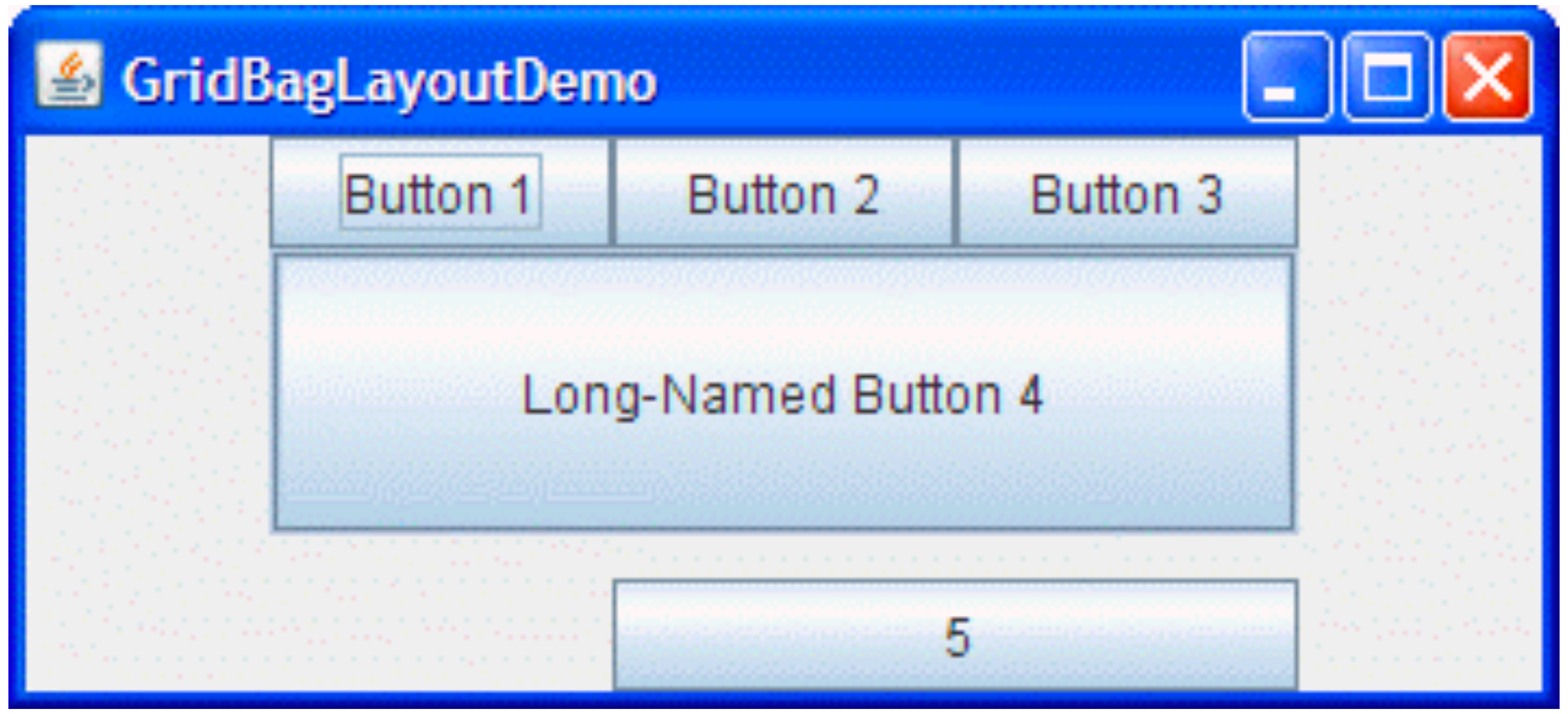
Gerenciadores de Layout

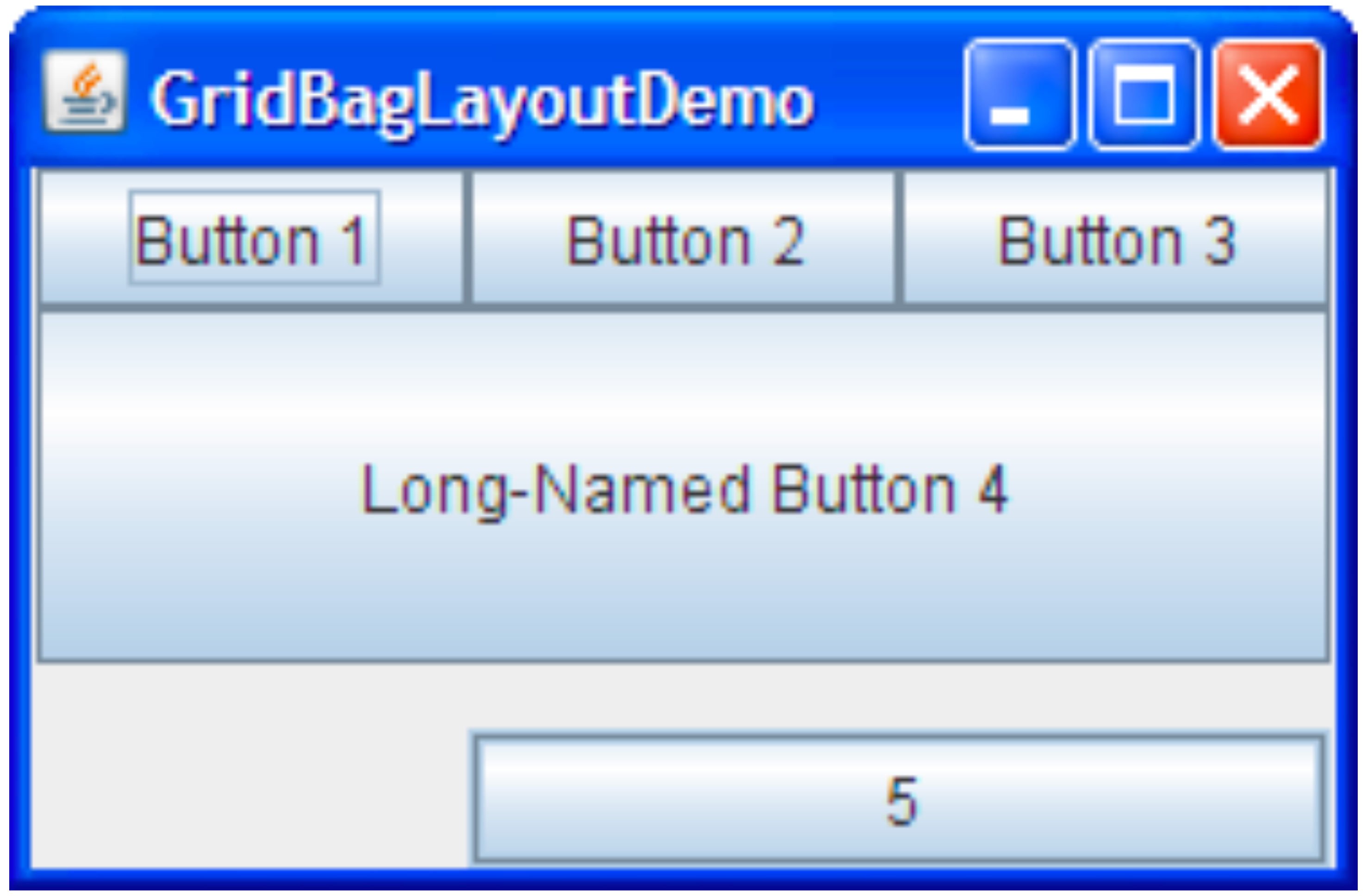
- GridBagLayout

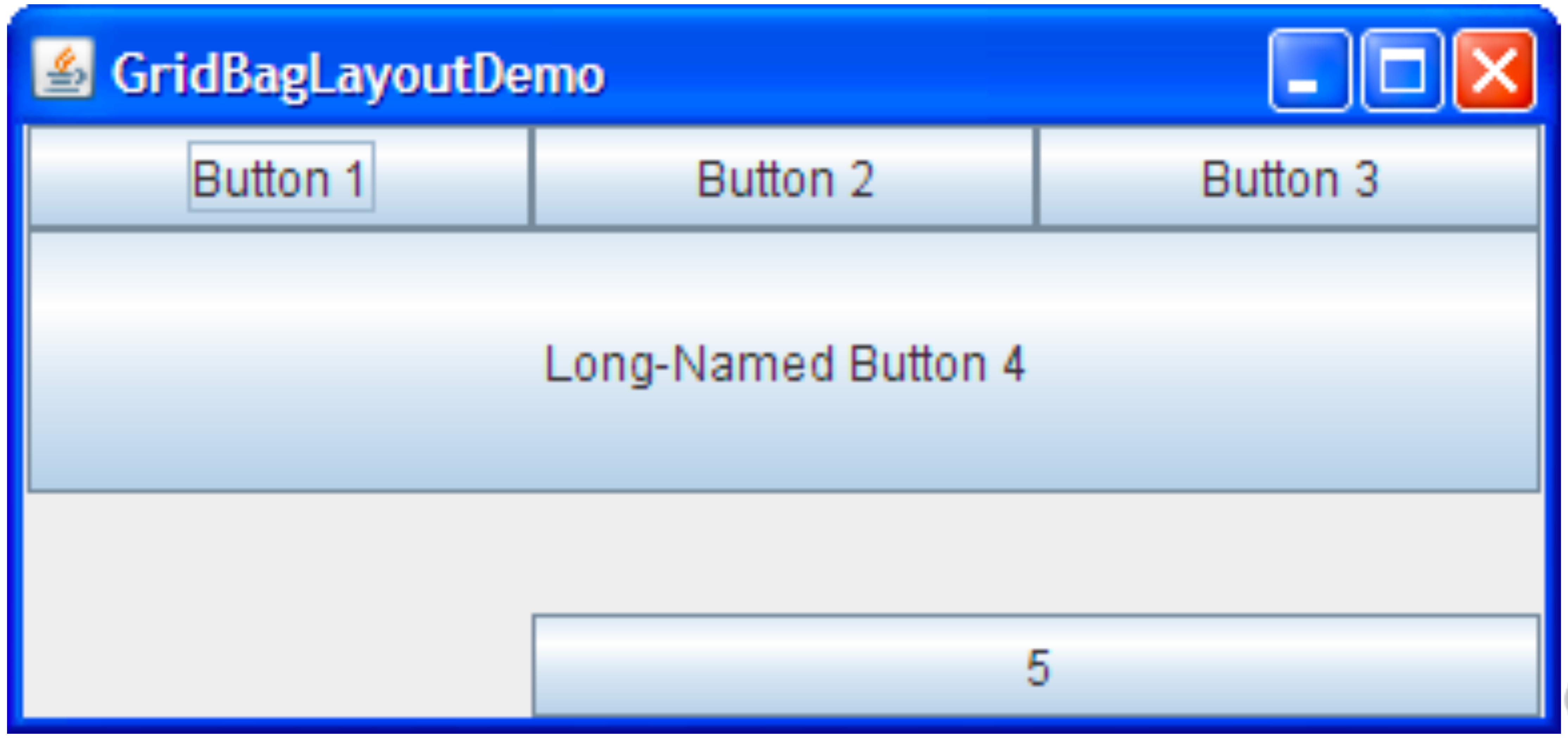
Gerenciadores de Layout

- GridBagLayout



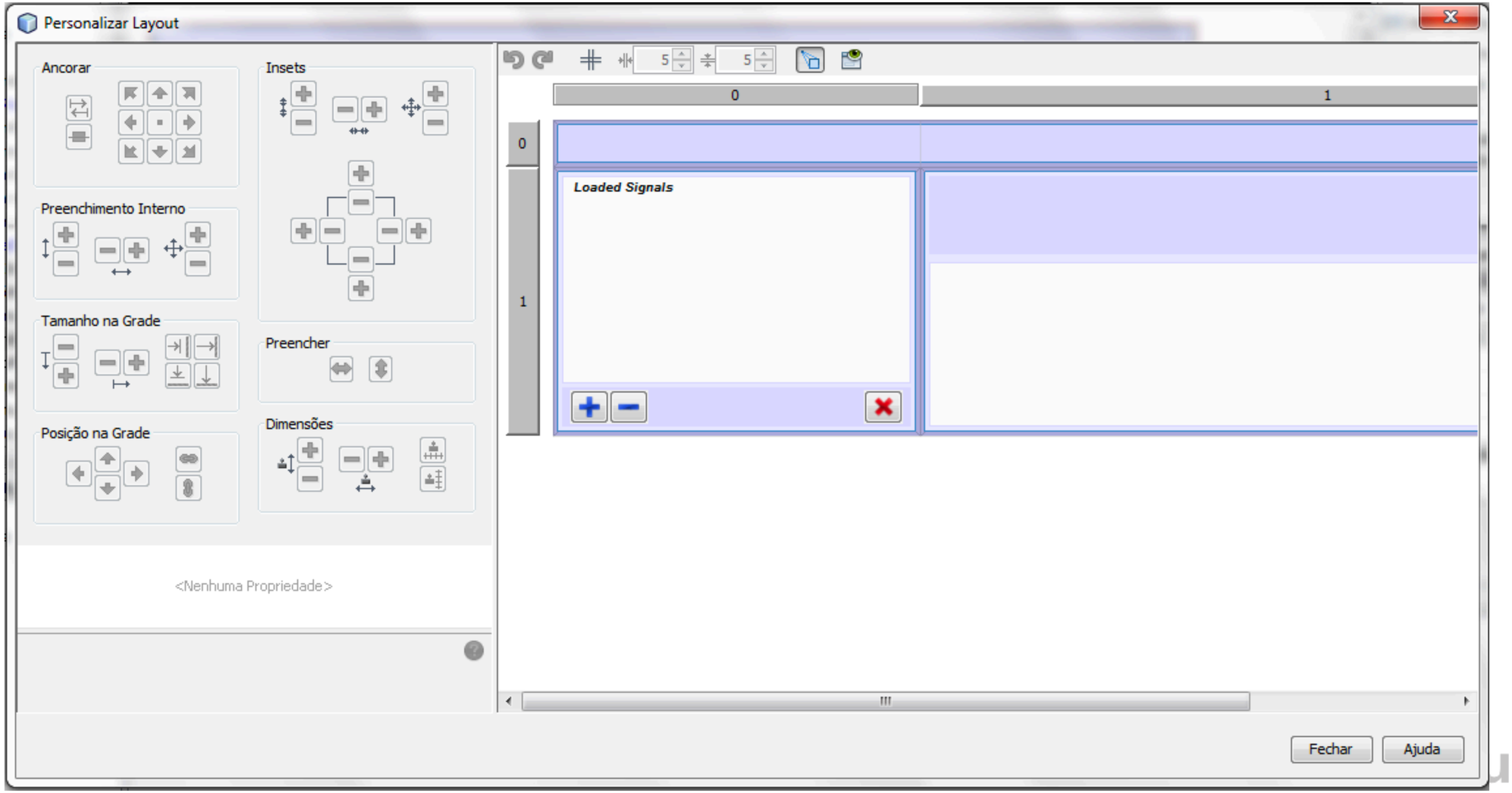






Gerenciadores de Layout

- GridBagLayout



Look and Feel

Look and Feel

- As GUIs Swing permitem modificar a aparência de seus componentes através do que o Java chama de Look and Feel (L&F)
- Em geral, L&F deve ser ajustado antes de criar os componentes
- Tipos de L&F
  - CrossPlatformLookAndFeel → É o L&F padrão do Java (também conhecido como Metal)
  - SystemLookAndFeel → Utiliza a aparência dos componentes do sistema nativo
  - Nimbus → L&F cross-plataforma do Java que utiliza o Java 2D *graphics* para criar os componentes gráficos

unesp

Look and Feel

- L&F do Windows só funciona no windows
- GTK+ só funciona se GTK+ 2.2 (ou mais recente) estiver instalado

Platform	Look and Feel
Solaris, Linux with GTK+ 2.2 or later	GTK+
Other Solaris, Linux	Motif
IBM UNIX	IBM*
HP UX	HP*
Classic Windows	Windows
Windows XP	Windows XP
Windows Vista	Windows Vista
Macintosh	Macintosh*

* Supplied by the system vendor.

Look and Feel

- L&F do Java (padrão)

```
try {
    // Set cross-platform Java L&F (also called "Metal")
    UIManager.setLookAndFeel(
            UIManager.getCrossPlatformLookAndFeelClassName());
} catch (UnsupportedLookAndFeelException e) {
    // handle exception
} catch (ClassNotFoundException e) {
    // handle exception
} catch (InstantiationException e) {
    // handle exception
} catch (IllegalAccessException e) {
    // handle exception
}
```

unesp

Look and Feel

- L&F do Java (outra maneira)

```
try {
    // Set cross-platform Java L&F (also called "Metal")
    UIManager.setLookAndFeel(
                    "javax.swing.plaf.metal.MetalLookAndFeel");
} catch (UnsupportedLookAndFeelException e) {
    // handle exception
} catch (ClassNotFoundException e) {
    // handle exception
} catch (InstantiationException e) {
    // handle exception
} catch (IllegalAccessException e) {
    // handle exception
}
```

unesp

Look and Feel

- L&F do sistema

```
try {
    // Set System L&F
    UIManager.setLookAndFeel(
                        UIManager.getSystemLookAndFeelClassName());
} catch (UnsupportedLookAndFeelException e) {
    // handle exception
} catch (ClassNotFoundException e) {
    // handle exception
} catch (InstantiationException e) {
    // handle exception
} catch (IllegalAccessException e) {
    // handle exception
}
```

unesp

Look and Feel

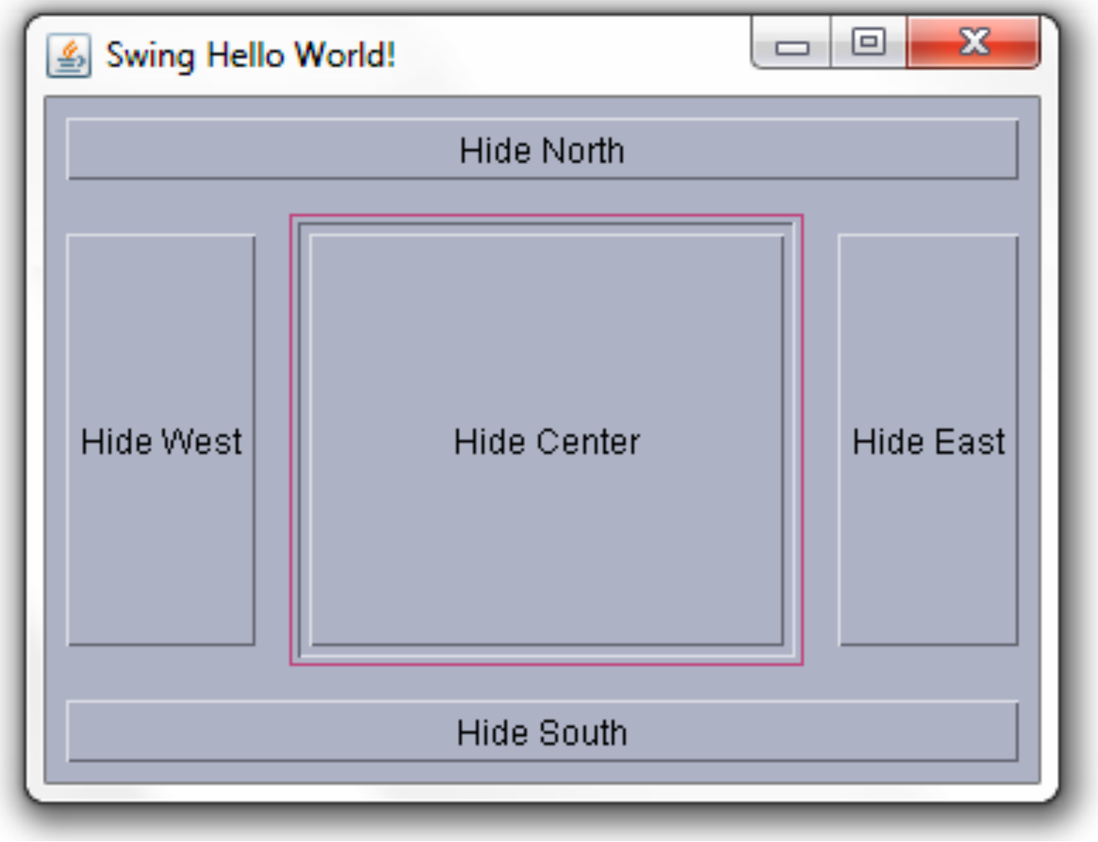
- Motif L&F

```
try {
    // Set Motif L&F on any platform
    UIManager.setLookAndFeel(
        "com.sun.java.swing.plaf.motif.MotifLookAndFeel");
} catch (UnsupportedLookAndFeelException e) {
    // handle exception
} catch (ClassNotFoundException e) {
    // handle exception
} catch (InstantiationException e) {
    // handle exception
} catch (IllegalAccessException e) {
    // handle exception
}
```

unesp

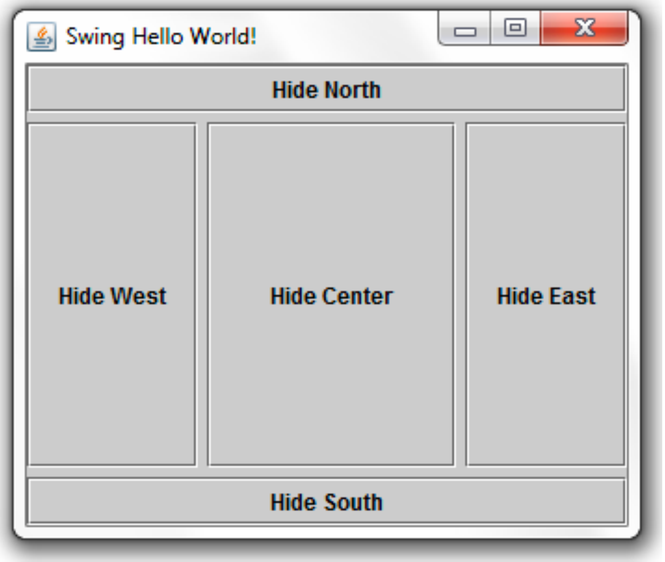
Look and Feel

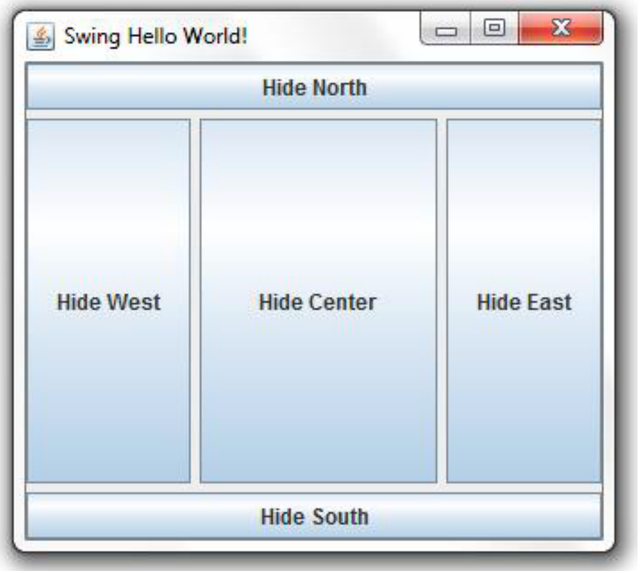
- Motif L&F



Look and Feel

- É possível alterar o tema do L&F do Java
  - Metal
  - Ocean
  - …





Look and Feel

- Nimbus
  - Para definir o Nimbus como L&F, verificamos se ele está disponível

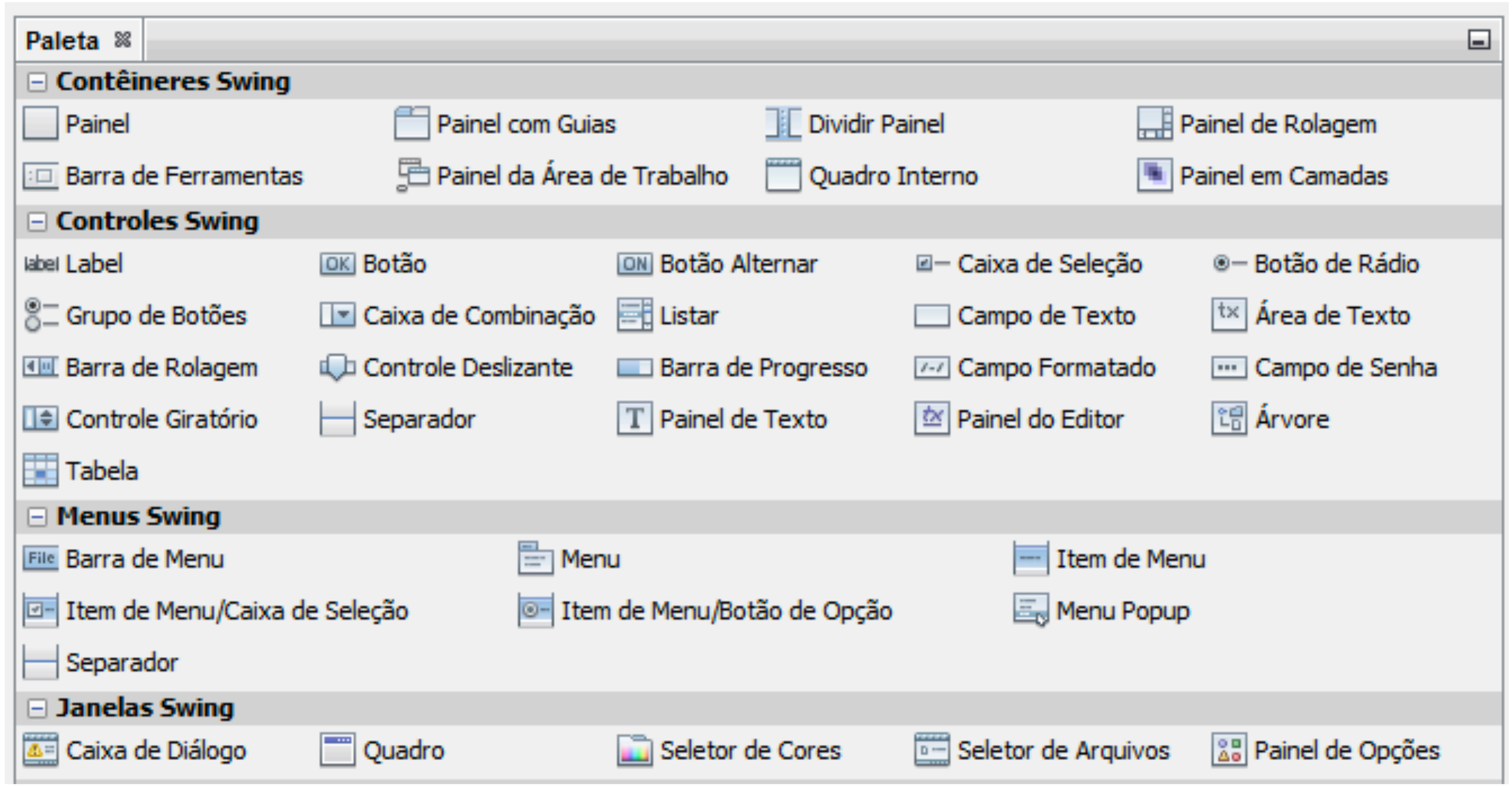
```
try {
    for (LookAndFeelInfo info : UIManager.getInstalledLookAndFeels())
    {
        if ("Nimbus".equals(info.getName()))
        {
            UIManager.setLookAndFeel(info.getClassName());
            break;
        }
    }
} catch (Exception e)
{
    // If Nimbus is not available, set another look and feel.
}
```

Look and Feel

- Nimbus

GUI e NetBeans

GUI e NetBeans



GUI e NetBeans

- Componentes GUI podem ser criados utilizando as ferramentas de edição
- NetBean cria os códigos automaticamente
- Código fica protegido contra edição
  - Caso contrário, o NetBeans não consegue ter controle sobre o processo

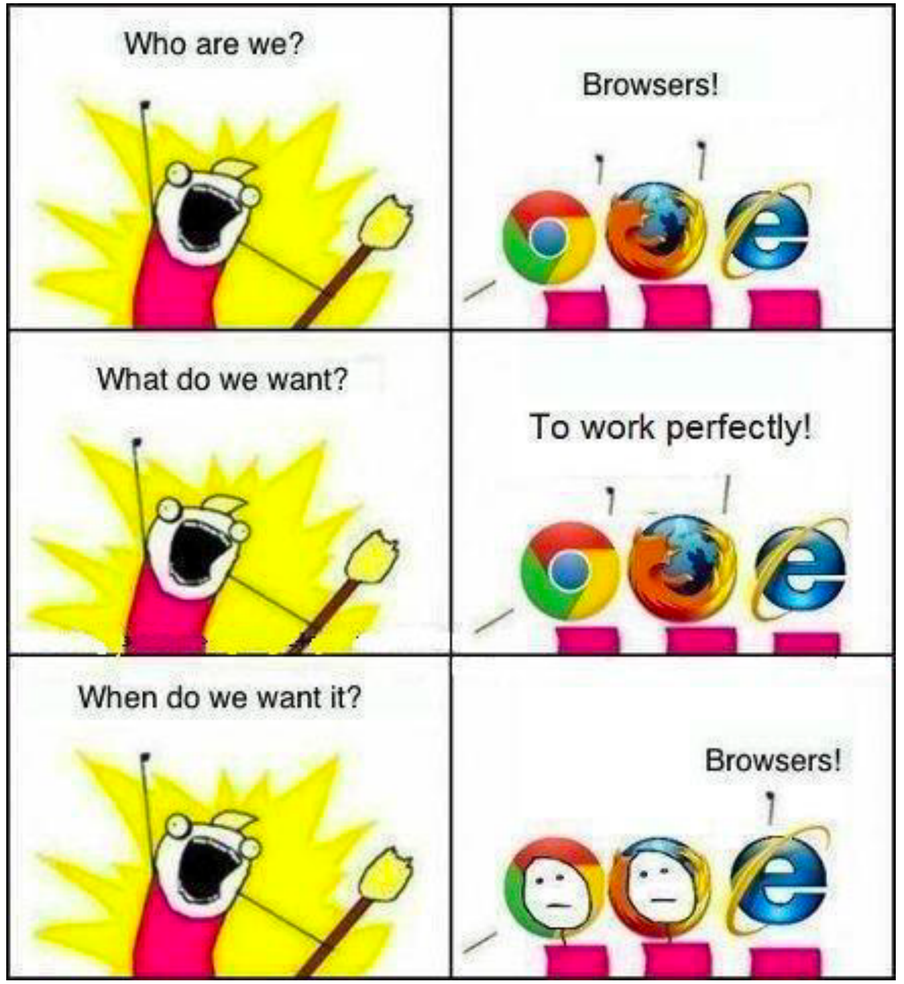
unesp

GUI e NetBeans

- Introdução
- Componentes Swing
- Gerenciadores de Layout
- Look and Feel
- GUI e NetBeans

unesp

Dúvidas?


Who are we?
Browsers!
What do we want?
To work perfectly!
When do we want it?
Browsers!

Bibliografia

- DEITEL, H. M. & DEITEL, P.J. "Java : como programar", Bookman, 2017.
- Material baseado nos slides:
  - Luiz E. Virgilio da Silva. Notas de Aula de Programação Orientada a Objetos (ICMC/USP).
  - José Fernando Junior. Notas de Aula de Programação Orientada a Objetos (ICMC/USP).