Interfaces Gráficas (GUIs) em Java usando Swing

Profa. Flávia Cristina Bernardini

* Baseado em slides de Ricardo Linden, João Bosco Sobral e Samuel Cristhian Schwebel

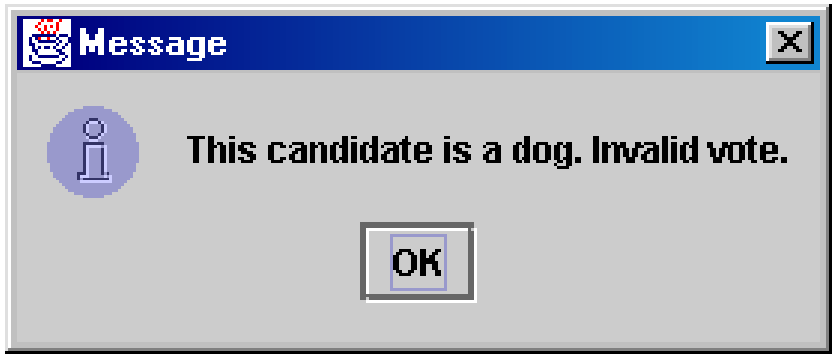
GUI (Graphical User Interface)

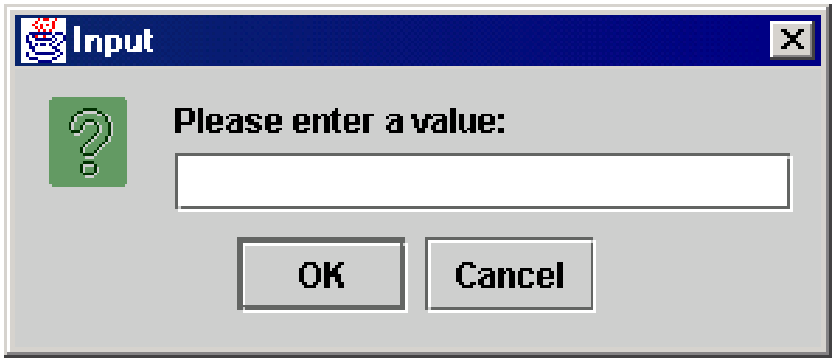
- A interface gráfica com o usuário (GUI - Graphical User Interface) dão, de forma intuitiva, ao usuário um nível básico de familiaridade, sem que jamais tenha usado o programa. Dessa forma, é reduzido o tempo de aprendizado do programa pelo usuário.

- As GUIs são construídas a partir de componentes GUI. O componente GUI é um objeto com o qual o usuário interage através de, por exemplo:
  - Mouse;
  - Teclado;
  - Alguma forma de entrada;
  - Reconhecimento de voz.

Interfaces Mais Simples…

- Nós usamos `javax.swing.JOptionPane`
  - Não é muito OO!
  - Nem muito poderoso...





Elementos de uma GUI

- **componentes**: Elementos desenhados na tela. Exemplos: botão, textbox, label, etc.
- **containers**: elementos que servem como agrupadores lógicos para componentes. Exemplo: Panel.
- **Containers de alto nível**: cidadãos de primeira classe, que efetivamente ocupam um espaço no desktop. Exemplos: Frame, DialogBox.

GUI em Java: AWT e Swing

- Idéia inicial da Sun: **Abstract Windowing Toolkit – AWT**
  - Criar um conjunto de classes e métodos que possam ser usados para escrever uma GUI multi-plataforma
  - Não era poderosa o suficiente, sendo extremamente limitada.

- Segunda edição (JDK v1.2): **Swing**
  - Nova biblioteca, construída do zero que permite gráficos e GUIs muito mais poderosos e flexíveis.

- Por compatibilidade retroativa, ambos existem no Java e, às vezes, nós usamos os dois...

Swing

- Porque o nome **swing**?
  - Apelido usado pelos desenvolvedores das Java Foundations Classes (JFC) usadas para o desenvolvimento do GUI
  - Vem do ícone que era usado em uma release inicial.

- Porque swing e não awt?
  - awt é a versão mais velha e mais primitiva.
  - Swing é mais poderosa
  - Swing pode ser enviado com as aplicações, sendo não dependente de máquina

Look and Feel

- Cada figuta abaixo mostra o mesmo programa com um look and feel diferente.
- Conforme mudamos de SO ou de ambiente, o look and feel se adapta para aquele que é padrão na máquina onde rodamos.
- Isto é uma obrigação da JVM, não do programador.

Componentes Swing

- O Swing fornece vários componentes padrão de GUI que podem ser combinados de forma a criar sua interface.
- Alguns exemplos:
  - Botões, Listas, Menus, Áreas de texto, etc.
- Swing também fornece containers tais como janelas e barras de ferramentas.
- Pode ser de vários níveis:
  - Nível mais alto: frames, diálogos
  - Nível Intermediário: panel, scroll pane, tabbed pane, ...

Hierarquia de Componentes Swing

```
java.lang.Object
  +--java.awt.Component
        +--java.awt.Container
              +--javax.swing.JComponent
              |     +--javax.swing.JButton
              |     +--javax.swing.JLabel
              |     +--javax.swing.JMenuBar
              |     +--javax.swing.JOptionPane
              |     +--javax.swing.JPanel
              |     +--javax.swing.JTextField
              |
              +--java.awt.Window
                    +--java.awt.Frame
                          +--javax.swing.JFrame
```

Métodos comuns em todos os componentes Swing

- `get/setPreferredSize`
- `get/setSize`
- `get/setLocation`
- `getLength/Width`
- `repaint`
- `setBackground(Color)`
- `setFont(Font)`
- `isEnabled / setEnabled(boolean)`
- `isVisible / setVisible(boolean)`

Containers

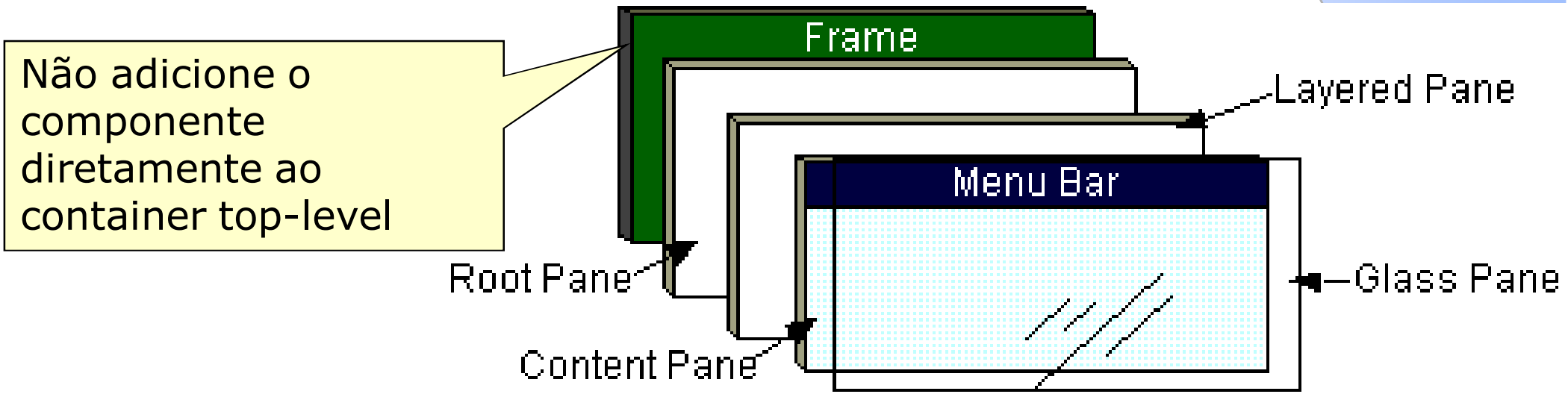
- Descendentes da classe **java.awt.Container**
- Componentes que podem conter outros componentes.
- Usamos um layout manager para posicionar e definir o tamanho dos componentes contidos neles.
- Componentes são adicionados a um container usando as várias formas do método **add**

Containers Top-Level

- Todo programa que usa uma GUI Swing tem pelo menos um container de alto nível (top-level).
- Um container top-level provê o suporte que os componenetes siwng necessitam para realizar o desenho da tela e o tratamento de eventos.
- O Swing fornece três containers top-level :
  - JFrame (Janela principal)
  - JDialog (Janela secundária)
  - JApplet (Um applet mostra uma área desenhada dentro de um navegador internet)

Containers Top-Level

- Para aparecer na tela, todo componente GUI deve ser parte de uma hierarquia de contenção, cuja raiz é um container top-level.
- Todo container top-level tem um content pane que contém todos os componentes visíveis dentro da interface daquele container top-level.



Content Pane

- Todo container top-level contém indiretamente um container intermediário denominado content pane
- Este content pane contém todos os componentes visíveis n GUI da janela.
- Os containers são adicionados ao content pane usando um dos vários tipos de métodos add
- Exemplo:

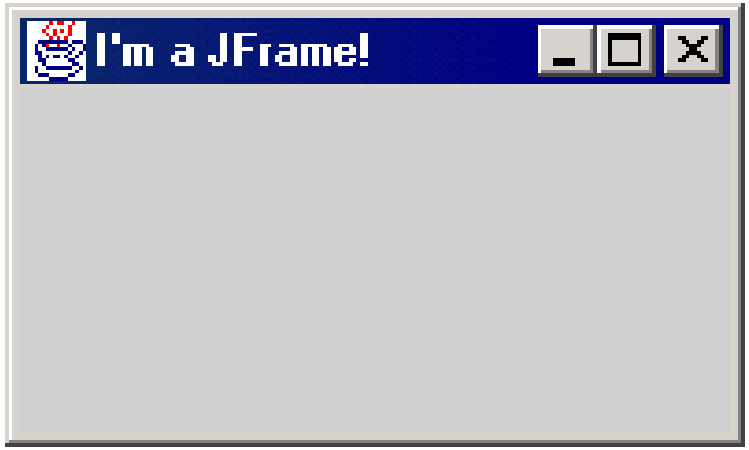
```
frame = new JFrame(...);
label = new JLabel(...);
frame.getContentPane().add(label, BorderLayout.CENTER);
```

JFrame

- Um frame, implementado como uma instância da classe JFrame, é uma janela que tem acessórios tais como borda, título e botões para fechar e minimizá-la.
- Estes acessórios são totalmente dependentes de plataforma.
- As aplicações com uma GUI tipicamente usam ao menos um frame.

JFrame

- Métodos principais
  - `public void setTitle(String title)`: Coloca um título na barra de título da janela.
  - `public void show()`: Faz o frame aparecer na tela.
  - `public void setVisible(boolean v)`: Faz com que o frame se torne visível (`v=true`) ou não (`v=false`).
  - `public void setDefaultCloseOperation(int op)` : Faz a janela realizar uma determinada operação quando fecha. Mais comum: `JFrame.EXIT_ON_CLOSE`



JFrame

- A programação gráfica está muito ligada à ocorrência de eventos, que devem ser tratados
- Ao contrário de outras linguagens de programação, os containers Java não vêm 100% para lidar com os eventos básicos que ocorrem em uma janela.
- Um exemplo é o evento de fechamento.
- Qualquer aplicativo Delphi ou VB sabe fechar sua janela sem problema, mas Java não.
- Mais à frente nós vamos aprender mais sobre eventos, então vamos nos contentar com o `setDefaultCloseOperation` por enquanto

Primeiro programa GUI em Java


```
import javax.swing.*;
public class HelloWorldSwing {
  public static void main(String[] args) {
    JFrame frame = new JFrame("HelloWorldSwing");
    final JLabel label = new JLabel("Hello World");
    frame.getContentPane().add(label);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.pack();
    frame.setVisible(true);
  }
}
```

pack() faz com que a janela seja ajustada para o tamanho preferido de todos os seus sub-componentes.

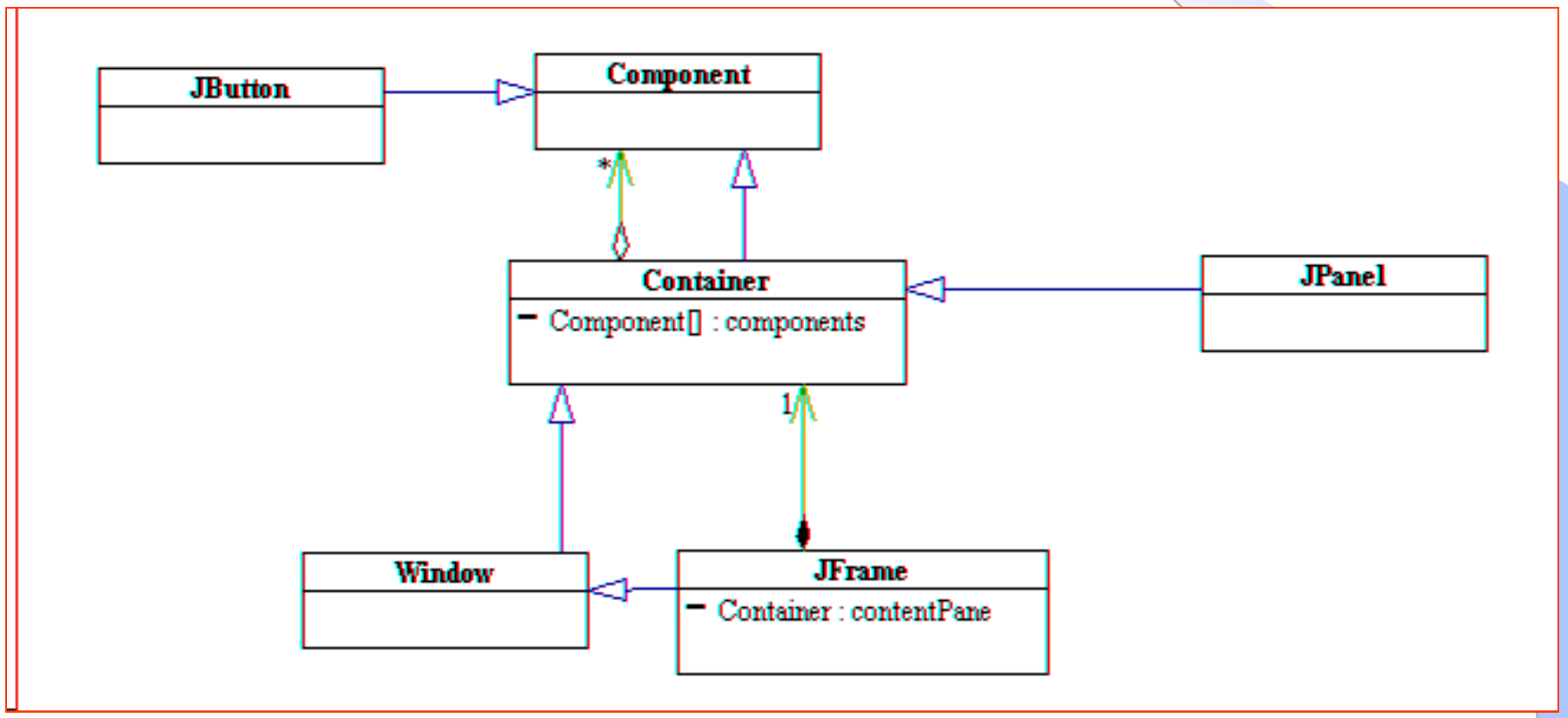
Exemplo 2

Faz a mesma coisa, só que criando uma classe nova, definida por nós .

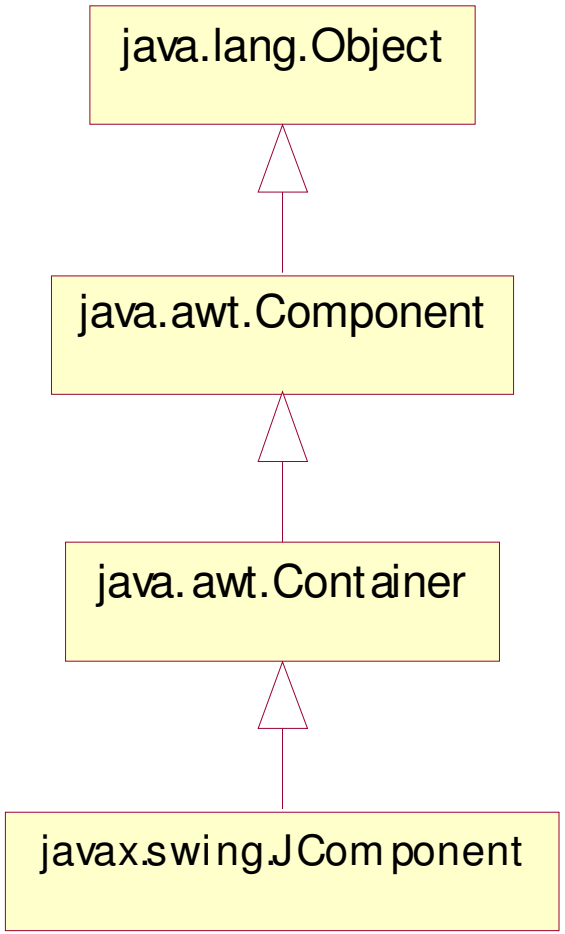
```
import javax.swing.*;
public class HelloWorldFrame extends JFrame {
  public HelloWorldFrame() {
    super(“HelloWorldSwing”);
    final JLabel label = new JLabel("Hello World");
    getContentPane().add(label);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    pack();
    setVisible(true);
  }
  public static void main(String[] args) {
    HelloWorldFrame frame = new HelloWorldFrame();
  }
}
```

Relacionamentos conceituais

- Diagrama UML



Superclasses comuns da maioria dos componentes Swing



JDialog

- Um diálogo é uma maneira de conseguir com que o usuário realize uma entrada de dados.
- Existem vários tipos de diálogos - nós já usamos um há muito tempo...
- Todo diálogo é dependente de um frame.
  - Destruir um frame destrói todos os diálogos que são seus dependentes
  - Quando um frame é minimizado, somem da tela todos os seus diálogos
  - Eles voltam quando o frame volta ao seu tamanho normal
- Um diálogo pode ser **modal**. Quando um diálogo modal está visível, todas as entradas para outros componentes estarão bloqueadas.

JDialog

- Para criar diálogos customizados, use a classe JDialog diretamente.
- O Swing provê vários diálogos padrões:
  - JProgressBar, JFileChooser, JColorChooser, ...
- A classe JOptionPane pode ser usada para criar diálogos modais simples, que permitem a costumização de ícones, títulos, textos e botões

Usando JOptionPane

- `JOptionPane` faz com que seja fácil exibir um diálogo padrão que leia um valor do usuário ou informe-o de algo.

- A classe `JOptionPane` parecer complexa dado o seu grande número de métodos.

- Entretanto, a maioria dos uso desta classe são simplesmente chamadas de uma única linha para um dos métodos estáticos `showXxxDialog`

showXxxDialog

- showConfirmDialog ⇨ Faz uma pergunta confirmatória e espera uma resposta tal como yes/no/cancel.
- showInputDialog ⇨ Lê entrada do teclado (velho conhecido)
- showMessageDialog ⇨ Informa ao usuário que algo aconteceu.
- showOptionDialog ⇨ Os três anteriores em um só

⇨**Todos os diálogos acima são modais.**

Usando JOptionPane: Exemplo

```
Object[] options = {”Sim!", "Não",”Quem sabe?"};
int n = JOptionPane.showOptionDialog(
        frame, ”Tem certeza?”
        ”Confirmação",
        JOptionPane.YES_NO_CANCEL_OPTION,
        JOptionPane.QUESTION_MESSAGE,
        null,
        options,
        options[2]);
```

Container

- Um container é um objeto que armazena componentes, governando suas posições, tamanhos e comportamentos quando o usuário altera seu tamanho.
- Principais métodos:
  - public void add(Component comp)
  - public void add(Component comp, Object O): adiciona um componente usando informações adicionais (veremos mais adiante).
  - public void remove(Component comp)
  - public void setLayout(LayoutManager mgr): Usa o layout manager especificado para posicionar os componentes no container.
  - public void validate(): Manda o layout manager reposicionar todos os objetos no container.

JComponent

- Classe base para todos os componentes Swing, com exceção dos containers top-level
- Para usar um componente que herde de JComponent, ele deve ser colocado na hierarquia de contenção cuja raiz seja um container top-level
- A classe JComponent provê, entre outros, tratamento de eventos de teclado, look and feel, infra-estrutura para desenho, suporte a bordas, etc
- Todos os descendentes de JComponent também são Containers. Exemplo: um JButton pode conter textos, ícones, etc

JComponent

- Componentes típicos
  - JLabel
  - JButton
  - JTextField
  - JPanel
  - JTable
  - Etc.

JButton

- Um botão é uma região clicável com a qual o usuário interage de forma a realizar um comando.

- Principais métodos:
  - `public JButton(String text)`: Cria um novo botão com o texto dado como parâmetro definido como texto de face.
  - `public String getText()`:Retorna o texto mostrado no botão.
  - `public void setText(String text)`: Muda o texto de face do botão.

JButton

- Apesar de termos criado um botão, ele ainda não faz nada.
- Mesmo clicando nele, nada ocorre
- Isto ocorre porque nós não associamos nenhum **tratador de evento** a este botão
  - Tratamento de eventos: veremos adiante

JLabel

- Um label é um texto a ser mostrado na tela que normalmente oferece informação para o usuário de forma a tornar a interface mais compreensível.

- Principais métodos:
  - public JLabel(String text): Cria um novo label com o texto dado como parâmetro definido como texto de face.
  - public String getText():Retorna o texto mostrado no label.
  - public void setText(String text): Muda o texto de face do label.

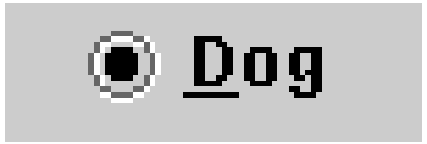
JTextField

- Um textfield é como um label, só que pode ser editado e modificado pelo usuário.
- Textfields são usados geralmente para entrada de dados pelo usuário.
- Métodos interessantes:
  - `public JTextField(int columns)`: Cria um novo textfield com uma largura, em caracteres, dada pelo parâmetro.
  - `public String getText()`: Retorna o texto atualmente armazenado no campo.
  - `public void setText(String text)`: Modifica o texto do campo para ser igual à string dada como parâmetro.

JCheckBox

- Objeto de tela que permite que escolhamos entre duas opções (marcado e não marcado)
- Métodos interessantes:
  - public JCheckBox(String text): Cria uma checkbox cujo texto é dado pelo parâmetro e cujo valor default é não marcada
  - public JCheckBox(String text, boolean isChecked)
    Cria uma checkbox cujo texto é dado pelo primeiro parâmetro e cujo valor default é dado pelo segundo parâmetro.
  - public boolean isSelected():Retorna true se a check box foi selecionada.
    public void setSelected(boolean selected): Muda o estado da checkbox de acordo com o parâmetro passado.

JRadioButton



- Um botão que pode ser selecionado.
- Normalmente é parte de um grupo de botões mutuamente exclusivos (isto é, apenas um pode ser selecionado de cada vez)
- Métodos interessantes
  - `public JRadioButton(String text)`
  - `public JRadioButton(String text, Icon icon)`
  - `public boolean isSelected()`
  - `public void setSelected(boolean b)`

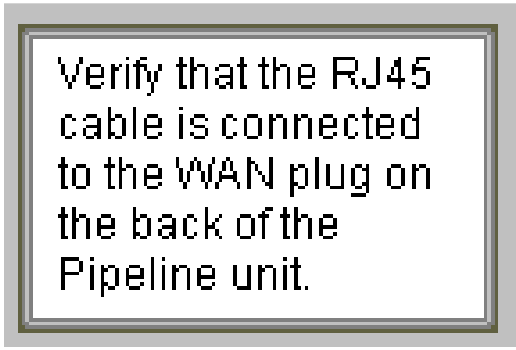
ButtonGroup

- Um grupo lógico de radiobuttons que garante que apenas um deles pode ser selecionado de cada vez.
- Métodos interessantes:
  - `public ButtonGroup()`
  - `public void add(AbstractButton b)`
- O objeto da classe ButtonGroup não é um objeto gráfico, apenas um grupo lógico.
- Os objetos da classe RadioButton é que são adicionados ao container

Usando Icon

- Nas classes `JButton`, `JRadioButton`, `JCheckBox`, `JLabel`, etc, temos o seguinte.
  - O construtor pode receber um ícone
  - `public void setIcon(Icon)`
  - `public void setSelectedIcon(Icon)`
  - `public void setRolloverIcon(Icon)`

JTextArea



- Uma área de texto multi-linha
- Útil para escrever textos longos que não necessitam formatação.
- Métodos interessantes:
  - `public JTextArea(int rows, int columns)`: Cria um novo campo de texto com um tamanho determinado em termos de número de linhas e colunas
  - `public String getText()`:Retorna o texto armazenado no momento na área.
  - `public void setText(String text)`:Define o texto a ser armazenado na área como sendo o parâmetro passado.

Problema: Posicionando componentes

- Como o programador pode especificar onde fica cada componente na janela, quão grande ele deve ser e o que ele deve fazer quando a janela muda de tamanho?
  - Em C++, Delphi e outros: usar posicionamento absoluto.
    - O programador especifica as coordenadas de cada componente.
  - Em Java : usamos os gerenciadores de formato (Layout Managers).
    - São objetos especiais que decidem onde posicionar cada componente baseados em critérios específicos.

Bordas

- Todo JComponent pode ter uma ou mais bordas
- A classe usada para criar bordas padronizadas é a BorderFactory
- Exemplo de uso:

```
p.setBorder(BorderFactory.createLineBorder(Color.black));
```

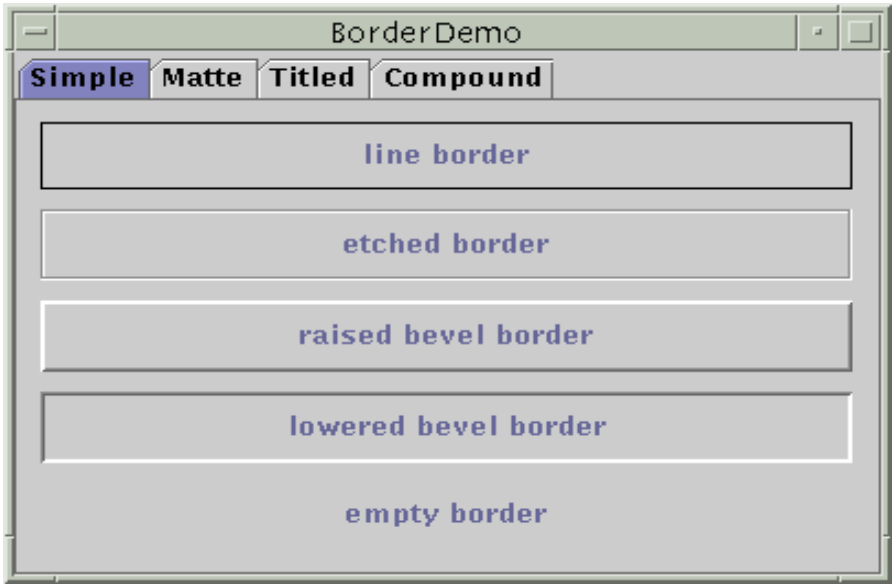
- Podemos criar uma borda composta, combinando uma ou mais bordas, da seguinte maneira:

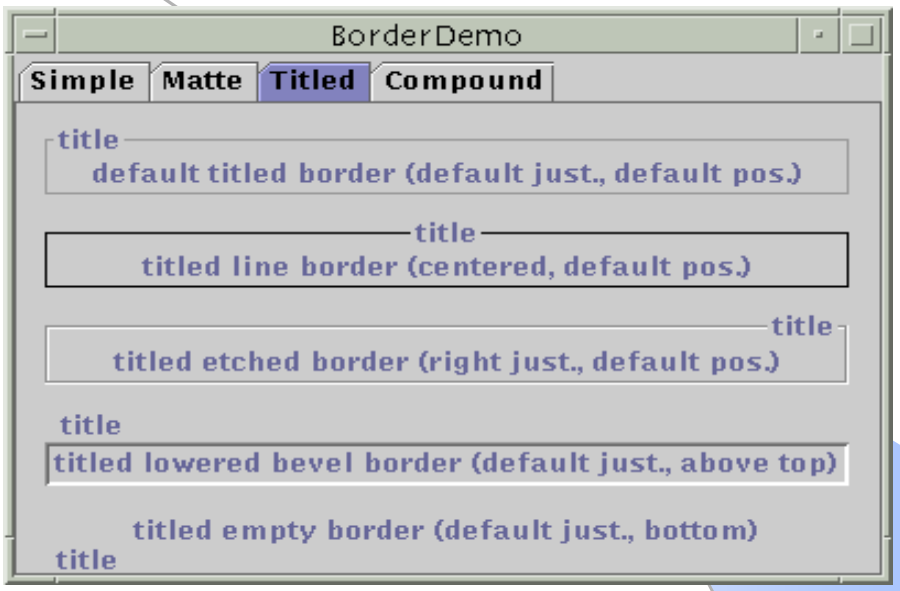
```
BorderFactory.createCompoundBorder(border1, border2);
```

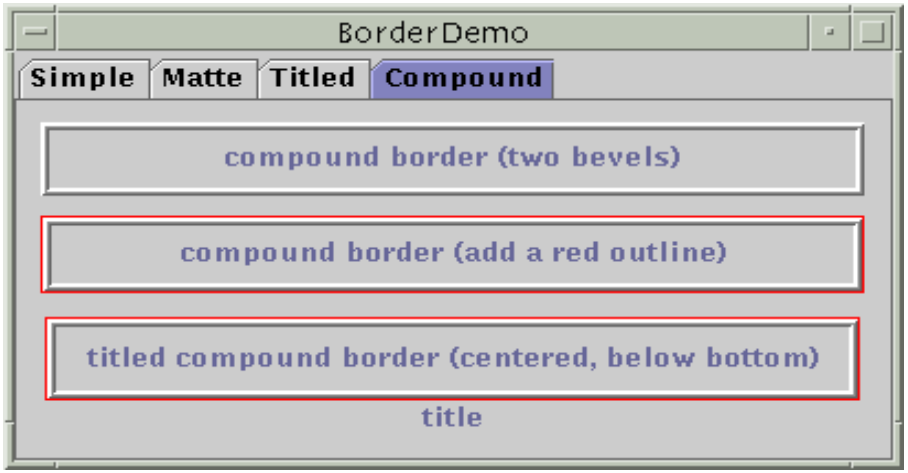
Classe BorderFactory

- Hierarquia
  java.lang.Object
  ↙ **javax.swing.BorderFactory**
- Servem para implementar bordas padronizadas.
- Use métodos estáticos que retornam elementos da classe Border:
  - **createBevelBorder**
  - **createEtchedBorder**
  - **createTitledBorder**
  - **createEmptyBorder**
  - **createLineBorder**
  - **etc.**

Tipos de bordas







Containers de nível intermediário

- Também conhecidos como panels ou panes
- Simplificam o posicionamento de outros componentes, como no caso do JPanel
- Têm um papel visível e interativo na GUI do programa, como no caso do JScrollPane e do JTabbedPane.
- Possuem um gerenciador de layout padrão, que é o FlowLayout.
- Este gerenciador pode ser modificado com o comando:
  - **panel.setLayout(new BorderLayout());**

Containers de nível intermediário

- Por default, os panels não pintam nada, exceto o seu fundo (background).
- Por default, eles são opacos.
- Um panel opaco pode ser definido como o content pane de um container de top-level.
- Panels opacos não desenham um background.

JPanel

- Um panel é um container que podemos usar (além do JFrame)

- Principais métodos:
  - public JPanel(): Cria um JPanel com o layour manager padrão
  - public JPanel(LayoutManager mgr): Cria um Jpanel com o layour manager especificado.

- Um JPanel pode depois ser acrescentado em um Jframe para melhorar o layout de nossas telas.

Resumo

- JFrame = É um contêiner (formulário) para outros componentes GUI.
- JLabel = Área em que podem ser exibidos textos não-editáveis ou ícones.
- JTextField = Área em que o usuário insere dados pelo teclado.
- JButton = Área que aciona um evento quando o usuário clica.
- JCheckBox = Possui dois estados: selecionado ou não-selecionado.
- JComboBox = Lista de itens que o usuário pode fazer uma seleção clicando em um item na lista ou digitando na caixa.
- JList = Área em que uma lista é exibida, possibilitando a seleção clicando em qualquer item da lista.
- JPanel = Contêiner em que os componentes podem ser colocados.

Alguns Componentes GUI Básicos