

# Interface Gráfica em Java - *Swing*

# GUI – *Graphical User Interface*



## ■ GUI – Graphical User Interface

- Apresenta um mecanismo amigável ao usuário para interagir com um aplicativo
- Interfaces consistentes permitem que o usuário aprenda mais rapidamente novos aplicativos

## ■ Componentes GUI

- Também conhecidos como controles ou widgets (*window gadgets*)
- São objetos com que o usuário interage (mouse, teclado, etc)

# GUI – *Graphical User Interface*



- Java possui três bibliotecas principais para GUI:
  - AWT
  - Swing
  - JavaFX
- *AWT - Abstract Window Toolkit*
  - Biblioteca original do Java para GUIs
- **Swing**
  - Biblioteca mais sofisticada e completa, desenvolvida para substituir o AWT
- **JavaFX**
  - Biblioteca mais completa e complexa que o Swing
  - Desenvolvido para substituir o Swing em aplicações de grande porte

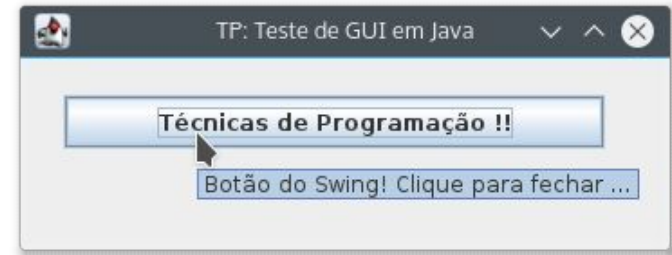
# Swing



- Por ser mais simples e ainda ser completo, neste curso iremos focar na utilização do Swing para GUIs

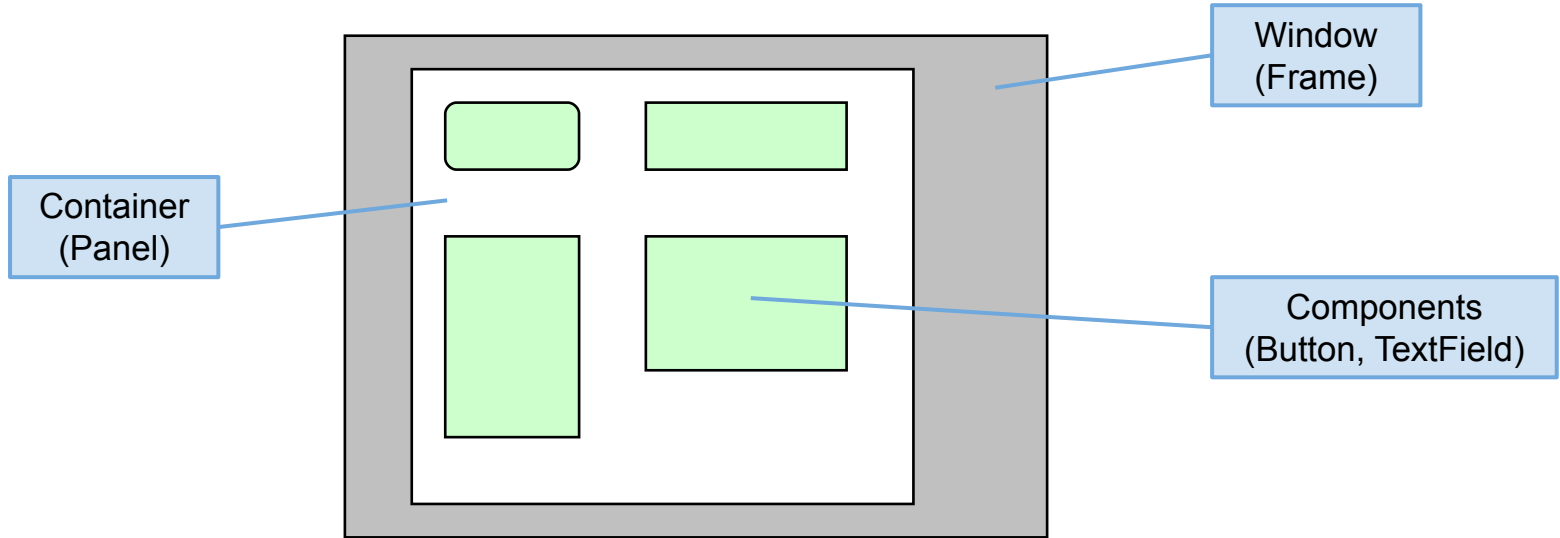
# Swing

- Swing é um toolkit que inclui um extenso conjunto de componentes para construir GUIs e adicionar interatividade em aplicações Java
  - Swing é parte do *Java Foundation Classes* (JFC)
  - Inclui componentes como: botões, caixas de texto, caixas de seleção, tabelas, listas, estruturas de árvores
  - Até mesmo o suporte a internacionalização e à acessibilidade é possível
  - A aplicação a ser executada em sistemas gráficos diferentes:
    - SOs: *Linux, Mac, Windows*



# Componentes GUI

- Componentes podem ser divididos em três partes:



- Ao se criar uma Janela (*Window/Frame*), automaticamente um *Container (Panel)* é criado para a nova janela.

# Criando uma Nova Janela

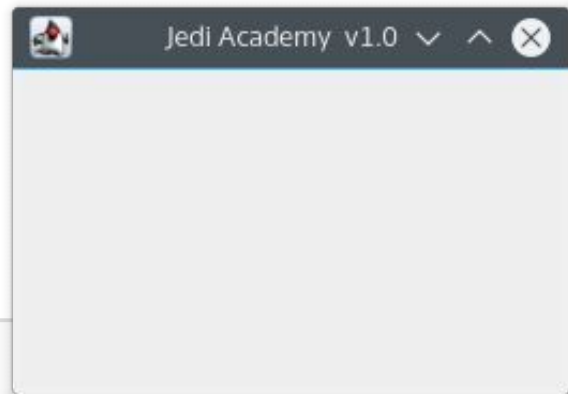
- Para criar uma nova janela, cria-se uma classe que herda a classe `JFrame`

```
import java.awt.*;
import javax.swing.*;

public class JediAcademy extends JFrame {

    public JediAcademy() {
        super("Jedi Academy v1.0");
        this.setLayout(null);
        this.setSize(260, 180);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    public static void main(String args[]) {
        JediAcademy mainWindow = new JediAcademy();
        // Preencha aqui os dados da janela ..
        mainWindow.setVisible(true);
    }
}
```



Título da janela

Layout (próximos slide)

Tamanho da janela

Ao clicar no botão fechar

Torna a janela visível

# Criando uma Nova Janela (2a Opção)

- Cria-se um objeto da classe JFrame diretamente

```
import java.awt.*;
import javax.swing.*;

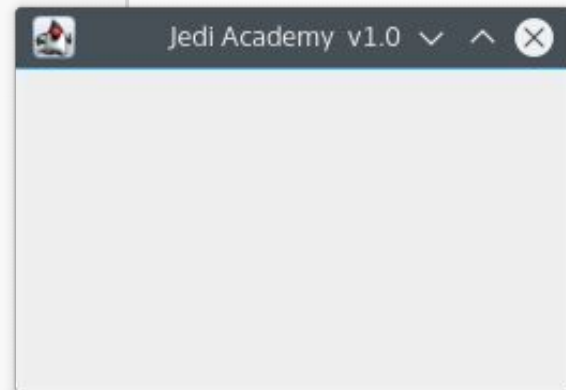
public class JediAcademy2 {

    private JFrame frame;

    public JediAcademy2() {
        frame = new JFrame("Jedi Academy v1.0");
        frame.setSize(260, 180);
        frame.setLayout(null);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    public static void main(String args[]) {
        JediAcademy2 mainWindow = new JediAcademy2();
        mainWindow.frame.setVisible(true);
    }
}
```

Objeto JFrame





# Criando um Texto (Label)

## ■ Usa-se a classe JLabel

```
// Início da classe ..

public JediAcademy() {
    super("Jedi Academy v1.0");
    this.setLayout(null);
    this.setSize(260, 180);
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    JLabel lTitulo = new JLabel("Jedi Academy v1.0");
    lTitulo.setBounds(0, 10, 260, 20);
    lTitulo.setHorizontalAlignment(SwingConstants.CENTER);
    this.add(lTitulo);

    JLabel lDesc = new JLabel("Sistema de Controle e Gerenciamento");
    lDesc.setBounds(0, 25, 260, 20);
    lDesc.setHorizontalAlignment(SwingConstants.CENTER);
    lDesc.setFont(new Font("Dialog", Font.ITALIC, 8));
    this.add(lDesc);
}

// Main ..
```



Objeto JLabel

X, Y, Largura, Altura

Centraliza o texto

Adiciona o componente  
ao container da janela

# Criando Botões

## ■ Usa-se a classe JButton

```
// Início da classe e do construtor ...
```

```
JButton bGer = new JButton("Gerenciar Jedi Initiates");  
bGer.setBounds(20, 55, 220, 25);  
this.add(bGer);
```

```
JButton bRel = new JButton("Relatórios de Controle");  
bRel.setBounds(20, 85, 220, 25);  
this.add(bRel);
```

```
JButton bSobre = new JButton("Sobre o Sistema");  
bSobre.setBounds(20, 115, 220, 25);  
this.add(bSobre);
```

```
// Fim do construtor e main ...
```



Objeto JButton

X, Y, Largura, Altura

Adiciona o componente  
ao container da janela

# Manipulando Eventos

- Tratamento de eventos são feitos através do `ActionListener`

- `ActionListener` é uma interface e seu único método é:

```
void actionPerformed(ActionEvent e);
```

- O método `actionPerformed` irá ser executado quando um evento ocorrer. O objeto “e”, da classe `ActionEvent`, terá as informações do evento (tipo, descrição, etc).

# Manipulando Eventos

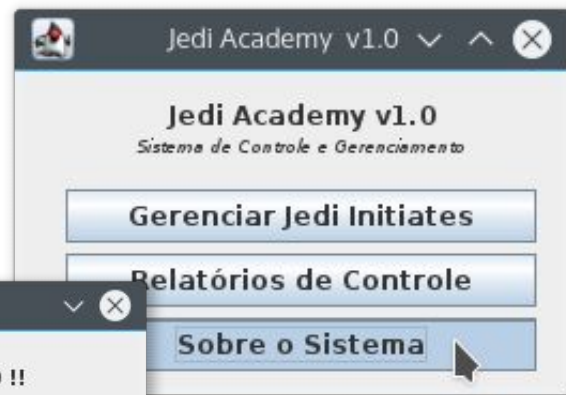
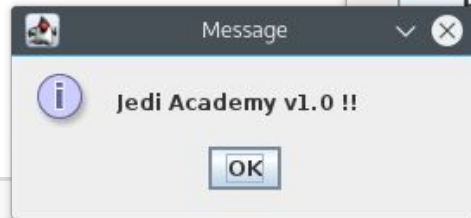


- Para não precisar criar uma classe nova apenas para tratar um evento (uma vez que o `ActionListener` é uma interface), em geral classes anônimas internas são utilizadas:
  - *Anonymous Inner Classes*
  - Nas classes anônimas internas, uma classe é declarada, seus métodos implementados, ela é instanciada e um objeto da classe (que nem tem nome) é retornado, tudo isso no mesmo bloco de código

# Manipulando Eventos

- Declaração e instanciação de uma classe que implementa o ActionListener

```
// import java.awt.event.*;  
// Início da classe, do construtor, e criação dos botões ..  
bSobre.addActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent e) {  
        JOptionPane.showMessageDialog(null, "Jedi Academy v1.0 !!");  
    }  
});  
// Fim do construtor e main ..
```



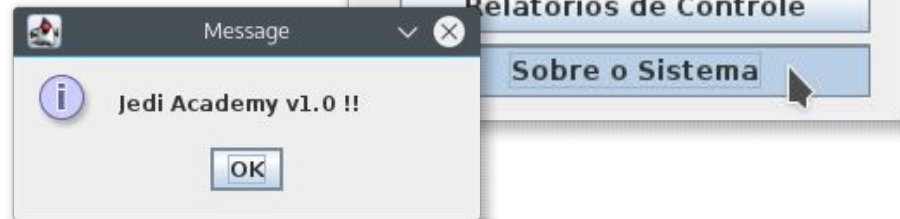
Classe interna anônima

Nova instância da classe

Implementação do evento

# Manipulando Eventos

- Faz o mesmo do slide anterior, mas usando expressões *lambda*, que simplificam o uso de classes internas anônimas



```
// import java.awt.event.*;

// Início da classe, do construtor, e criação dos botões ..
bSobre.addActionListener((ActionEvent e) -> {
    JOptionPane.showMessageDialog(null, "Jedi Academy v1.0 !!");
});

// Fim do construtor e main ..
```

Expressão *lambda*

Implementação do evento

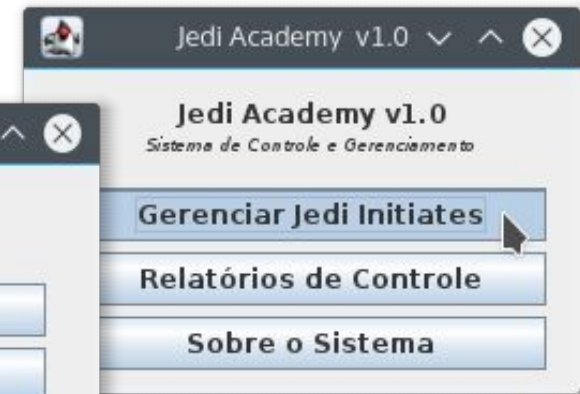
# Criando uma 2a Janela

```
public class ManagerWindow extends JFrame {  
  
    public ManagerWindow() {  
        super("Jedi Initiates Management");  
        this.setLayout(null); this.setSize(260, 180);  
        this.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);  
  
        JLabel lTitulo = new JLabel("Jedi Academy v1.0");  
        lTitulo.setBounds(0, 10, 260, 20);  
        lTitulo.setHorizontalAlignment(SwingConstants.CENTER);  
        this.add(lTitulo);  
  
        JLabel lDesc = new JLabel("Gerenciamento de Jedi Initiates");  
        lDesc.setBounds(0, 25, 260, 20);  
        lDesc.setHorizontalAlignment(SwingConstants.CENTER);  
        lDesc.setFont(new Font("Dialog", Font.ITALIC, 8));  
        this.add(lDesc);  
  
        JButton bGer = new JButton("Adicionar");  
        bGer.setBounds(20, 55, 220, 25);  
        this.add(bGer);  
  
        // Outros botões ..  
    }  
}
```



# Abrindo a 2a Janela

- Implementando o evento do botão “Gerenciar”



```
// Início da classe, do construtor, e criação dos botões ..

bGer.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent e) {
        ManagerWindow managerWindow = new ManagerWindow();
        managerWindow.setVisible(true);
    }

});

// Fim do construtor e main ..
```



# Layouts



- Um *Layout* define como os componentes adicionados no container ficarão posicionados
  - Nos últimos exemplos não usamos Layout e tivemos que setar a posição/tamanho dos componentes manualmente
    - *Conhecido como Layout Absoluto*
    - *Não muda quando a janela é redimensionada*
  - Mas Layouts nos permitem um maior controle e dinamicidade do posicionamento dos componentes na janela
  - Principais Layouts:
    - *FlowLayout*,
    - *BorderLayout*
    - *GridLayout*

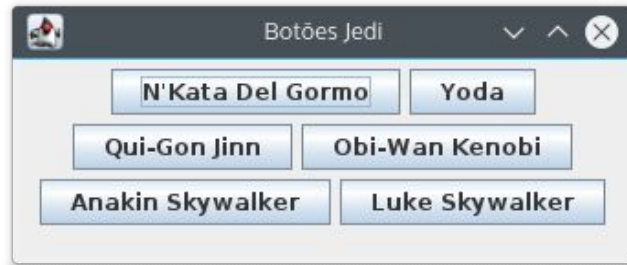
# Layouts

## ■ FlowLayout: usa a área disponível

```
import java.awt.*;
import javax.swing.*;

class JediButtons {

    public static void main(String[] args) {
        JFrame janela = new JFrame("Botões Jedi");
        janela.setLayout(new FlowLayout());
        janela.add(new JButton("N'Kata Del Gormo"));
        janela.add(new JButton("Yoda"));
        janela.add(new JButton("Qui-Gon Jinn"));
        janela.add(new JButton("Obi-Wan Kenobi"));
        janela.add(new JButton("Anakin Skywalker"));
        janela.add(new JButton("Luke Skywalker"));
        janela.setSize(340, 140);
        janela.setVisible(true);
        janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```



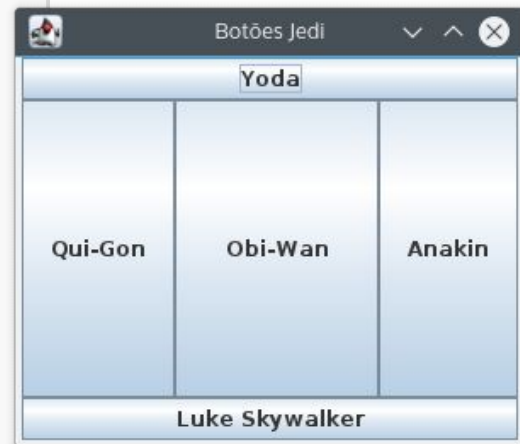
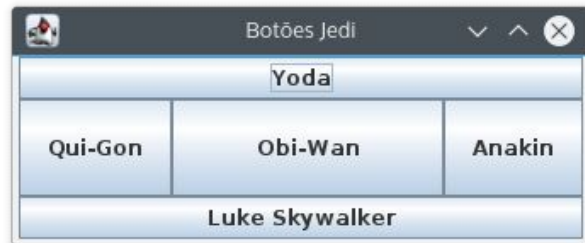
# Layouts

- BorderLayout: usa pontos cardeais

```
import java.awt.*;
import javax.swing.*;

class JediButtons {

    public static void main(String[] args) {
        JFrame janela = new JFrame("Botões Jedi");
        janela.setLayout(new BorderLayout());
        janela.add(new JButton("Yoda"), BorderLayout.NORTH);
        janela.add(new JButton("Qui-Gon"), BorderLayout.WEST);
        janela.add(new JButton("Obi-Wan"), BorderLayout.CENTER);
        janela.add(new JButton("Anakin"), BorderLayout.EAST);
        janela.add(new JButton("Luke Skywalker"), BorderLayout.SOUTH);
        janela.setSize(340, 140);
        janela.setVisible(true);
        janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```



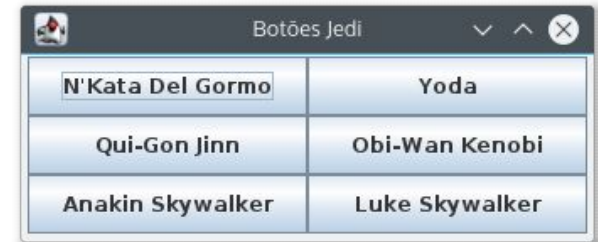
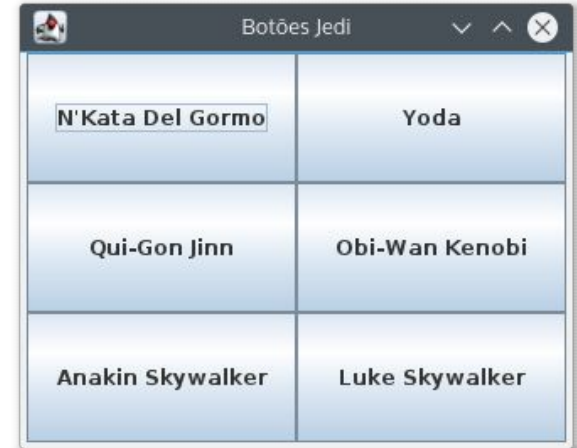
# Layouts

## ■ GridLayout: posiciona elementos em tabela

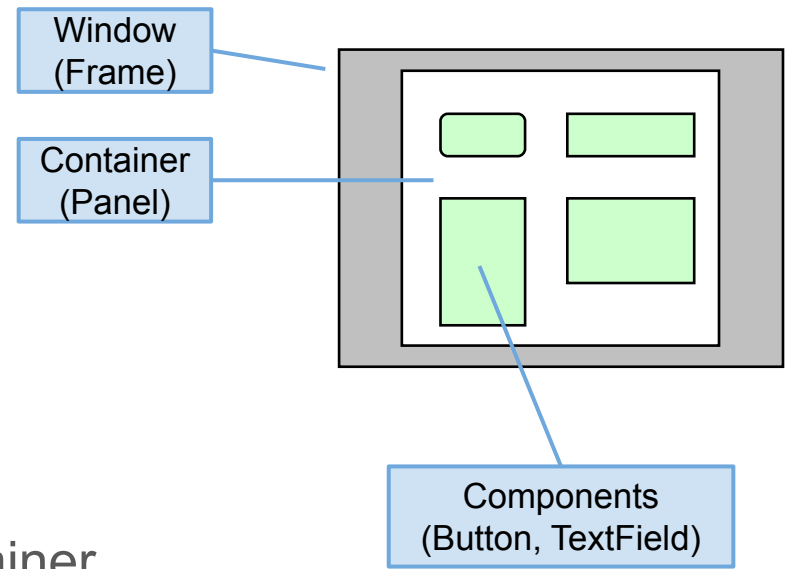
```
import java.awt.*;
import javax.swing.*;

class JediButtons {

    public static void main(String[] args) {
        JFrame janela = new JFrame("Botões Jedi");
        janela.setLayout(new GridLayout(3, 2));
        janela.add(new JButton("N'Kata Del Gormo"));
        janela.add(new JButton("Yoda"));
        janela.add(new JButton("Qui-Gon Jinn"));
        janela.add(new JButton("Obi-Wan Kenobi"));
        janela.add(new JButton("Anakin Skywalker"));
        janela.add(new JButton("Luke Skywalker"));
        janela.setSize(340, 140);
        janela.setVisible(true);
        janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```



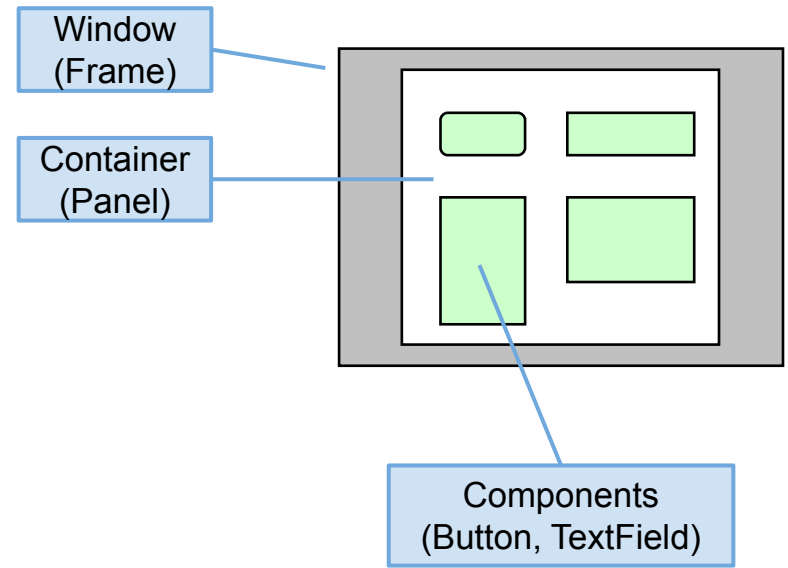
# Containers / Panels



- Conforme mencionado no início, os componentes são colocados em um container
- Quando criamos uma janela, um container é criado automaticamente e os métodos `setLayout()` e `add()` executados na janela são “repassados” para o container
  - Portanto, os Layouts citados anteriormente, fazem parte do container, e não da janela

# Containers / Panels

- Um container pode possuir outros containers, que possuirão seus próprios layouts e componentes.
- Para criar um novo container:
  - Cria-se um objeto da classe `JPanel`
  - Adiciona-se o objeto a um container (ou janela) já existente



# Containers

```
import java.awt.*;  
import javax.swing.*;
```

```
class JediButtons {
```

```
    public static void main(String[] args) {  
        JFrame janela = new JFrame("Botões Jedi");  
        janela.setSize(340, 140);  
        janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

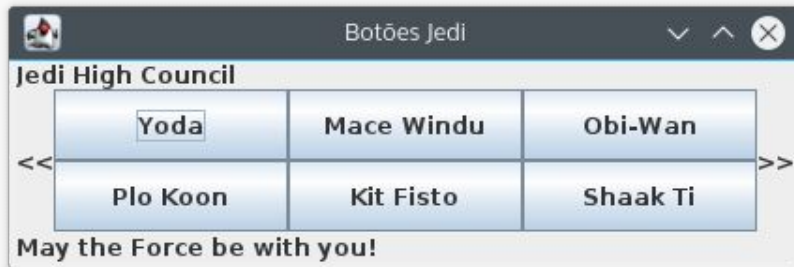
```
        janela.setLayout(new BorderLayout());  
        janela.add(new JLabel("Jedi High Council"), BorderLayout.NORTH);  
        janela.add(new JLabel("<<"), BorderLayout.WEST);  
        janela.add(new JLabel(">>"), BorderLayout.EAST);  
        janela.add(new JLabel("May the Force be with you!"), BorderLayout.SOUTH);
```

```
        JPanel pCenter = new JPanel();  
        pCenter.setLayout(new GridLayout(2, 3));  
        pCenter.add(new JButton("Yoda")); pCenter.add(new JButton("Mace Windu"));  
        pCenter.add(new JButton("Obi-Wan")); pCenter.add(new JButton("Plo Koon"));  
        pCenter.add(new JButton("Kit Fisto")); pCenter.add(new JButton("Shaak Ti"));  
        janela.add(pCenter, BorderLayout.CENTER);
```

```
        janela.setVisible(true);
```

```
    }
```

```
}
```



Layout principal da Janela

Novo container para o "centro" usando o GridLayout

Componentes do container

Adiciona o novo container ao container da janela

# Caixas de Diálogo

## ■ Entrada e saída rápida de dados

```
import java.awt.*;
import javax.swing.*;

class TesteDialogos {

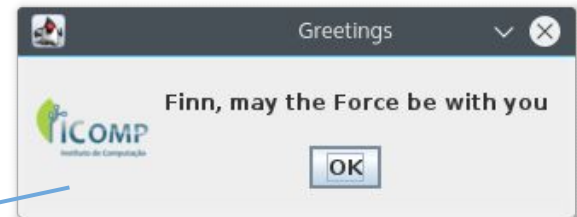
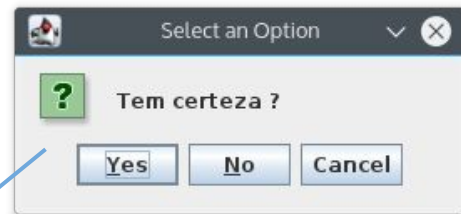
    public static void main(String[] args) {

        String nome =
            JOptionPane.showInputDialog("Digite seu nome:");

        int confirm =
            JOptionPane.showConfirmDialog(null, "Tem certeza ?");

        JOptionPane.showMessageDialog(null,
            nome + ", may the Force be with you",
            "Greetings", JOptionPane.INFORMATION_MESSAGE,
            new ImageIcon("/tmp/icom2.png")
        );

    }
}
```





# Principais Componentes

- Todos os componentes possuem:
  - Uma posição e tamanho (no Absolute Layout)

```
botao.setBounds(0, 0, 100, 20);
```

- Uma cor e uma cor de fundo

```
label.setBackground(new Color(0, 0, 255));  
label.setForeground(new Color(255, 0, 0));
```

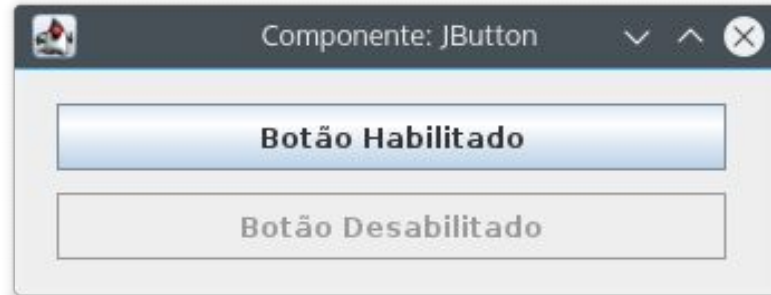
- Um tooltip (dica/mensagem de ajuda)

```
botao.setTooltipText("Use the Force ...");
```

# Principais Componentes

- Componentes podem ser habilitados ou desabilitados

```
botao1.setEnabled(true);  
  
botao1.setEnabled(false);
```



# Principais Componentes: JLabel

## ■ JLabel



Um "label" só  
com imagem

```
new JLabel("Exemplo de uso do JLabel");  
new JLabel(new ImageIcon("/home/disciplinas/icom.png"));
```

```
label.setFont(new Font("Dialog", Font.ITALIC, 8));  
label.setHorizontalAlignment(SwingConstants.CENTER);  
label.setIcon(new ImageIcon("/home/disciplinas/icom.png"));  
label.setText("Yoda");  
label.getText();
```

# Principais Componentes: JLabel

## ■ JLabel: texto, figura, fundo

```
class TesteComponentes {  
    public static void main(String[] args) {  
        JFrame janela = new JFrame("Componente: JLabel");  
        janela.setLayout(null);  
        janela.setSize(345, 130);  
        janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
  
        janela.setContentPane(new JLabel(new ImageIcon("/tmp/fundo.png")));  
  
        JLabel comp = new JLabel("Exemplo de uso do JLabel");  
        comp.setBounds(20, 10, 280, 15); janela.add(comp);  
  
        JLabel comp2 = new JLabel(new ImageIcon("/tmp/icom.png"));  
        comp2.setBounds(20, 30, 118, 65); janela.add(comp2);  
  
        janela.setVisible(true);  
    }  
}
```



# Principais Componentes: JButton

## ■ JButton

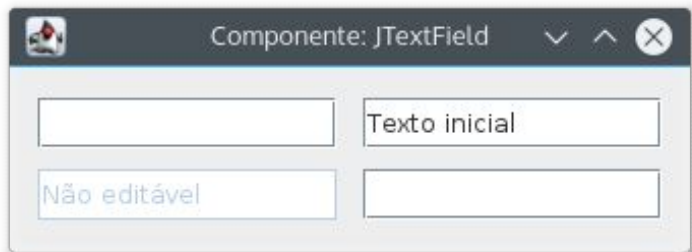


```
new JButton("Botão");  
new JButton("Botão", new ImageIcon("/tmp/icom.png"));  
new JButton(new ImageIcon("/tmp/icom.png"));
```

```
button.setFont(new Font("Dialog", Font.ITALIC, 8));  
button.setHorizontalAlignment(SwingConstants.CENTER);  
button.setHorizontalTextPosition(SwingConstants.CENTER);  
button.setIcon(new ImageIcon("/home/teste/icom.png"));  
button.setText("Botão com Figura");  
button.getText();
```

# Principais Componentes: JTextField

- JTextField
- JPasswordField



```
new JTextField();  
new JTextField("Texto inicial");  
new JPasswordField();
```

```
textField.setEditable(false);  
textField.setText("Texto Inicial");  
textField.getText(); // Retorna o texto da caixa  
textField.getSelectedText();  
textField.setFont(new Font("Dialog", Font.ITALIC, 8));  
textField.setHorizontalAlignment(SwingConstants.CENTER);
```

# Principais Componentes

## ■ JComboBox



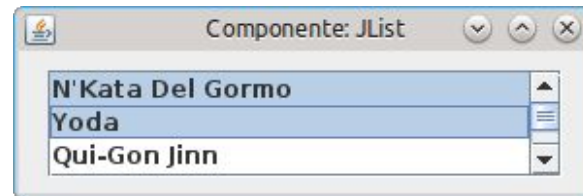
## ■ JCheckBox



## ■ JRadioButton

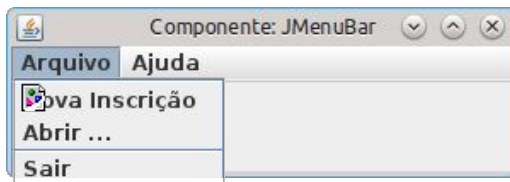


## ■ JList + JScrollPane

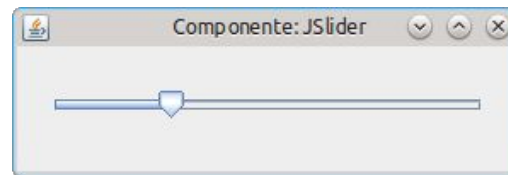


# Principais Componentes

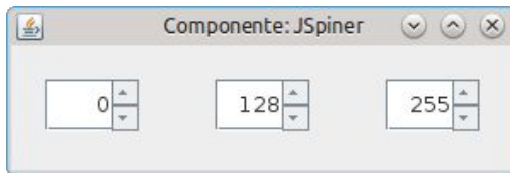
## ■ JMenuBar



## ■ JSlider



## ■ JSpinner



## ■ JTree





# Usando o Eclipse

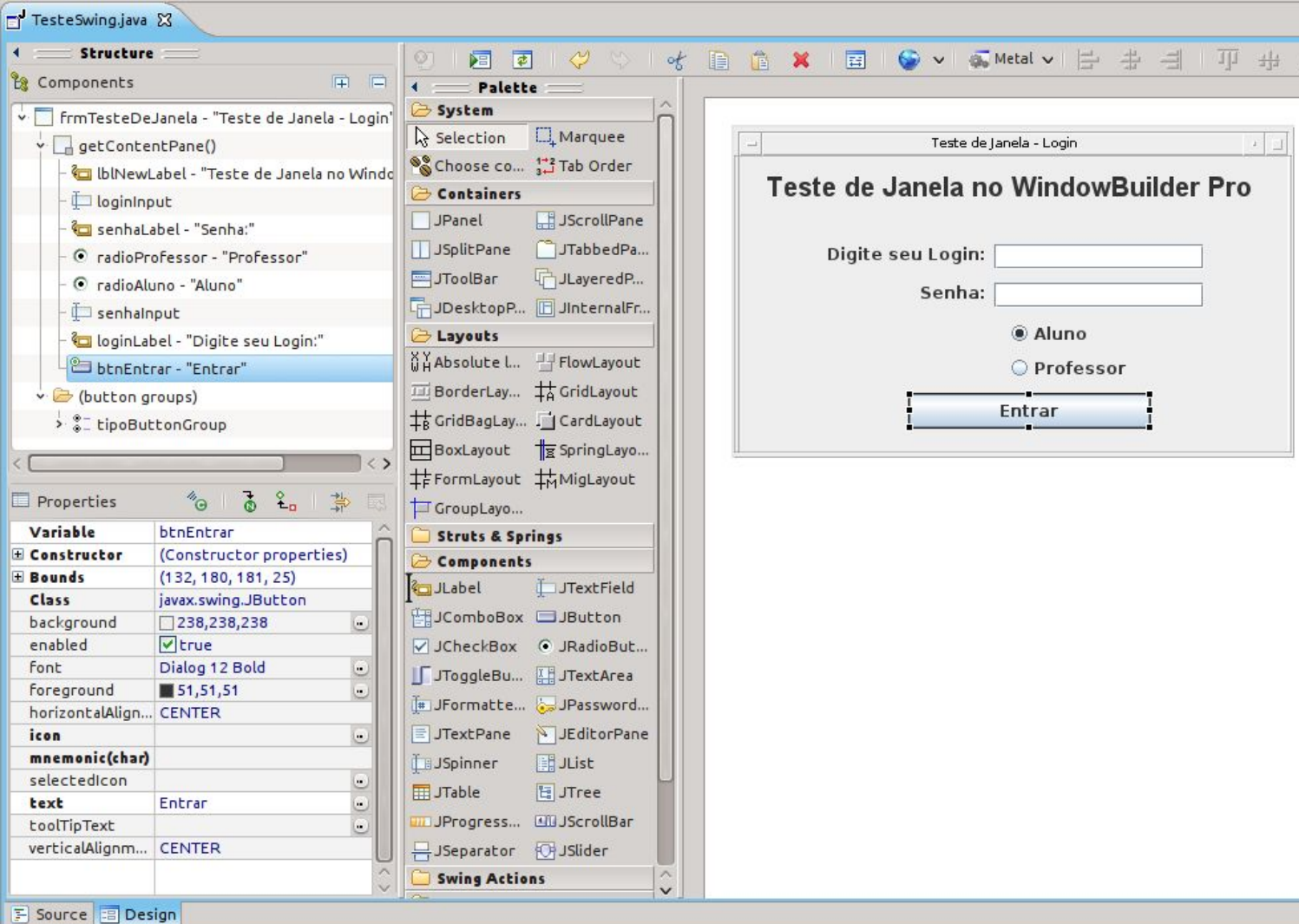


- O Eclipse facilita o trabalho com GUI através de complementos. Um dos mais conhecidos é o WindowBuilder Pro, da Google.
  - Antes de Instalar, certifique-se que você está usando a última versão do Eclipse
    - O *WindowBuilder* é conhecido por não funcionar em versões desatualizadas
  - Vá no menu *Help* → *Eclipse MarketPlace*
  - Pesquise por *WindowBuilder*
  - Selecione *WindowBuilder Current*
  - Clique em *Install Now*

# Usando o Eclipse



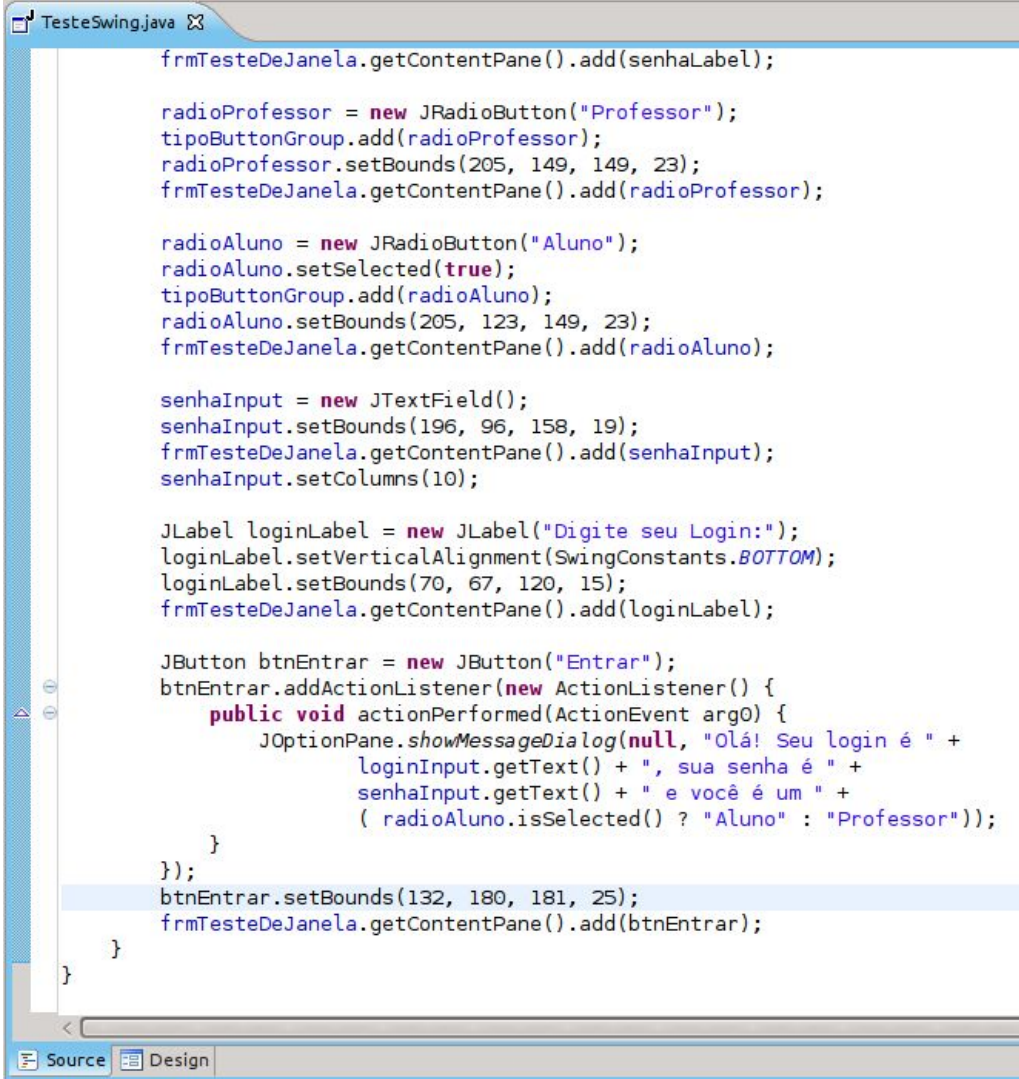
- Para criar uma nova Janela:
  - File → New → Other... → WindowBuilder → Swing Designer → App Wind.
- Na parte de baixo da janela, encontre as abas “Source” e “Design”
  - Clique em “Design” para ver a interface do WindowBuilder
- Para alternar entre Source e Design:
  - Navigate → Switch between Source/Design Views (F12)



# Usando o Eclipse

# Usando o Eclipse

- Eventos:
  - Ao clicar duas vezes em um componente no "Designer", ele abre o código para manipulação de eventos daquele componente.



```
TesteSwing.java

frmTesteDeJanela.getContentPane().add(senhaLabel);

radioProfessor = new JRadioButton("Professor");
tipoButtonGroup.add(radioProfessor);
radioProfessor.setBounds(205, 149, 149, 23);
frmTesteDeJanela.getContentPane().add(radioProfessor);

radioAluno = new JRadioButton("Aluno");
radioAluno.setSelected(true);
tipoButtonGroup.add(radioAluno);
radioAluno.setBounds(205, 123, 149, 23);
frmTesteDeJanela.getContentPane().add(radioAluno);

senhaInput = new JTextField();
senhaInput.setBounds(196, 96, 158, 19);
frmTesteDeJanela.getContentPane().add(senhaInput);
senhaInput.setColumns(10);

JLabel loginLabel = new JLabel("Digite seu Login:");
loginLabel.setVerticalAlignment(SwingConstants.BOTTOM);
loginLabel.setBounds(70, 67, 120, 15);
frmTesteDeJanela.getContentPane().add(loginLabel);

JButton btnEntrar = new JButton("Entrar");
btnEntrar.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
        JOptionPane.showMessageDialog(null, "Olá! Seu login é " +
            loginInput.getText() + ", sua senha é " +
            senhaInput.getText() + " e você é um " +
            (radioAluno.isSelected() ? "Aluno" : "Professor"));
    }
});
btnEntrar.setBounds(132, 180, 181, 25);
frmTesteDeJanela.getContentPane().add(btnEntrar);
}
```

# Usando o Eclipse



- Conclusões:
  - Use o WindowBuilder Pro no Eclipse para montar sua interface
  - Utilize seus conhecimentos de Swing para fazer alterações menores que o editor não consegue fazer facilmente