



# SIN143 Laboratório de Programação

---

Prof. João Batista Ribeiro

[joao42ibatista@gmail.com](mailto:joao42ibatista@gmail.com)



---

Universidade Federal de Viçosa

---



# Aula de Hoje

---

- Introdução a Programação Gráfica com Swing



# AWT x Swing

---

- Java 1.0 possui a biblioteca de classe chamada AWT
- AWT (*Abstract Window Toolkit*): kit de ferramentas para janelas abstratas
- Biblioteca AWT delega a criação de interface gráfica ao kit de ferramentas GUI nativo em cada plataforma (Windows, Solaris, Mac, etc)



# AWT x Swing

---

- Problemas com AWT:
  - Menus, barras de rolagem, campos de texto tinham diferenças sutis de comportamento em diferentes plataformas.
  - Alguns ambientes gráficos não tinham uma coleção abundante de componentes de interface de usuário como o Windows e Mac.
  - Ocorria erros diferentes na biblioteca da interface de usuário AWT em cada plataforma.
  - Testar aplicativos em cada plataforma: “escrever uma vez, depurar em todo lugar”

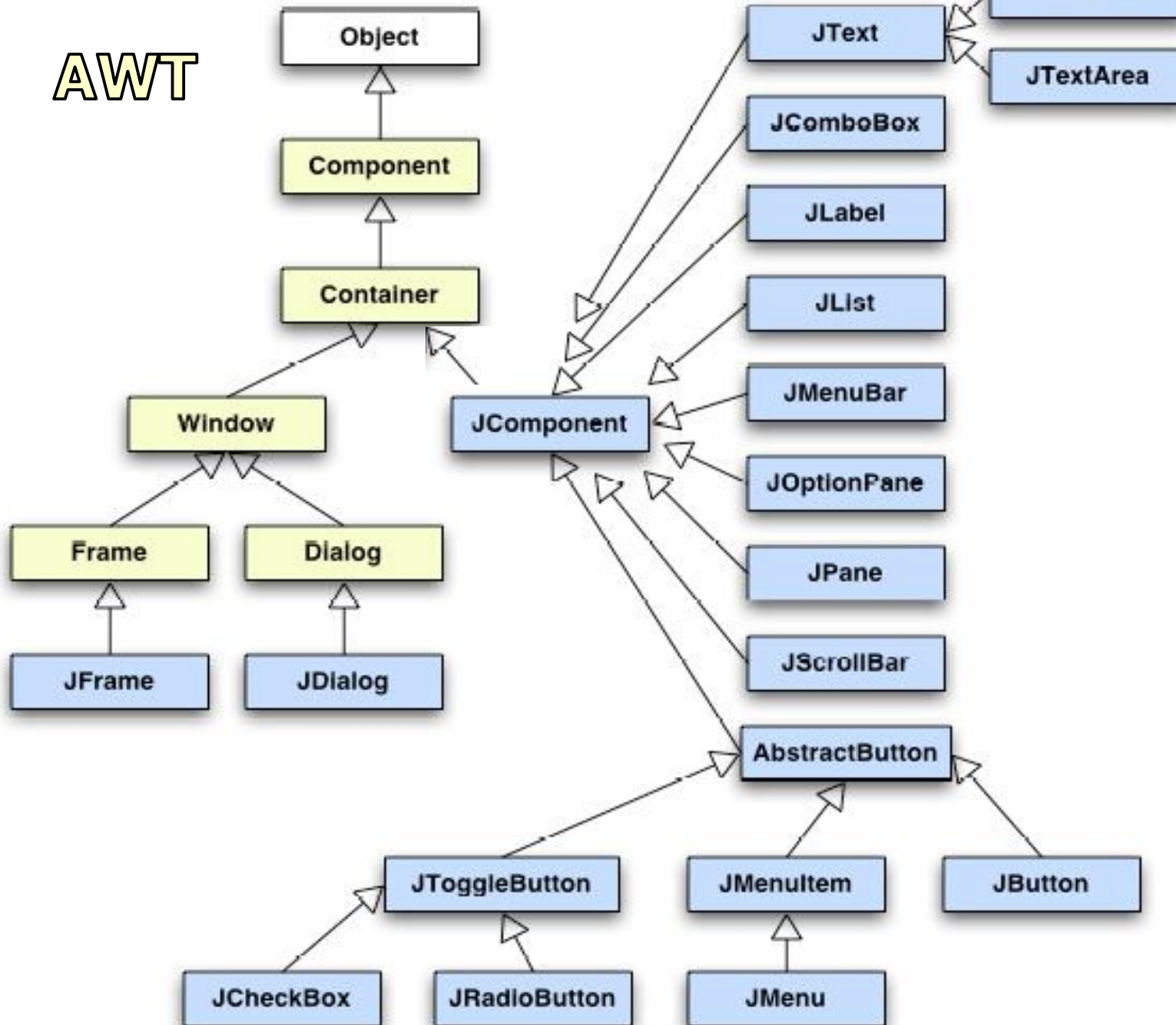


# AWT x Swing

---

- Swing não é substituto completo do AWT
- Swing é mais robusto, tem mais recursos, é mais portátil e é mais fácil de usar que os componentes GUIs AWT.
- Slogan da Sun: “escrever uma vez, executar em qualquer lugar.”. WORA: "Write once, run anywhere"
- GUIs baseadas em Swing são mais lentas que GUIs AWT

AWT



Swing

# Criando um Quadro - Swing

- AWT (classe “Frame”) x Swing (classe “JFrame”)
- Os frames (quadros) são contêineres que podem conter outros elementos de interface de usuário: botões, campos de texto, etc.





# Exemplo

---

- Classes:
  - OlaMundoFrame.java
  - OlaMundo.java



<pre>package fonte; import javax.swing.*; public class OlaMundoFrame extends JFrame{     public OlaMundoFrame(){         this.setTitle("Meu primeiro Frame");         this.setLocation(350,250);         this.setSize(300,200);          this.setDefaultCloseOperation(JFrame.EXIT_ON CLOSE);     } }</pre>	OlaMundoFrame.java
---	--------------------

<pre>package fonte; import javax.swing.*; public class OlaMundoTest{      public static void main(String[] args){         OlaMundoFrame frame = new OlaMundoFrame();         frame.setVisible(true);     } }</pre>	OlaMundoTest.java
--	-------------------

- As classes Swing são colocadas no pacote *javax.swing*
- *javax* indica um pacote de extensão java (swing) e não um pacote do núcleo (awt)

```
import javax.swing.*;  
public class OlaMundoFrame extends JFrame
```

← → ⓘ <https://docs.oracle.com/javase/8/docs/api/>

→ javax.swing  
javax.swing.border  
javax.swing.colorchooser  
javax.swing.event  
javax.swing.filechooser  
javax.swing.plaf  
javax.swing.plaf.basic  
javax.swing.plaf.metal  
javax.swing.plaf.multi

→ JFrame  
JInternalFrame  
JInternalFrame.JDesktopIcon

## Constructor Summary

### Constructors

#### Constructor and Description

**JFrame()**

Constructs a new frame that is initially invisible.

**JFrame(GraphicsConfiguration gc)**

Creates a Frame in the specified GraphicsConfiguration of a screen device and a

**JFrame(String title)**

Creates a new, initially invisible Frame with the specified title.

**JFrame(String title, GraphicsConfiguration gc)**

Creates a JFrame with the specified title and the specified GraphicsConfiguration

## Method Summary

```
import javax.swing.*;
public class OlaMundoFrame extends JFrame{
    public OlaMundoFrame(){
        setTitle("Meu primeiro Frame");
        setLocation(350,250);
        setSize(300,200);
    }
}
```

- A classe “OlaMundoFrame” se comporta como a classe JFrame com três exceções:
  - O construtor de OlaMundoFrame coloca na barra de título a string “Meu primeiro Frame”
  - Posiciona o quadro no centro da tela
  - Faz o tamanho do quadro ser de 300 x 200 pixels

```
import javax.swing.*;

public class olaMundo {
    public static void main(String[] args){
        olaMundoFrame frame = new OlaMundoFrame();
        frame.setVisible(true);
    }
}
```

- Na classe “olaMundo” o método main exibe o quadro na tela.
- A linha em **vermelho** cria um objeto da classe “OlaMundoFrame” fornecendo todas as informações para que a janela seja criada.
- Para exibir o quadro na tela o método “setVisible” é chamado.



# Etapas para exibir um quadro na tela

---

- Criar um objeto quadro usando o operador *new*.
- Posicioná-lo na tela usando o método *setLocation*.
- Chamar o método *setVisible* para tornar o quadro visível e trazê-lo para frente se estiver por trás de outra janela.



# Layout de Quadros

---

- A classe JFrame possui poucos métodos para alterar a disposição visual (*layout*) dos quadros.
- A maior parte dos métodos para se trabalhar o dimensionamento e o posicionamento de um quadro vêm da várias superclasses de JFrame.

java.lang.Object

\*\* └ java.awt.Component

└ java.awt.Container

\*\* └ java.awt.Window

└ java.awt.Frame

└ **javax.swing.JFrame**

\*\* São encontrados métodos para redimensionar e reformatar os quadros

# API - Métodos para reformatar/redimensionar



---

- **java.awt.Component**

- boolean isVisible()
- void setVisible()
- boolean isShowing()
- boolean isEnabled()
- void setLocation(int x, int y)
- void setBounds(int x, int y, int largura, int altura)
- etc.

- **java.awt.Frame**

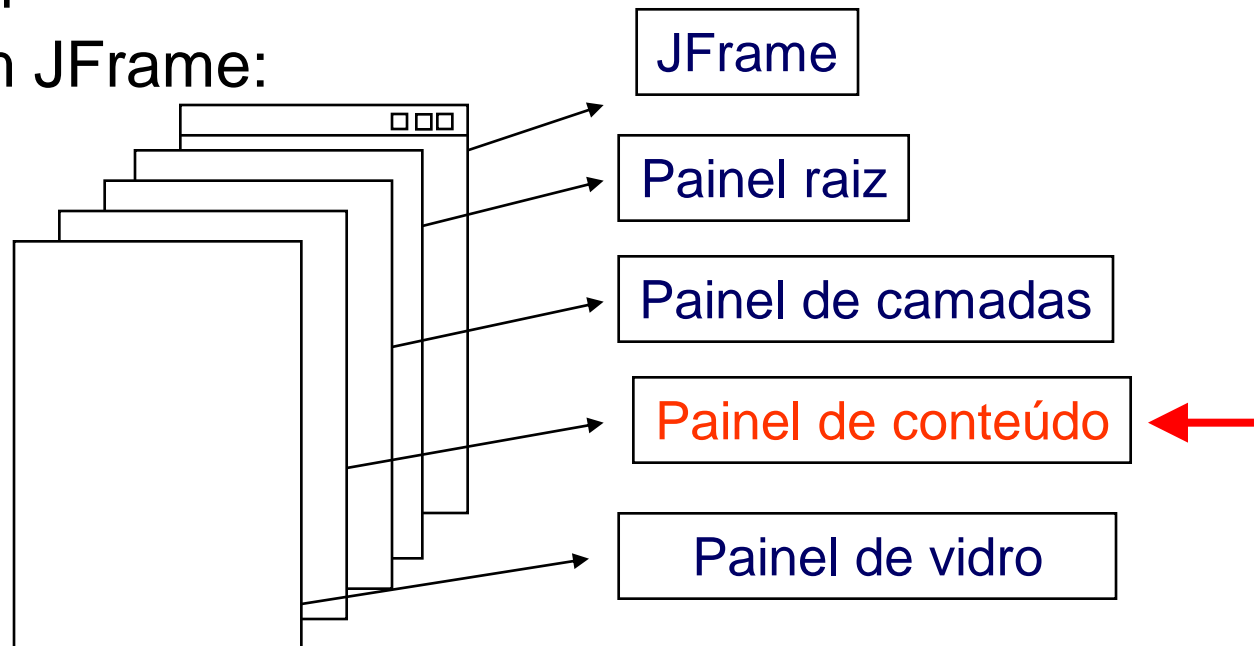
- void setResizable(boolean b)
- void setTitle(String s)
- void setIconImage(Imagem image)

- **java.awt.Window**

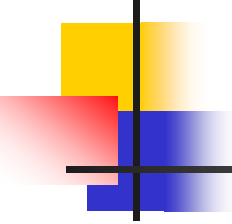
- void toFront()
- void toBack()

# Exibição de Informações em um Quadro

- Em Java os quadros (frames) são projetos para serem contêineres de componentes.
- O recomendado é escrever dentro de um *panel*, que é adicionado ao quadro.
- Estrutura de um JFrame:







# JFrame + JPanel

---



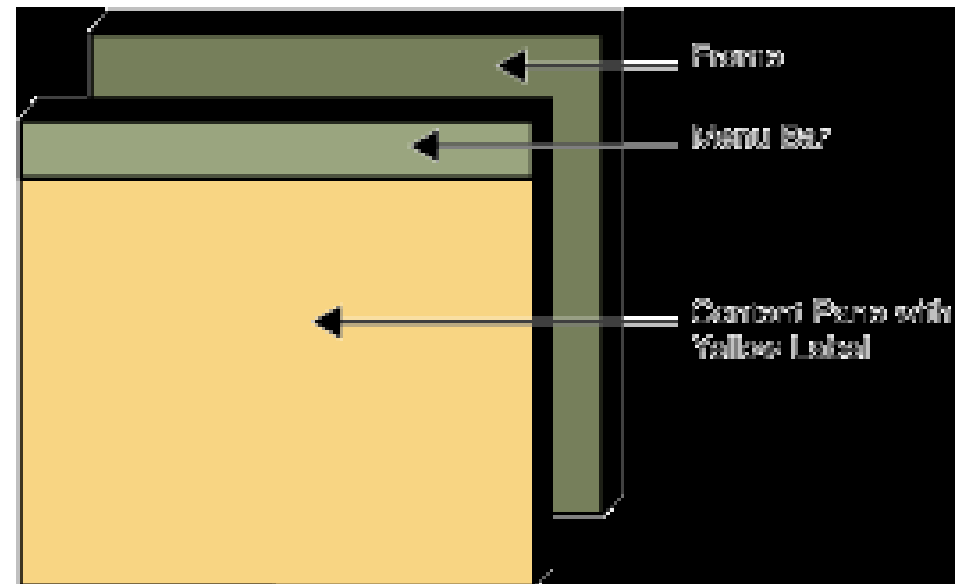
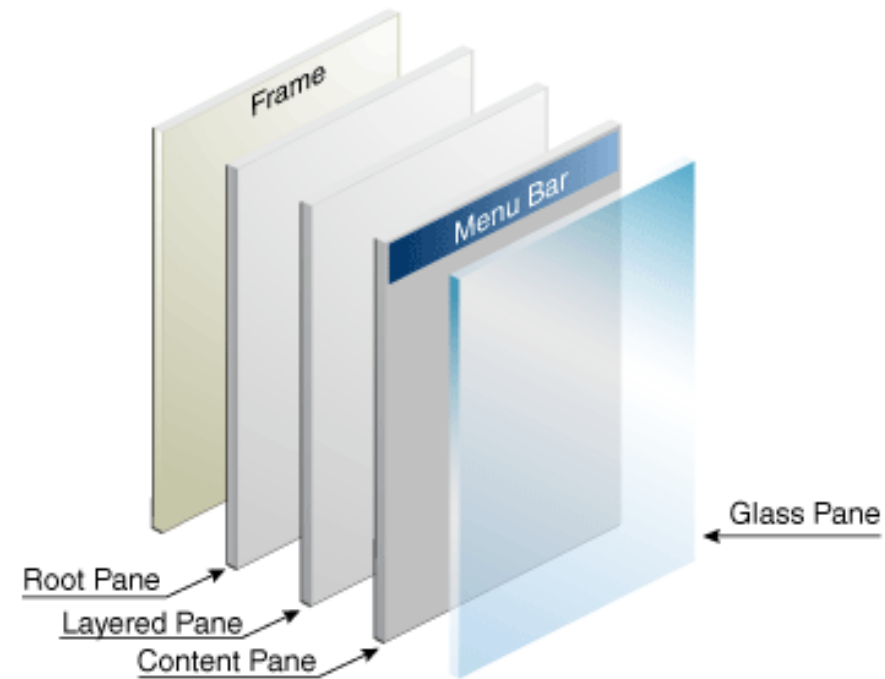
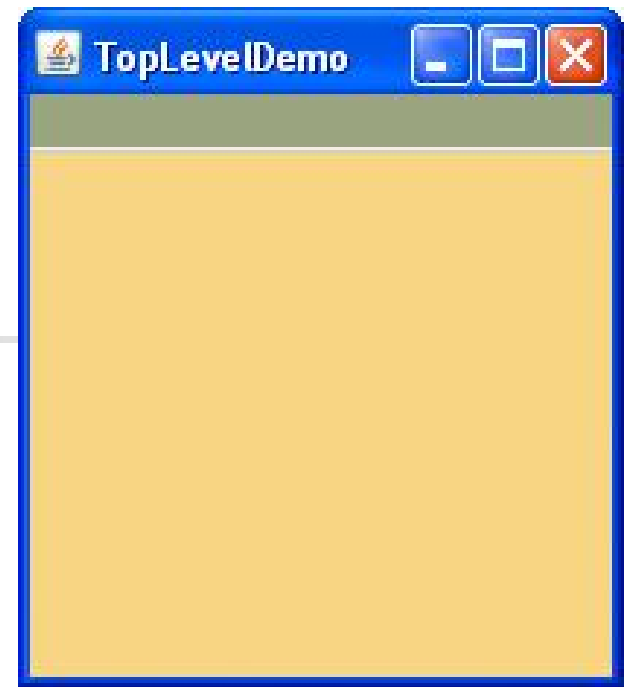
+



=



# Camadas





# Exibição de Informações em um Quadro

---

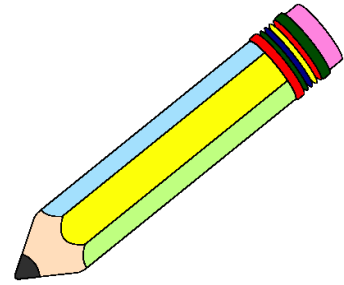
- A parte que mais interessa aos programadores Swing é a “área de conteúdo”
- A classe JPanel é usada para implementar painéis.
- Código para adicionar um painel a um quadro:

```
Container painelConteudo = frame.getContentPane();  
JPanel p = new JPanel();  
painelConteudo.add(p);
```

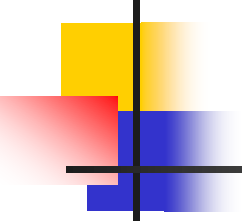


# Método `paintComponent`

---



- O método *paintComponent* é usado para redesenhar componentes na tela.
- O método *paintComponent* possui como parâmetro um elemento do tipo *Graphics*.
- O **objeto *Graphics*** possui funcionalidades para desenhar imagens, texto.
- Toda a ação de desenhar em Java precisa usar um objeto *Graphics*.

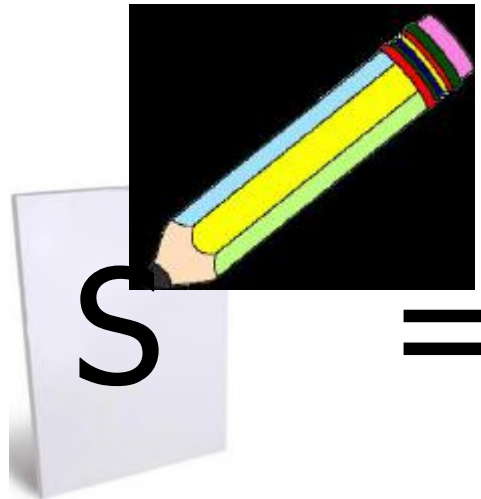


# JFrame + JPanel

---



+



=





# Método paintComponent

---

- A impressão de um texto é considerado um tipo de desenho.
- Um objeto Graphics tem um método *drawString* que usa a seguinte sintaxe:

```
g.drawString(texto, Coordx, Coordy);
```



# Três Classes

---

- Classes:
  - 1) OlaMundoFrame.java
  - 2) OlaMundoPanel.java
  - 3) OlaMundo.java



# OlaMundoFrame.java

```
package fonte;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class OlaMundoFrame extends JFrame {
    public OlaMundoFrame() {
        setTitle("Meu primeiro Frame");
        setSize(300,200);
        setLocation(350,250);

        Container container = getContentPane();
        container.add(new OlaMundoPanel());
    }
}
```



# OlaMundoPanel.java

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

public class OlaMundoPanel extends JPanel{
    public void paintComponent(Graphics g){
        super.paintComponents(g);
        g.drawString("Olá Mundo", 100, 70);
    }
}
```



# OlaMundo.java

```
package fonte;
import javax.swing.*;

public class OlaMundo{
    public static void main(String[] args){
        OlaMundoFrame frame = new OlaMundoFrame();
        frame.setDefaultCloseOperation(
                                JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```



# Texto e Fontes

---

- Um objeto fonte é composto das seguintes partes:
  - Nome da fonte: “Times New Roman”, “Arial”, “Courier”, etc.
  - Estilo: “Bold”, “Italic”, “Plain”
  - Tamanho da fonte: 8, 9, 10, ..., 14, ..., 20
- Para descobrir as fontes que estão disponíveis em um computador use o seguinte comando:

```
String[]fontes =  
    GraphicsEnvironment.getLocalGraphicsEnvironment().  
    getAvailableFontFamilyNames();
```



# Texto e Fontes

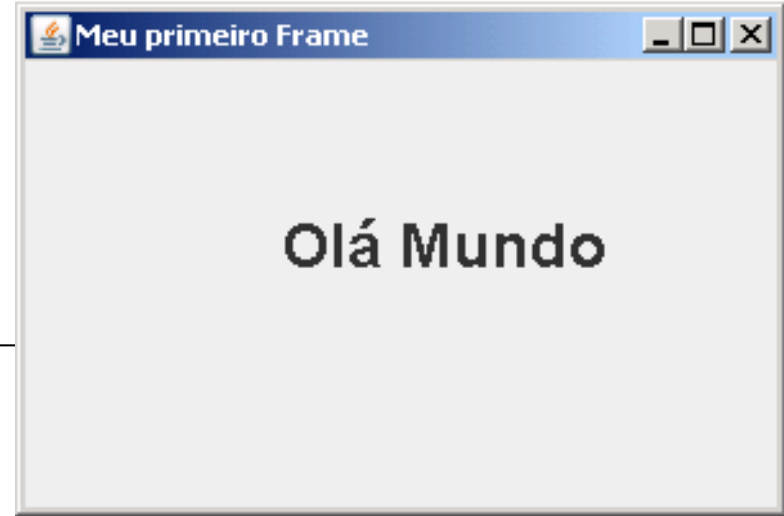
---

- Como criar um objeto do tipo Font:

```
Font arialB14 = new Font("Arial", Font.BOLD, 14);
```

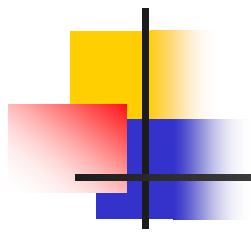
# OlaMundoPanel.java

## Setando a fonte



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

public class OlaMundoPanel extends JPanel{
    public void paintComponent(Graphics g){
        super.paintComponents(g);
        Font arialB24=new Font("Arial", Font.BOLD,24);
        g.setFont(arialB24);
        g.drawString("Olá Mundo", 100, 70);
    }
}
```



# Cores

---

- Um objeto cor é composto pelas seguintes cores na escala de 0 a 255:
  - Cor vermelha
  - Cor verde
  - Cor azul
  
- Como criar um objeto do tipo Color:

```
Color x = new Color(int nivelVermelho, int nivelVerde, int nivelAzul);
```



# Cores

---

- Constantes de cores padrões:
  - black (preto)
  - blue (azul)
  - green (verde)
  - orange (laranja)
  - red (vermelho)
  - white (branco)
  - pink (rosa)
  - yellow (amarelo)
- Para alterar a cor de um texto use o método `setColor(Color x)`
- Para alterar a cor de um fundo de um componente use o método: `setBackground(Color x)`

# OlaMundoPanel.java

## Alterando a cor da fonte

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

public class OlaMundoPanel extends JPanel{
    public void paintComponent(Graphics g){
        super.paintComponents(g);
        Font arialB14=new Font("Arial", Font.BOLD,24);
        g.setFont(arialB14);
        g.setColor(Color.red);
        g.drawString("Olá Mundo", 100, 70);
    }
}
```





# OlaMundo.java

## Alterando a cor de fundo



```
import javax.swing.*;
import java.awt.*;
public class OlaMundo {
    public static void main(String[] args) {
        OlaMundoFrame frame = new OlaMundoFrame();
        frame.setBackground(Color.black);
        frame.setVisible(true);
    }
}
```



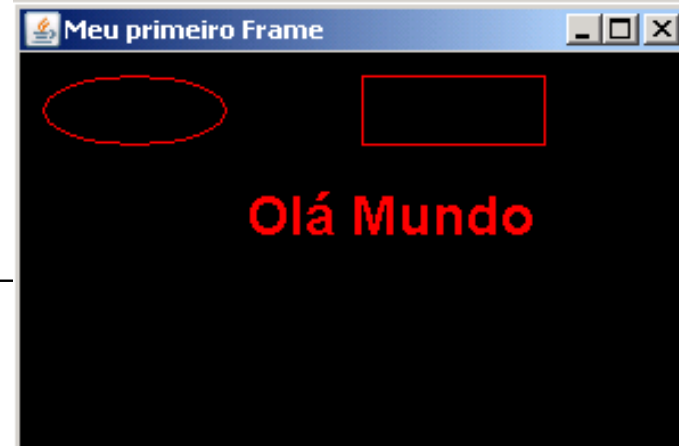
# Como desenhar ...

---

- Linha: `void drawLine(int x1, int y1, int x2, int y2)`
- Retângulo: `void drawRect(int x, int y, int largura, int altura)`
- Elipse: `void drawOval(int x, int y, int largura, int altura)`

# OlaMundoPanel.java

## Desenhando figuras



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

public class OlaMundoPanel extends JPanel{
    public void paintComponent(Graphics g){
        super.paintComponents(g);
        Font arialB14 = new Font("Arial", Font.BOLD,14);
        g.setFont(arialB14);
        g.setColor(Color.red);
        g.drawOval(10,10, 80, 30);
        g.drawRect(150,10, 80, 30);
        g.drawString("Olá Mundo", 100, 80);
    }
}
```

# Botão

The image is a screenshot of a web browser displaying the Oracle Java API documentation for the `JButton` class. The browser's address bar shows the URL `https://docs.oracle.com/javase/8/docs/api/`. On the left side, there is a navigation pane with a list of Java packages and classes. The package `javax.swing` is highlighted with a red box and a red arrow pointing to it. Below it, the class `JButton` is also highlighted with a red box and a red arrow. The main content area on the right is titled "Constructor Summary" and lists several constructors for the `JButton` class. The constructor `JButton(String text)` is circled in red. The constructors listed are:

- `JButton()`: Creates a button with no set text or icon.
- `JButton(Action a)`: Creates a button where properties are taken from the `Action` supplied.
- `JButton(Icon icon)`: Creates a button with an icon.
- `JButton(String text)`: Creates a button with text.
- `JButton(String text, Icon icon)`: Creates a button with initial text and an icon.



# Três Classes

---

- Classes:
  - BotaoFrame.java
  - BotaoPanel.java
  - BotaoPrincipal.java

# BotaoFrame.java

BotaoFrame.java

```
package botao;
import java.awt.*;
import javax.swing.*;
public class BotaoFrame extends JFrame {
    public BotaoFrame() {
        setTitle("Tela de botões!!!");
        setSize(300,200);
        setLocation(350,250);

        Container container = getContentPane();
        container.add(new BotaoPanel());
    }
}
```

# BotaoPanel.java

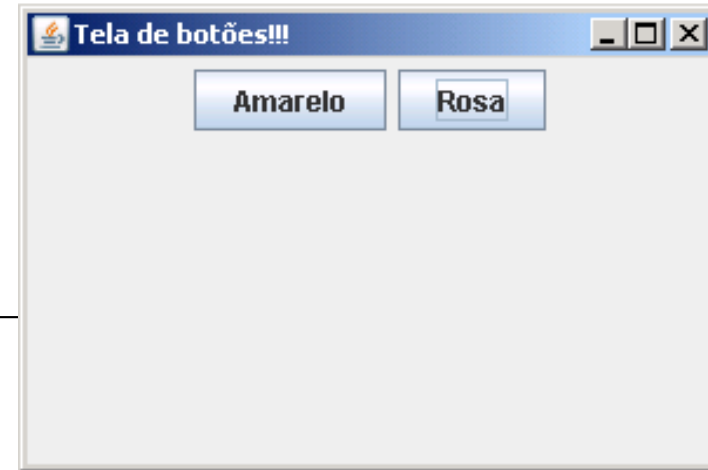
```
package botao;

import javax.swing.*.*;

public class BotaoPanel extends JPanel{
    private JButton botao1;
    private JButton botao2;

    public BotaoPanel(){
        botao1 = new JButton("Amarelo");
        botao2 = new JButton("Rosa");

        add(botao1);
        add(botao2);
    }
}
```



BotaoPanel.java

# BotaoPrincipal.java

OlaMundoBotao.java

```
package botao;

import javax.swing.*.*;

public class OlaMundoBotao{
    public static void main(String[] args){
        BotaoFrame frame = new BotaoFrame();
        frame.setDefaultCloseOperation(
            JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```





# Tratamento de eventos

---

- As GUIs são baseadas em eventos (isto é, geram eventos quando o usuário de programa interage com a GUI)
- Tipos de interações que são classificadas como eventos:
  - Mover o mouse,
  - Clicar em um botão,
  - Digitar em um campo de texto,
  - Selecionar um item de menu,
  - Fechar uma janela,
  - Etc.



# Tratamento de eventos

---

- O mecanismo de tratamento de eventos tem 3 partes:
  - Origem do evento: objeto que gerou o evento e notificou o ouvinte(o componente GUI com o qual o usuário interage).
  - Objeto evento: encapsula as informações sobre o evento que ocorreu.
  - “Ouvinte” (*listener*) do evento: é um objeto que é notificado pela origem do evento quando ocorre um evento.



# “Ouvintes” (*listener*) dos eventos

---

- Os ouvintes para um evento GUI é um objeto que implementa uma ou mais interfaces de eventos dos pacotes:
  - `java.awt.event`: tipos de ouvintes de eventos comuns aos componentes Swing e AWT
  - `java.swing.event`: tipos adicionais de ouvintes específicos para componentes Swing

Exemplos de ouvintes do pacote `java.awt.event`:

- |                               |                                   |                              |
|-------------------------------|-----------------------------------|------------------------------|
| - <code>ActionListener</code> | - <code>AdjustmentListener</code> | - <code>ItemListener</code>  |
| - <code>TestListener</code>   | - <code>KeyListener</code>        | - <code>FocusListener</code> |

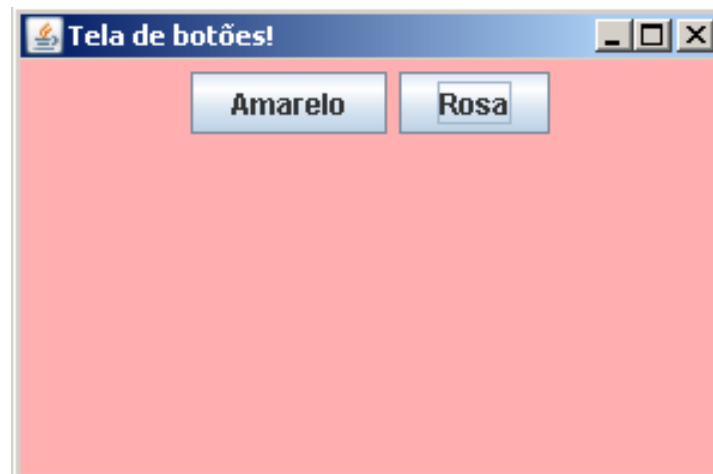


# Eventos do Botão

---

- Interface: *ActionListener*
- Métodos: *actionPerformed*
- Parâmetros: *ActionEvent*
- Eventos gerados por: *Button*, *List*, *MenuItem*, *TextField*

# Saída do Programa



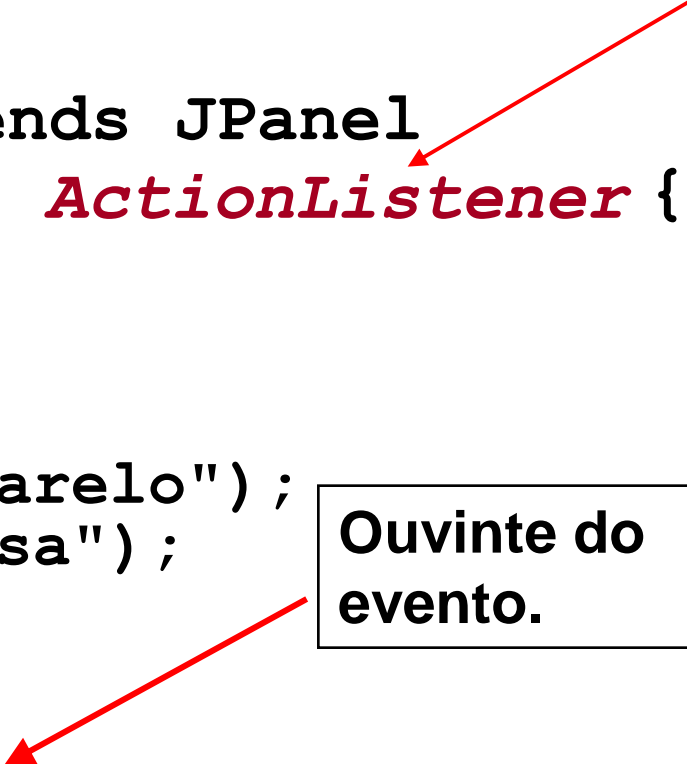
# BotaoPanel.java

Interface com um único método.

```
import java.awt.*;
import javax.swing.*;
public class BotaoPanel extends JPanel
    implements ActionListener {
    private JButton botao1;
    private JButton botao2;
    public BotaoPanel() {
        botao1 = new JButton("Amarelo");
        botao2 = new JButton("Rosa");

        add(botao1);
        add(botao2);

        botao1.addActionListener(this);
        botao2.addActionListener(this);
    }
    ...
}
```




Ouvinte do evento.

Esse objeto fornece as informações sobre o evento que ocorreu.



```
public void actionPerformed(ActionEvent evt) {  
    Object source = evt.getSource();  
    Color color = getBackground();  
  
    if(source == botao1)  
        color = Color.yellow;  
    else if(source == botao2)  
        color = Color.pink;  
  
    setBackground(color);  
    repaint();  
}  
}
```



getSource informa a origem do evento.



# Eventos Semânticos x Baixo Nível

---

- **Eventos Semânticos**

- *ActionEvent*: clique de botão, seleção de um menu, clique duplo em um item de uma lista, tecla ENTER pressionada em um campo texto.
- *AdjustmentEvent*: o usuário ajusta uma barra de rolagem.
- *TextEvent*: o conteúdo de um campo de texto ou área de texto foi modificado.





# Eventos Semânticos x Baixo Nível

---

- **Baixo Nível**

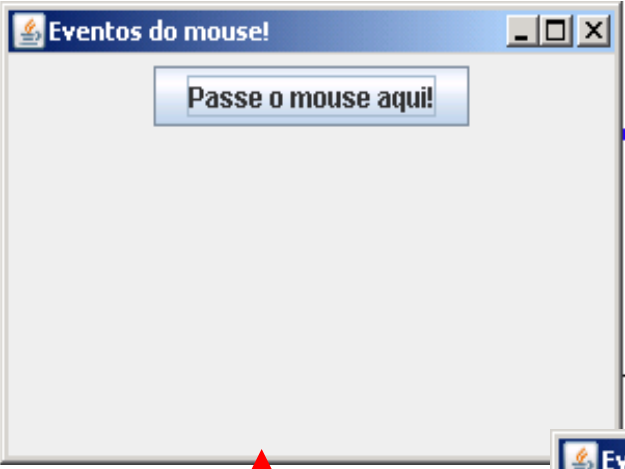
- *ComponentEvent*: o componente foi redimensionado, movido, exibido ou ocultado.
- *KeyEvent*: uma tecla foi liberada ou pressionada.
- *MouseEvent*: o botão do mouse foi pressionado, liberado, movido ou arrastado.
- *WindowEvent*: uma janela foi ativada, desativada, minimizada, restaurada ou fechada.



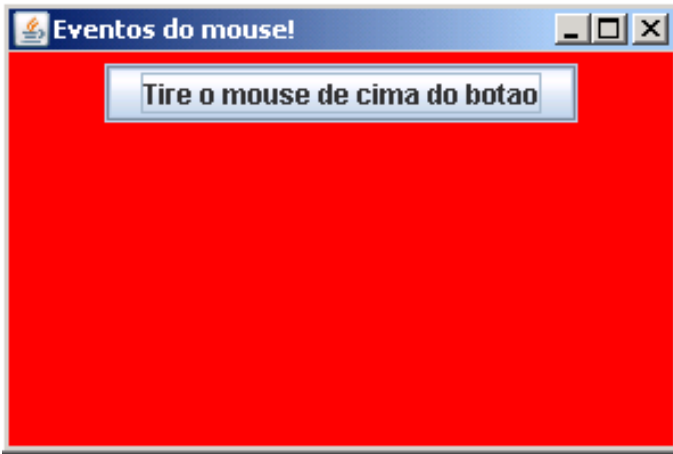
# Eventos de Mouse

---

- Interface: *MouseListener*
- Métodos: *mousePressed*  
*mouseRelease*  
*mouseEntered*  
*mouseExited*  
*mouseClicked*
- Parâmetros: *MouseEvent*
- Eventos gerados por: *Component*

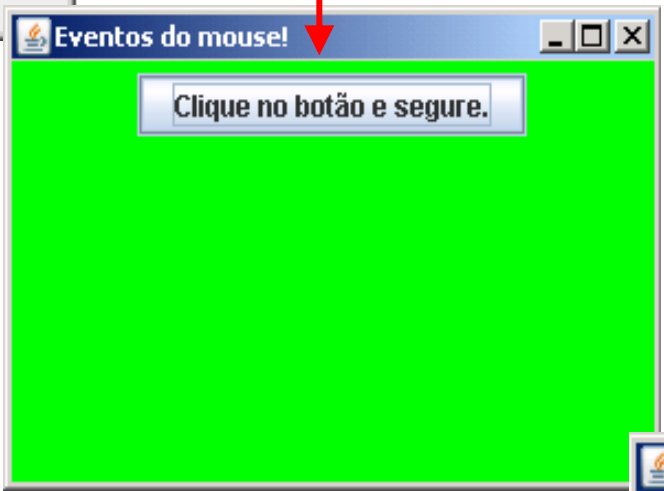


mouseEntered



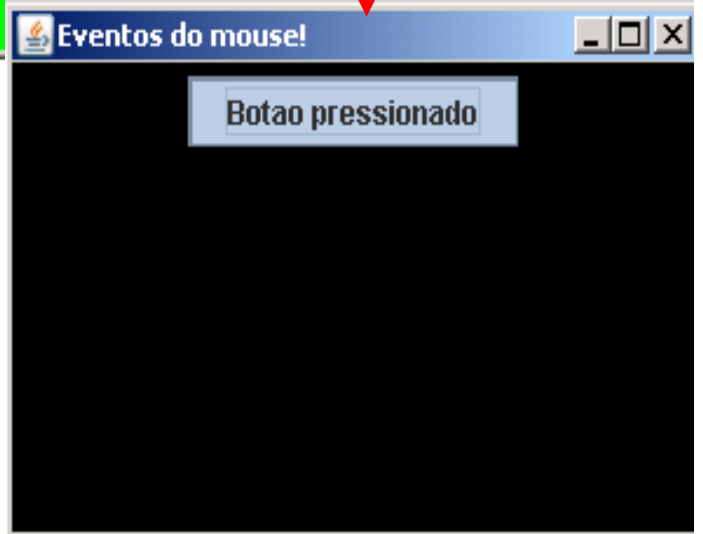
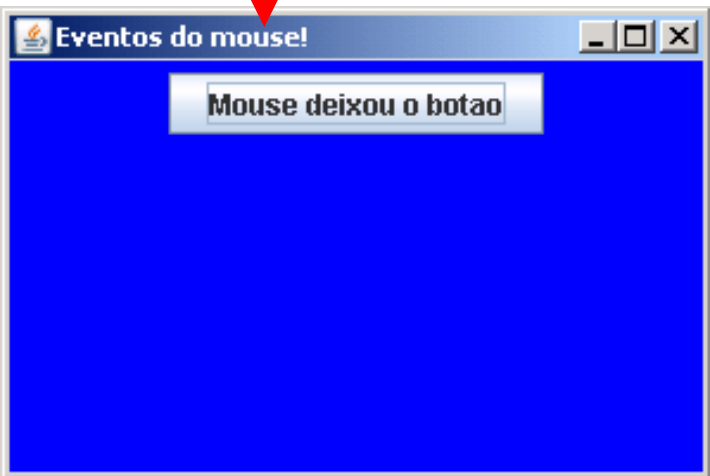
Mensagem do construtor

mouseExited



mouseClicked

mousePressed





# Três Classes

---

- Pacote “mouse”
- Classes:
  - 1) MouseFrame.java
  - 2) MousePanel.java
  - 3) MousePrincipal.java

# MouseFrame.java

```
package mouse;
import java.awt.*;
import javax.swing.*;
public class MouseFrame extends JFrame {
    public MouseFrame() {
        setTitle("Eventos do mouse");
        setSize(300,200);
        setLocation(350,250);

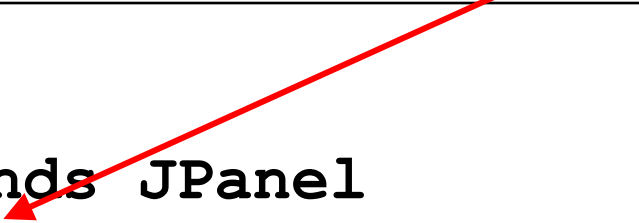
        Container container = getContentPane();
        container.add(new MousePanel());
    }
}
```

MouseFrame.java

# MousePanel.java

Interface com cinco  
métodos

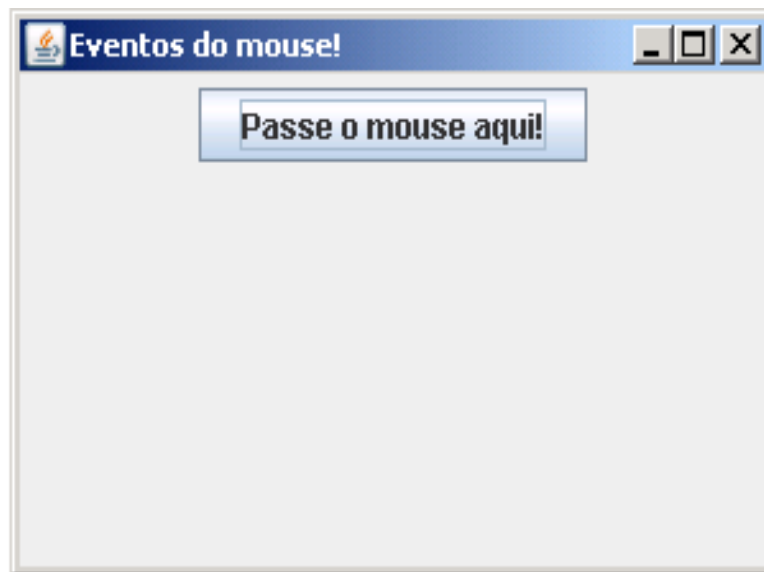
```
import javax.swing.*;  
  
public class MousePanel extends JPanel  
    implements MouseListener  
{  
    private JButton botao1;  
  
    public MousePanel() {}  
  
    public void mouseReleased(MouseEvent evt) {}  
    public void mouseEntered(MouseEvent evt) {}  
    public void mousePressed(MouseEvent evt) {}  
    public void mouseExited(MouseEvent evt) {}  
    public void mouseClicked(MouseEvent evt) {}  
}
```



MousePanel.java

# Código do construtor

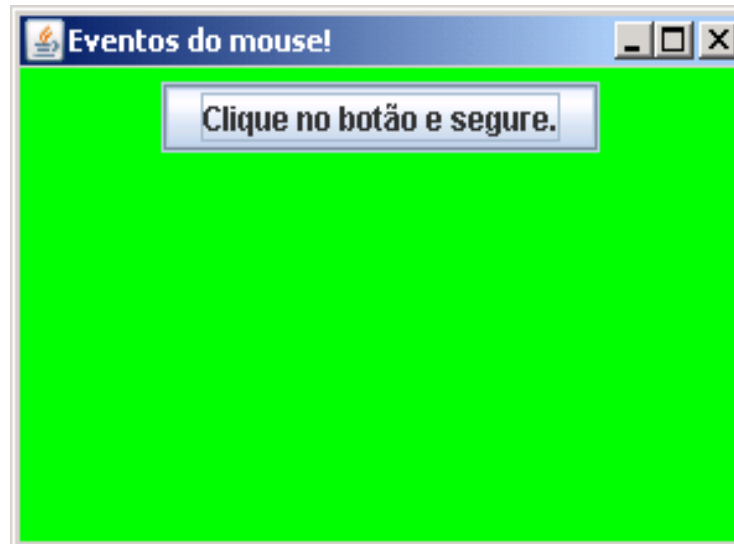
```
public MousePanel()  
{  
    botao1 = new JButton("Passe o mouse aqui!");  
    add(botao1);  
    botao1.addMouseListener(this);  
}
```



Ouvinte do evento.

# Código dos métodos

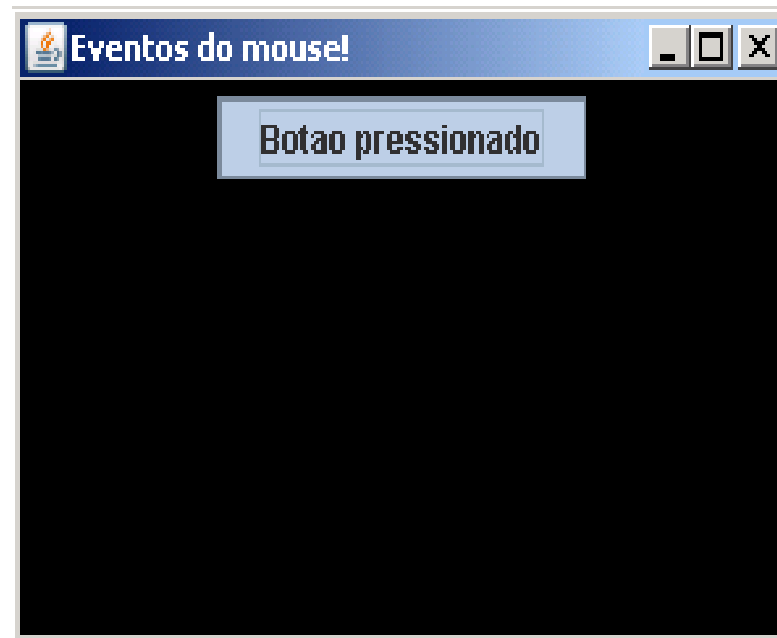
```
public void mouseReleased(MouseEvent evt) {}  
  
public void mouseEntered(MouseEvent evt)  
{  
    botao1.setText("Clique no botão e segure.");  
    setBackground(Color.green);  
    repaint();  
}
```





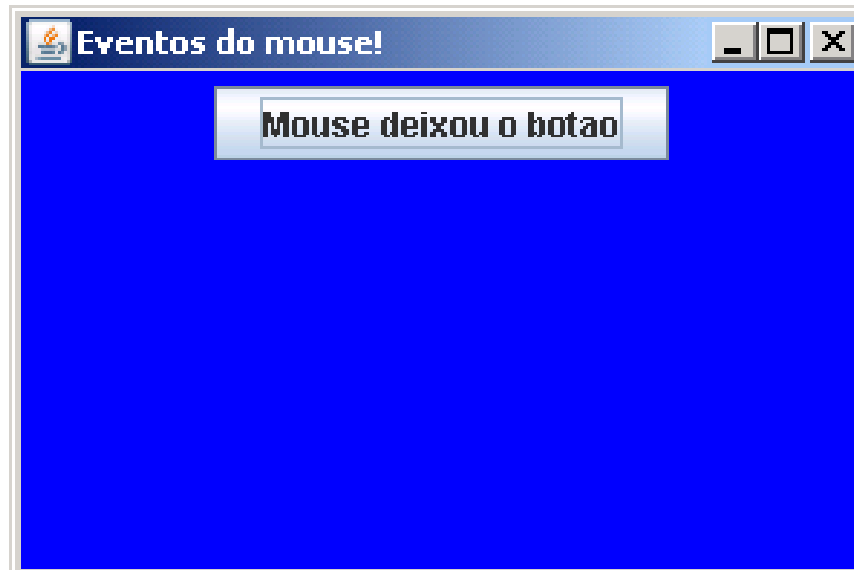
# Código dos métodos

```
public void mousePressed(MouseEvent evt) {  
    botao1.setText("Botao pressionado");  
    setBackground(Color.black);  
    repaint();  
}
```



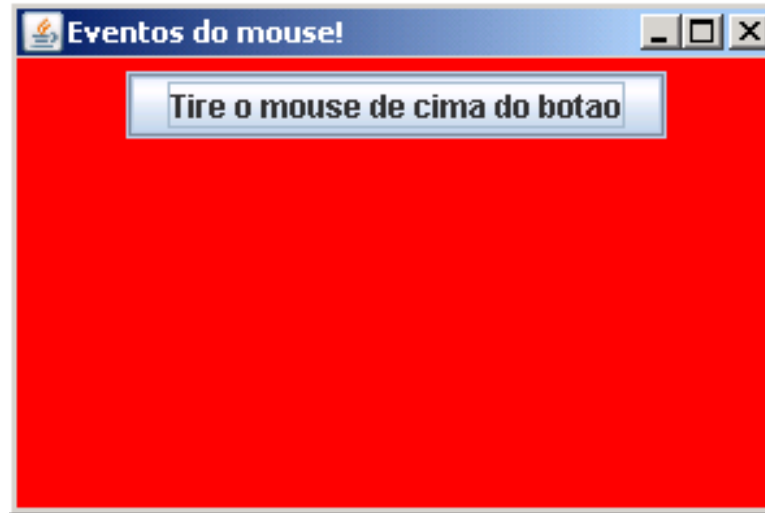
# Código dos métodos

```
public void mouseExited(MouseEvent evt){  
    botao1.setText("Mouse deixou o botao");  
    setBackground(Color.blue);  
    repaint();  
}
```



# Código dos métodos

```
public void mouseClicked(MouseEvent evt){  
    botao1.setText("Tire o mouse de cima do botao");  
    setBackground(Color.red);  
    repaint();  
}
```



# MousePrincipal.java

MousePanel.java

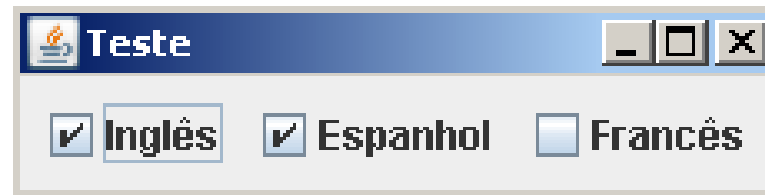
```
package mouse;

import javax.swing.*.*;

public class MousePrincipal{
    public static void main(String[] args){
        JFrame frame = new JFrame();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

# Botões de escolha

- Caixa de seleção: *JCheckBox*



- *JRadioButton*



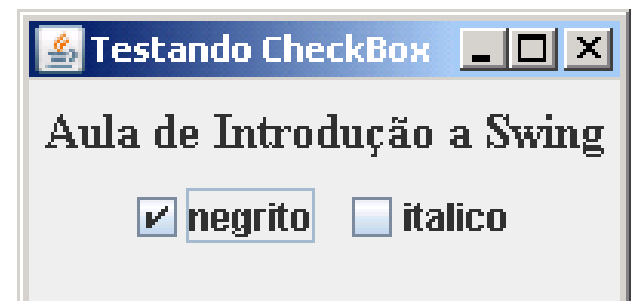
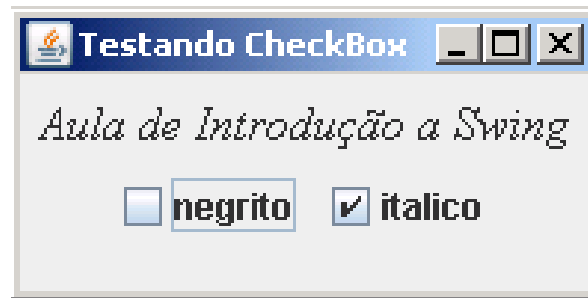
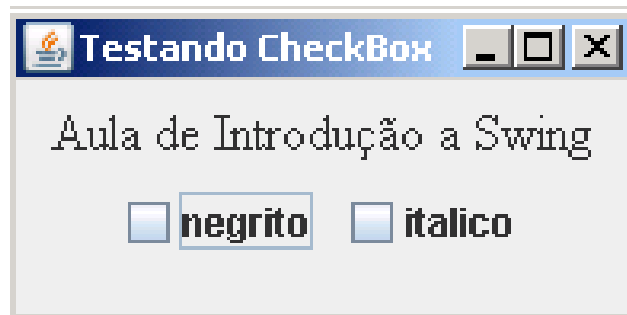


# Três Classes - JCheckBox

---

- Interface: ItemListener
- Método: itemStateChanged
- Classes:
  - 1) CheckBoxFrame.java
  - 2) CheckBoxPanel.java
  - 3) CheckBoxPrincipal.java

# Saída do Programa





# CheckBoxFrame.java

---

```
package checkbox;  
import java.awt.*;  
import javax.swing.*;  
public class CheckBoxFrame extends  
    JFrame {  
  
    ...  
}
```

CheckBoxFrame.java





# CheckBoxFrame.java

---

```
public CheckBoxFrame() {  
    setTitle("Eventos do mouse");  
    setSize(200,100);  
    setLocation(400,200);  
    Container container = getContentPane();  
    container.add(new CheckBoxPanel());  
}  
}
```

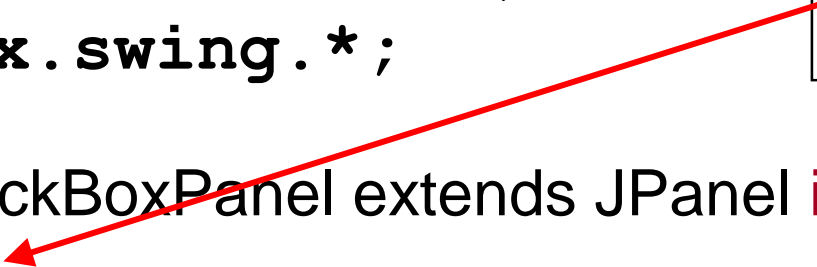
# CheckBoxPanel.java

CheckBoxPanel.java

```
package checkbox;  
import java.awt.*;  
import java.awt.event.*;  
import javax.swing.*;
```

Interface com um  
único método.

```
public class CheckBoxPanel extends JPanel implements {  
    ItemListener  
        public CheckBoxPanel() { }  
  
        public void itemStateChanged(ItemEvent evt) { }  
}
```



# Declaração dos atributos

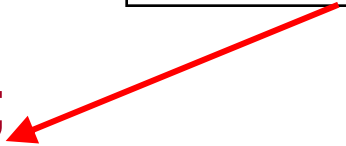
CheckBoxPanel.java

```
package checkbox;  
import java.awt.*;  
import java.awt.event.*;  
import javax.swing.*;  
public class CheckBoxPanel extends JPanel implements ItemListener {  
  
    private JLabel label;  
    private JCheckBox opNegrito;  
    private JCheckBox opItalico;  
    private int valNegrito;  
    private int valItalico;  
    private Font fonte;  
  
    public CheckBoxPanel() { }  
  
    public void itemStateChanged(ItemEvent evt) { }  
}
```

# Código do Construtor

```
public CheckBoxPanel() {  
    fonte = new Font("Serif", Font.PLAIN, 16);  
    label = new JLabel("Aula de Introdução a Swing");  
    label.setFont(fonte);  
    opNegrito = new JCheckBox("negrito");  
    opItalico = new JCheckBox("italico");  
    add(label);  
    add(opNegrito);  
    add(opItalico);  
    opNegrito.addItemListener(this);  
    opItalico.addItemListener(this);  
}
```

Ouvinte do evento.



# Código do método

```
public void itemStateChanged(ItemEvent evt) {  
    Object source = evt.getSource();  
    if(source == opNegrito){  
        if(opNegrito.isSelected())  
            valNegrito = Font.BOLD;  
        else  
            valNegrito = Font.PLAIN;  
    }  
    if(source == opItalico){  
        if(opItalico.isSelected())  
            valItalico = Font.ITALIC;  
        else  
            valNegrito = Font.PLAIN;  
    }  
}
```



# CheckBoxPrincipal.java

CheckBoxPrincipal.java

```
package checkbox;  
  
import javax.swing.*;  
  
public class CheckBoxPrincipal{  
  
    public static void main(String[] args){  
        CheckBoxFrame frame = new CheckBoxFrame();  
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
        frame.setVisible(true);  
    }  
}
```