



## Principais Classes Swing AWT

- As classes e interfaces localizadas nos pacotes:
  - ❑ java.awt
  - ❑ javax.swing
- são recursos para o desenvolvimento de GUIs (Graphic User Interface – Interface Gráfica do Usuário).



## Principais Classes Swing AWT

- Uma GUI é desenvolvida utilizando-se componentes, que são objetos visuais que possibilitam ao usuário realizar a interação com o programa por meio do mouse e do teclado. Os componentes mais comuns são:
  - ☐ rótulos (Label)
  - ☐ botões (button)
  - ☐ campo de texto (text field)
  - ☐ áreas de texto (text area)
  - ☐ caixas de checagem (check box)
  - ☐ botões de rádio (radio button)
  - ☐ listas (list) e menus



## Principais Classes Swing AWT

- Na versão 1.0 de Java, encontramos o pacote:
  - ❑ AWT (Abstract Window Toolkit – kit de ferramentas de janelas abstratas)
  - ❑ Este pacote contém todas as classes necessárias para a construção de GUIs
- Estratégia adotada à época:
  - ❑ Delegar a criação de elementos da interface do usuário e seu comportamento ao kit de ferramentas nativo do sistema operacional específico onde fossem criados



## Principais Classes Swing AWT

- Problemas encontrados com esta estratégia:
  - ❑ Os componentes apresentavam diferenças entre as diversas plataformas e isto dificultava a construção de GUIs
  - ❑ Ocorriam erros diferentes com os componentes em cada plataforma, obrigando os desenvolvedores a testar seus aplicativos em cada uma delas



## Principais Classes Swing AWT

- Devido a este problema, a Sun começou a desenvolver uma nova biblioteca de componentes que aperfeiçoasse o método adotado
- Essa biblioteca foi chamada de **Swing** e passou a ser o nome oficial do kit de componentes para a construção de GUIs



## Principais Classes Swing AWT

- O pacote Swing não é sobrecarregado com as complexas capacidades da plataforma em que são utilizados
- Os componentes AWT precisam contar com o sistema de janelas da plataforma local para determinar sua funcionalidade, sua aparência e seu comportamento



## Principais Classes Swing AWT

- Alguns componentes Swing ainda requerem interação direta com o sistema local de janelas, como é o caso de todas as classes derivadas de Window (como JFrame)
- Os componentes AWT continuam disponíveis nas últimas versões do J2SDK e encontra-se no pacote `java.awt`. Os componentes Swing foram dispostos no pacote `javax.swing`



## Component

- A classe Component representa as características comuns de todos os componentes, ou seja, de todos aqueles objetos que possuem uma representação gráfica, que podem interagir como o usuário e que possuem uma aparência e um comportamento





## Container

- A classe Container representa um contêiner, ou seja, um componente que pode abrigar dentro de si outros componentes
- Os contêineres criados com a classe Container funcionam como painéis e janelas, que abrigam outros contêineres e/ou componentes que aparecem na tela.



## Classe `javax.swing.JFrame`

- `JFrame` é um container de janela
- Para adicionar um componente a um `JFrame`:
  - ❑ Deve-se invocar o método `add()` do objeto que representa seu painel de conteúdo. Como esse objeto não pode ser acessado diretamente, deve-se recuperá-lo utilizando o método `getContentPane()` da janela.



## Classe javax.swing.JFrame

- Instrução para adição de um componente a um JFrame.

```
JFrame f = new JFrame ();
```

```
f.getContentPane().add(<nome do Componente>);
```



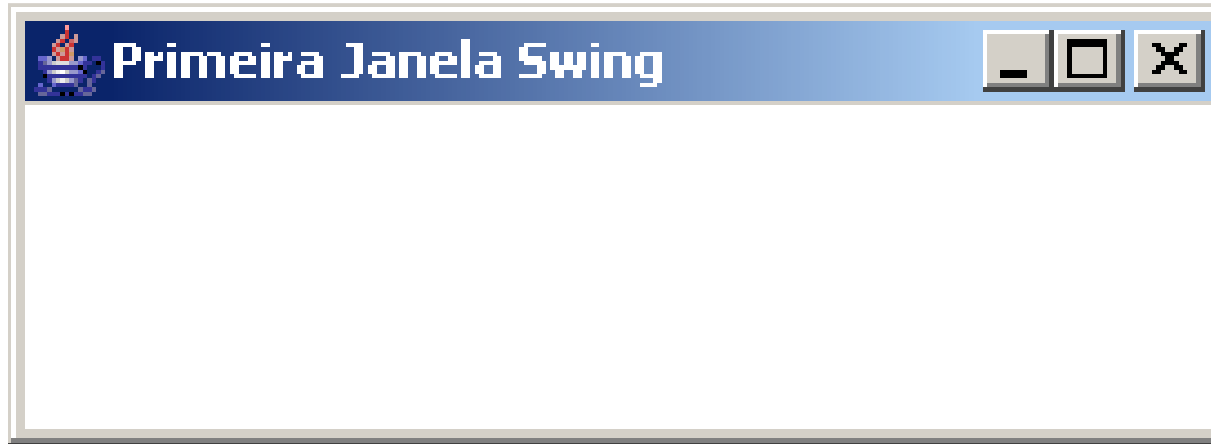
# Classe javax.swing.JFrame

```
import java.awt.*;
import javax.swing.*;
class TesteJFrame extends JFrame{
    public TesteJFrame(){
        //Titulo da janela
        setTitle("Primeira Janela Swing");
        //tamanho da janela
        setSize(275,100);
        // anula o layout padrao
        getContentPane().setLayout(null);
        //cor de fundo da janela no padrão RGB (Red Green Blue)
        getContentPane().setBackground(new Color(255,255,255));
        //provoca o termino da execução (encerra o programa)
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
    public static void main(String [] args){
        TesteJFrame janela = new TesteJFrame();
        janela.setVisible(true);
    }
}
```



## Classe `javax.swing.JFrame`

- Janela gerada pela execução do programa anterior:



- Esta janela não irá aparecer centralizada na tela e caso este seja o objetivo, devemos alterar o código da classe anterior



# Classe javax.swing.JFrame

```
public void centralizar() {  
    //obtem a altura e largura da resolução vídeo  
    Dimension screen =  
        Toolkit.getDefaultToolkit().getScreenSize();  
    //obtem a altura e largura da minha janela  
    Dimension janela = getSize();  
  
    if (janela.height > screen.height)  
        setSize(janela.width, screen.height);  
    if (janela.width > screen.width)  
        setSize(screen.width, janela.height);  
  
    setLocation((screen.width - janela.width)/2,  
                (screen.height - janela.height)/2);  
}
```



## **javax.swing.JLabel**

- Um rótulo (JLabel) é uma área para a exibição de um texto, uma imagem ou ambos. Ele não reage a eventos de entrada e, por isto, não pode receber o foco do teclado. Entretanto, pode incluir teclas de atalho para passar o foco para outro componente



# javax.swing.JLabel

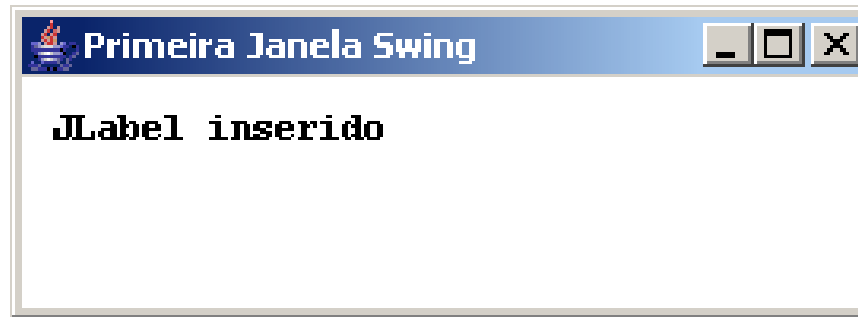
```
public JLabel criarJLabel() {  
    JLabel jl = new JLabel();  
    jl.setText("JLabel inserido");  
    jl.setLocation(10, 10);  
    jl.setSize(370, 50);  
    //Torna opaco o fundo do rótulo  
    jl.setOpaque(true);  
    jl.setBackground(new Color(255,255,255));  
    jl.setForeground(new Color(0,0,0));  
    jl.setFont(new Font("Courier new", Font.BOLD, 12));  
    jl.setToolTipText("JLabel Exemplo");  
    jl.setHorizontalAlignment(SwingConstants.LEFT);  
    jl.setVerticalAlignment(SwingConstants.TOP);  
  
    return jl;  
}
```





## javax.swing.JLabel

- Janela gerada pela execução do programa anterior:





# javax.swing.JTextField

- Esta classe representa um campo de texto para digitação pelo usuário, usualmente empregado para campos de cadastro de uma única linha. Métodos:
- **selectAll()** – seleciona todo texto
- **setHorizontalAlignment(JTextField.LEFT);** //0-Centro, 2-esquerda 4-Direita
- **setEditable(boolean)**
- **setText(texto)**
- **getText()**



# javax.swing.JTextField

```
JTextField jt = new JTextField();  
jt.setText("Insira texto aqui");  
jt.setHorizontalAlignment(JTextField.CENTER);  
//insere o JTextField na Janela  
add(jt);
```



## Exercício

- Crie uma tela com a altura e largura, cor de fundo e Título definidos pelo usuário definida pelo usuário.
- Insira na tela um rótulo nome e uma caixa de texto para o usuário digitar o nome.



# Solução

```
package gui.janelas;  
import java.awt.*;  
import javax.swing.*;  
public class TesteJFrame extends JFrame {  
    public TesteJFrame() {  
        int largura, altura;  
        largura = Integer.parseInt(JOptionPane.showInputDialog("Digite a  
        largura da tela:"));  
        altura = Integer.parseInt(JOptionPane.showInputDialog("Digite a  
        altura da tela:"));  
        setTitle("Primeira Janela Swing"); // Titulo da janela  
        setSize(largura , altura); // tamanho da janela  
        getContentPane().setLayout(null); // anula o layout padrao  
        // cor de fundo da janela no padrão RGB (Red Green Blue)
```



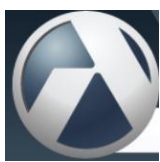
# Solução

```
int red = Integer.parseInt(JOptionPane.showInputDialog("Digite o  
valor para o vermelho:"));  
int green = Integer.parseInt(JOptionPane.showInputDialog("Digite o  
valor para o verde:"));  
int blue = Integer.parseInt(JOptionPane.showInputDialog("Digite o  
valor para o azul:"));  
getContentPane().setBackground(new Color(red , green ,  
blue));  
//217. 183, 216 - rosa  
//provoca o termino da execução (encerra o programa)  
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
setVisible(true);  
setLocationRelativeTo(null);  
//centraliza a tela  
}  
  
public static void main(String [] args){  
    new TesteJFrame(); } }
```



## **javax.swing.JButton**

- Um botão (button) é um componente que pode ser pressionado pelo usuário, utilizando o mouse ou teclado, para acionar uma ação específica



# javax.swing.JButton

```
Jbutton b1 = new JButton();  
b1.setText("Gravar");  
//50 posição x, 30 posição y, 100 largura, 30 altura  
b1.setBounds(50, 30, 100, 30); //D Esq, D Topo, larg, alt  
b1.setBackground(new Color(0,0,170));  
b1.setForeground(Color.YELLOW);  
b1.setFont(new Font("Helvetica", Font.BOLD, 12));  
b1.setToolTipText("Botao b1");  
b1.setHorizontalAlignment(SwingConstants.CENTER);  
b1.setVerticalAlignment(SwingConstants.CENTER);  
b1.setEnabled(false); //Botão desabilitado.  
b1.setMnemonic('G'); //Tecla de atalho
```





## **javax.swing.JButton**

➤ Vários métodos da classe JButton realizam as mesmas tarefas realizadas por métodos com nomes idênticos da classe JLabel e já foram analisados. Veja abaixo um resumo desses métodos:

- ❑ `setText()`: define o texto do botão
- ❑ `setBounds()`: define o tamanho e a posição em seu contêiner
- ❑ `setBackground()`: define a cor de fundo



## **javax.swing.JButton**

- ❑ **setForeground()**: define a cor do texto
- ❑ **setFont()**: define o tipo, o estilo e o tamanho da fonte
- ❑ **setToolTipText()**: define uma dica
- ❑ **setHorizontalAlignment()**: define o alinhamento horizontal do conteúdo
- ❑ **setVerticalAlignment()**: define o alinhamento vertical do conteúdo
- ❑ **setEnabled()**: exige um argumento booleano e serve para habilitar ou desabilitar o botão



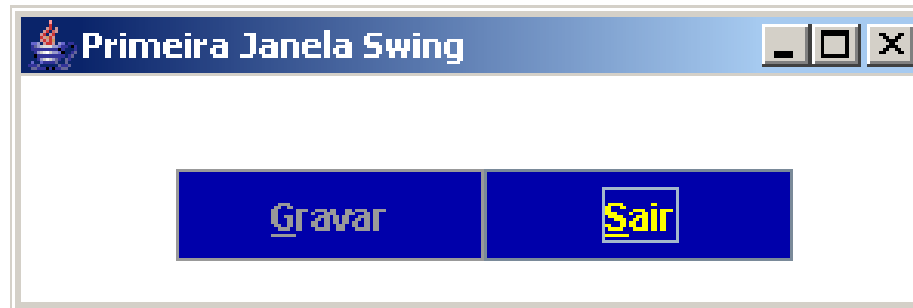
## **javax.swing.JButton**

- ❑ **setMnemonic()**: define um atalho de teclado para o botão. Deve-se informar para esse método um caracter que exista no seu texto. Posteriormente, o botão poderá ser acessado, a partir de qualquer ponto da janela, pressionando-se a tecla ALT em conjunto com esse caracter.



## javax.swing.JButton

- Crie o Botão b2 para que a imagem fique como na tela a seguir. Saída gerada:





# Exercício

Crie a tela a seguir.

CPF	<input type="text"/>	RG	<input type="text"/>
NOME	<input type="text"/>		
LOGRADOURO	<input type="text"/>	ENDEREÇO	<input type="text"/>

Gravar

Cancelar



## Listas e Caixas de Seleção

- Listas de seleção são objetos que nos permitem selecionar um item dentre uma **coleção** existente em uma lista, comumente vemos este tipo de facilidade em aplicativos. A classe que nos permite usar esta facilidade em Java é a classe **JList**, como veremos no exemplo a seguir:



# Listas e Caixas de Seleção

```
import java.awt.*;  
import java.awt.event.*;  
import javax.swing.*;  
import javax.swing.event.*;  
public class ExJList extends JFrame implements  
ListSelectionListener, ActionListener  
{  
    // Criando objetos Label, Texto e Botões  
    JLabel l1; JTextField t1; JButton bquant, bindice, bclear;  
    JList lista; // Criando objeto Lista  
    // Criando objetos listModel a partir da classe DefaultListModel  
    DefaultListModel listModel;  
  
    public static void main(String args[]){  
        new ExJList ();  
    }  
}
```



# Listas e Caixas de Seleção

```
public ExJList ()  
{  
    setSize(300,250);  
    setTitle("Uso do JList");  
    t1 = new JTextField();  
    t1.addActionListener(this); //adicionando o evento no objeto  
    l1 = new JLabel("Sem selecao");  
    bquant = new JButton("Quantidade de itens");  
    bquant.addActionListener(this); //adicionando o evento no objeto  
    bindice = new JButton("Indice selecionado");  
    bindice.addActionListener(this); //adicionando o evento no objeto  
    bclear = new JButton("Remove item");  
    bclear.addActionListener(this); //adicionando o evento no objeto
```





# Listas e Caixas de Seleção

```
listModel = new DefaultListModel(); //container com os itens
listModel.addElement("Banana");
listModel.addElement("Pera");
listModel.addElement("Maça");
listModel.addElement("Uva");
lista = new JList(listModel); //adicionando o listModel na Lista.
lista.addListSelectionListener(this); //Adicionando o evento na lista
// Criando painel do tipo barra de rolagem
JScrollPane Painel = new JScrollPane(lista);
setLayout(new GridLayout(6,1)); //Criando um Layout do tipo Grid
add(l1);
add(t1);
add(Painel);
add(bquant);
add(bindice);
add(bclear);
```



# Listas e Caixas de Seleção

```
setLocationRelativeTo(null);  
setVisible(true);  
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
}  
// Tratando os eventos  
public void actionPerformed(ActionEvent e)  
{  
if (e.getSource()==t1)//verifica se foi disparado objeto t1  
{  
listModel.addElement(t1.getText()); //adiciona itens a lista  
t1.setText(""); //Limpa a caixa de texto  
}  
if (e.getSource()==bquant)  
//verifica a quantidade de itens da lista  
t1.setText("Quantidade: " + listModel.getSize());
```



# Listas e Caixas de Seleção

```
if (e.getSource()==bindice)
```

```
//verifica o índice do item selecionado
```

```
t1.setText("Indice selecionado: " + lista.getSelectedIndex());
```

```
if (e.getSource()==bclear)
```

```
{
```

```
int resp = JOptionPane.showConfirmDialog(null, "Confirma a  
exclusão do item: " +
```

```
lista.getSelectedValue());
```

```
if(resp==0)
```

```
{
```

```
int index = lista.getSelectedIndex();
```

```
l1.setText("Removido : "+ lista.getSelectedValue());
```

```
listModel.remove(index);
```

```
} } }
```




# Listas e Caixas de Seleção

```
public void valueChanged(ListSelectionEvent e)  
{  
    l1.setText("Índice Seleccionado :  
        "+lista.getSelectedValue());  
}  
}
```



# Listas - Interface

 **Uso do JList** — □ ×

**Índice Seleccionado : Banana**

Quantidade: 4

Banana

Pera

▲

▼

Quantidade de itens

Índice seleccionado

Remove item



# Listas e Caixas de Seleção

## Métodos utilizados no exemplo

Método	Ação
<code>getSelectedvalue()</code>	Retorna o texto do item selecionado
<code>getSelectedIndex()</code>	Retorna o índice do item selecionado
<code>setSelectedIndex()</code>	Seleciona o índice do item selecionado
<code>setSelectedInterval(int, int)</code>	Seleciona vários índices conforme especificado
<code>isSelectionEmpty()</code>	Verifica se há ou não item selecionado
<code>addElement(String)</code>	Adiciona um novo item a lista
<code>getSize()</code>	Retorna com a quantidade total de itens da lista
<code>remove(int)</code>	Remove o item de acordo com seu índice
<code>setSelectionMode(int)</code>	1 seleção sequencial (Shift) 2 seleção alternada (Ctrl)



## Exercício

- Com base no exemplo anterior crie uma tela onde o usuário poderá selecionar e incluir na Lista de Seleção os esportes que pratica.



## Listas e Caixas de Seleção

- Além das listas de seleção comumente nos deparamos com caixas de seleção conhecidas como **ComboBox**.
- A grande diferença entre uma lista de seleção e uma caixa de seleção é que no caso das caixas de seleção **não é permitido** a seleção de mais de um item.
- A classe que nos permite a criação de caixas de seleção é a **JComboBox** como veremos no exemplo a seguir:





# Listas e Caixas de Seleção

Uso do JComboBox

<b>Índice: 0</b>	Branco ▼
Mostra Texto	Remove Item
Mostra Índice	Remove Todos
Adiciona Item	
Quant. Itens	5



# Listas e Caixas de Seleção

```
import java.awt.*;  
import java.awt.event.*;  
import javax.swing.*;  
public class ExComboBox extends JFrame implements ActionListener,  
ItemListener  
{  
    JLabel l1;  
    JTextField t1,t2,t3;  
    JComboBox combo;  
    JButton b1,b2,b3,b4,b5,b6;  
  
    public static void main(String args[]){  
        new ExComboBox();  
    }  
}
```



# Listas e Caixas de Seleção

ExComboBox ()

{

setTitle("Uso do JComboBox");setSize(400,170);  
getContentPane().setBackground(new Color(190,190,190));

l1 = new JLabel("Conteudo");

l1.setHorizontalAlignment(SwingConstants.CENTER);

l1.setForeground(Color.blue);

l1.setFont(new Font("Arial", Font.BOLD, 15));

b1 = new JButton("Mostra Texto");

b1.addActionListener(this);

b2 = new JButton("Mostra Índice");

b2.addActionListener(this);

b3 = new JButton("Adiciona Item");

b3.addActionListener(this);

b4 = new JButton("Remove Item");

b4.addActionListener(this);

b5 = new JButton("Remove Todos");

b5.addActionListener(this);

b6 = new JButton("Quant. Itens");

b6.addActionListener(this);



# Listas e Caixas de Seleção

```
t1 = new JTextField(); t2 = new JTextField();
t1.setHorizontalAlignment(SwingConstants.CENTER);
t2.setHorizontalAlignment(SwingConstants.CENTER);
String[] cores = {"Branco", "Vermelho", "Azul", "Verde"};
combo = new JComboBox(cores);
combo.addItemListener(this);
getContentPane().setLayout(new GridLayout(5,2));
getContentPane().add(l1);
getContentPane().add(b1);
getContentPane().add(b2);
getContentPane().add(b3);
getContentPane().add(b6);
setLocationRelativeTo(null);
setVisible(true);
}
```

```
getContentPane().add(combo);
getContentPane().add(b4);
getContentPane().add(b5);
getContentPane().add(t1);
getContentPane().add(t2);
```



# Listas e Caixas de Seleção

```
public void actionPerformed(ActionEvent e)
{
    if (e.getSource()==b1)
        l1.setText("Texto: "+combo.getSelectedItem());
    if (e.getSource()==b2)
        l1.setText("Índice: " + combo.getSelectedIndex());
    if (e.getSource()==b3)
        if (t1.getText().length()!=0)
            {combo.addItem(t1.getText());//adiciona item
            t1.setText(""); }//limpa o texto de t1
    if (e.getSource()==b4)
        combo.removeItemAt( (combo.getSelectedIndex())); // remove o item selecionado
    if (e.getSource()==b5)
        combo.removeAllItems(); //remove todos itens
    if (e.getSource()==b6)
        t2.setText(""+combo.getItemCount()); } //conta a quantidade total de itens
public void itemStateChanged(ItemEvent e)
{ t1.setText(""+combo.getSelectedItem()); } } //mostra o item selecionado
```



# javax.swing.JTextField

- Insira as Caixas de Seleção **Sexo** e **Falecido** na tela anteriormente criada conforme vemos na imagem a seguir.

The screenshot shows a Java Swing window titled "ATUALIZA DADOS CADASTRAIS" with standard window controls (minimize, maximize, close). The form contains the following fields:

- CPF**: A text field with a mask of two digits, a dot, two more digits, a dash, and a final digit.
- RG**: A text field with a mask of four digits, a dash, and a final digit.
- NOME**: A single-line text field.
- SEXO**: A dropdown menu.
- NASCIMENTO**: A text field with a mask of two digits, a dot, two more digits, a dash, and a final digit.
- FALECIDO**: A dropdown menu.
- PROF**: A single-line text field.
- SITUAÇÃO**: A single-line text field.



## Caixas de Checagem - CheckBox

Outro recurso muito utilizado em formulários são as caixas de checagem ou como conhecemos **CheckBox** elas são muito utilizadas quando é necessário obter uma resposta de uma série de perguntas que podem ser classificadas como sim ou não (true ou false), como por exemplo:

Times de futebol que gosta: São Paulo, Palmeiras, Santos ou Outros.

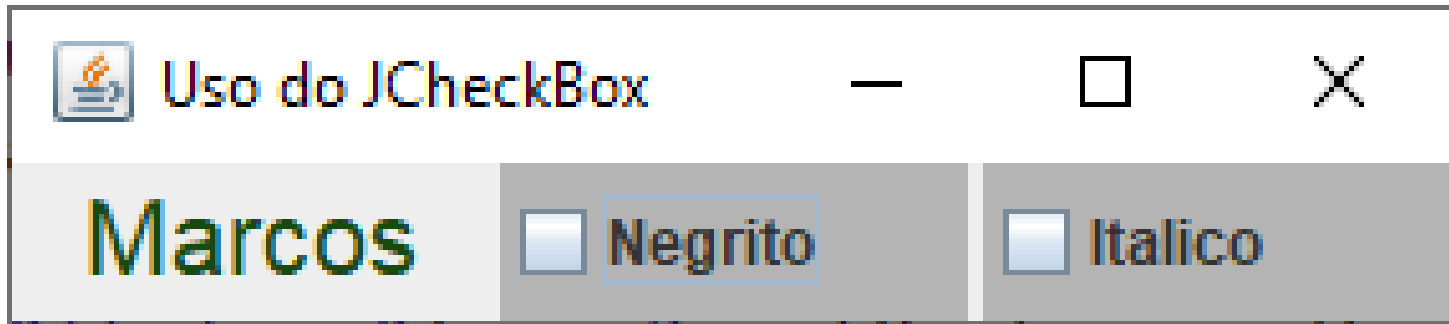
A cada clique na caixa indicará uma resposta positiva para os times que a pessoa entrevistada gosta (cabe ressaltar aqui que não colocamos o nome daquele outro time em respeito ao entrevistado).

A classe utilizada para implementar esta facilidade é a classe **JCheckBox** cujo exemplo veremos abaixo:



# Caixas de Checagem - CheckBox

Aparência da Tela







# Caixas de Checagem - CheckBox

```
import java.awt.*;      import java.awt.event.*; import javax.swing.*;
public class ExCheckBox extends JFrame implements ItemListener{
    JLabel l1;
    JCheckBox c1,c2;
    static int negrito=0,italico=0;
    public static void main(String args[]) {
        new ExCheckBox ();
    }
```

**ExCheckBox ()**

```
{
    setBackground(new Color(180,180,180));
    setTitle("Uso do JCheckBox");
    setSize(300,70);
    setLayout(new FlowLayout(FlowLayout.CENTER));
```



## Caixas de Checagem - CheckBox

```
l1 = new JLabel(JOptionPane.showInputDialog("Digite um texto"));
l1.setHorizontalAlignment(SwingConstants.CENTER);
l1.setFont(new Font("Arial",Font.PLAIN,20));
l1.setForeground(new Color(26,72,17));
setLayout(new GridLayout(1, 3, 3, 3));
c1 = new JCheckBox("Negrito");
c1.setBackground(new Color(180,180,180));
c1.addItemListener(this);
c2 = new JCheckBox("Italico");
c2.setBackground(new Color(180,180,180));
c2.addItemListener(this);
add(l1);      add(c1);      add(c2);
setLocationRelativeTo(null);  setVisible(true);
setDefaultCloseOperation(EXIT_ON_CLOSE); }
```




## Caixas de Checagem - CheckBox

```
public void itemStateChanged(ItemEvent e){  
    if(e.getSource()==c1){  
        if(e.getStateChange()==ItemEvent.SELECTED)  
            negrito=Font.BOLD;  
        else  
            negrito=Font.PLAIN;        }  
    if(e.getSource()==c2){  
        if(e.getStateChange()==ItemEvent.SELECTED)  
            italico=Font.ITALIC;  
        else  
            italico=Font.PLAIN;        }  
    l1.setFont(new Font("Arial",negrito+italico,20));  
}    }
```



## Exercício

- Usando como base o Exercício de Lista para Seleção dos esportes que pratica, altere-o para que o usuário possa selecionar os times que torce utilizando Caixas de Checagem do tipo CheckBox.

 **Times que torce** — □ ×

---

- ☐ **São Paulo**
- ☐ **Palmeiras**
- ☐ **Santos**
- ☐ **Corinthians**
- ☐ **Sport**
- ☐ **Santa Cruz**
- ☐ **Cruzeiro**



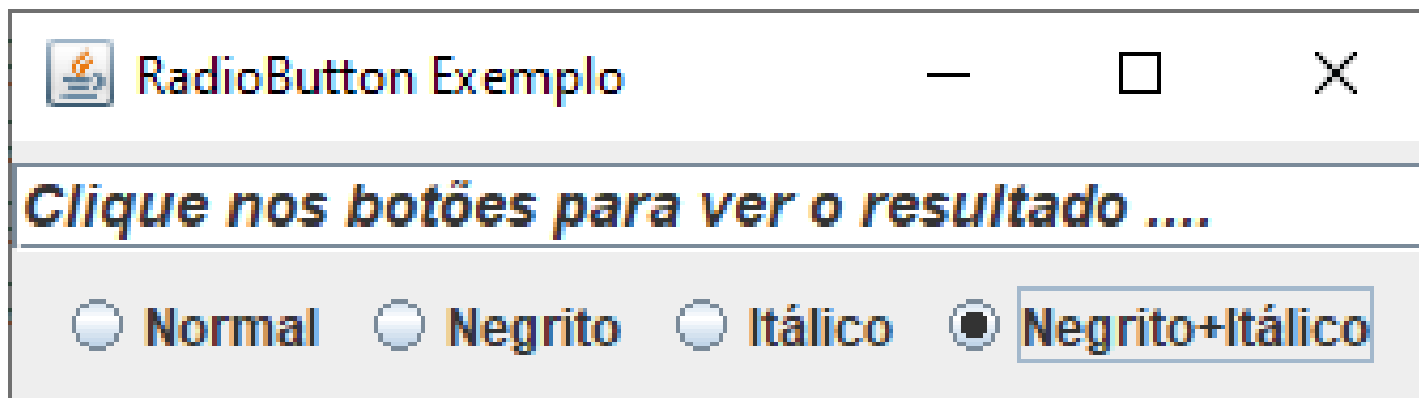
## Botões de Rádio - RadioButton

- Para criar botões de opção usaremos instancias da classe **JRadioButton** uma característica deste objeto é que ao contrário das caixas de seleção este botão é utilizado quando somente uma opção pode ser feita como por exemplo sexo, ou a pessoa é do sexo masculino ou é do sexo feminino.
- Vejamos exemplo a seguir como se dá o uso desta classe e seus componentes no Java.



# Caixas de Checagem - CheckBox

Interface do exemplo:





# Caixas de Checagem - CheckBox

```
import java.awt.*;import java.awt.event.*;import javax.swing.*;  
public class ExJRadioButton extends JFrame{  
    private JRadioButton normal, negrito, italico, itaNeg;  
    private ButtonGroup radioGrupo;  
    private JTextField texto;//criação do objeto tipo JTextField  
    private Font normalF , negritoF, italicoF, itaNegF;  
  
    private Container c ; //criação do objeto tipo Container  
    private RadioButtonTratar tratarRB ;  
    public ExJRadioButton()//criação do método construtor da classe  
    {  
        super( "RadioButton Exemplo" );  
        c = getContentPane();  
        texto = new JTextField( "Clique nos botões para ver o resultado ....", 30 );  
        c.setLayout( new FlowLayout() );    c.add( texto );  
        tratarRB = new RadioButtonTratar();//instanciação do objeto  
        radioGrupo = new ButtonGroup();//instanciação do objeto
```



# Caixas de Checagem - CheckBox

```
// criação dos JRadioButtons
normal = adicRadio( "Normal", true);
negrito = adicRadio( "Negrito", false);
italico = adicRadio( "Itálico", false);
itaNeg = adicRadio( "Negrito+Itálico", false);

normalF = new Font( "Arial", Font.PLAIN, 14 );
negritoF = new Font( "Arial", Font.BOLD, 14 );
italicoF = new Font( "Arial", Font.ITALIC, 14 );
itaNegF = new Font( "Arial", Font.BOLD + Font.ITALIC, 14 );

texto.setFont( normalF );
setSize( 350, 100 );
setVisible(true);
setLocationRelativeTo(null);
setDefaultCloseOperation(EXIT_ON_CLOSE); }

public static void main( String args[] )
{new ExJRadioButton(); }
```





## Caixas de Checagem - CheckBox

```
private JRadioButton adicRadio( String nome, boolean b)
{JRadioButton radioBot = new JRadioButton (nome, b);
radioBot.addItemListener( tratarRB );
c.add(radioBot);//adiciona o botao criado ao container
radioGrupo.add( radioBot );//adiciona o botao criado ao grupo RadioGrupo
return radioBot; }//retorna o foco para o objeto radioBot

private class RadioButtonTratar implements ItemListener {
    public void itemStateChanged( ItemEvent e )
    {if ( e.getSource() == normal )
        texto.setFont( normalF );//formate o texto como normal
    else if ( e.getSource() == negrito )
        texto.setFont( negritoF );
    else if ( e.getSource() == italico )
        texto.setFont( italicoF );
    else if ( e.getSource() == itaNeg )
        texto.setFont( itaNegF ); } }}
```



## Exercício

- Crie a tela vista a seguir utilizando os componentes que aprendemos até aqui.
- Implemente as seguintes regras de negócio:
  - ☐ Valide o CPF e o RG.
  - ☐ Somente habilite o botão **enviar** se o usuário preencher todos os campos
  - ☐ Após isto exiba as informações Digitadas e crie um botão enviar.
  - ☐ Após o clique deve dar a mensagem "Dados enviados com sucesso".



# Exercício

## DADOS BÁSICOS

**Nome \***

Seu nome completo, *exatamente* como aparece em sua carteira de identidade.

**Nascimento \***

▼

▼

▼

Informe a data do seu nascimento.

**Sexo \***

- ☐ Masculino
- ☐ Feminino

**RG**

Seu RG.

**CPF**

Seu CPF.



## Exemplo de Validação

- **RG** Para descobrir o **dígito** de controle do seu **RG**, basta multiplicar cada dígito como visto abaixo soma-los e ao final dividir por 11. Caso o resto da divisão seja menor que dez o dígito será o resto senão o dígito será “A”.

Registro Geral								DC
1	8	3	7	9	2	2	1	A
9	8	7	6	5	4	3	2	
9	64	21	42	45	8	6	2	
soma =	197							
resto =	10							

## Exemplo de Validação

➤ CPF veja o detalhamento no site: <https://campuscode.com.br/conteudos/o-calculo-do-digito-verificador-do-cpf-e-do-cnpj#:~:text=O%20c%C3%A1lculo%20de%20valida%C3%A7%C3%A3o%20do,2%20e%20somamos%20esse%20resultado.>

[illegible]



## Layouts de Tela

### ➤ Gerenciamento de layout

- ❑ Nos exemplos anteriores, o tamanho e o posicionamento dos componentes nas janelas foram definidos através dos métodos `setSize()`, **`setBounds()`** e **`setLocation()`**. Utilizou-se o sistema de coordenadas x e y para posicionar os componentes e a definição de valores absolutos para sua altura e largura.
- ❑ Entretanto, essa forma de organização esbarra em vários inconvenientes.



# Layouts de Tela

## ➤ Gerenciamento de layout

☐ Deve-se evitar o uso de valores absolutos para definição do tamanho e posição de componentes porque eles comprometem duas importantes características da qualidade do software:

☐ Portabilidade

☐ Manutenção



## Layouts de Tela

### ➤ Gerenciamento de layout

- ❑ Em função destes problemas, é aconselhável evitar o uso de valores absolutos para o dimensionamento e posicionamento de componentes. Para isso, deve-se substituir esse procedimento pelo uso dos gerenciadores de leiaute (layout) disponíveis na API Java





## Layouts de Tela

- Gerenciadores de Layout (layout): São classes que aplicam um conjunto de regras predefinidas para determinar o tamanho e a posição de cada componente em um contêiner
- Objetivo
  - ❑ Organizar os componentes em seu contêiner para uma apresentação adequada.
  - ❑ Além de serem mais eficientes, eles fornecem capacidades que são muito mais fáceis de utilizar do que a definição de tamanho e posição por valores absolutos.



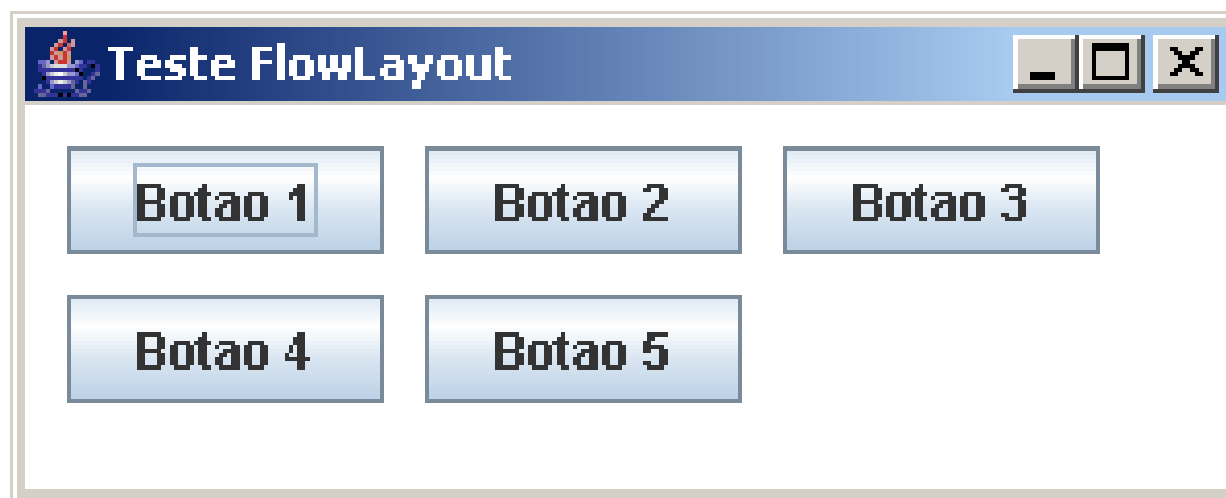
## Layout de Fluxo (FlowLayout)

- O leiaute (layout) de fluxo (FlowLayout) alinha os componentes em um fluxo, da esquerda para a direita e de cima para baixo, muito parecido com o texto de um parágrafo.
- Esse tipo de leiaute (layout) é representado pela classe `java.awt.`**FlowLayout**



## Layout de Fluxo (FlowLayout)

➤ Veja este exemplo:





# Exemplo FlowLayout

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class ExFlowLayout extends JFrame
{
    private JButton botao1, botao2, botao3, botao4,
    botao5;

    private Container c; //criação do objeto tipo Container

    public ExFlowLayout()//criação do método construtor da classe
    {
        super("Exemplo de Layout de Fluxo");
        c = getContentPane();
```



# Exemplo FlowLayout

//definindo o layout como de Fluxo

```
c.setLayout( new FlowLayout());  
botao1 = criarBotao("Botão 1", '1');  
botao2 = criarBotao("Botão 2", '2');  
botao3 = criarBotao("Botão 3", '3');  
botao4 = criarBotao("Botão 4", '4');  
botao5 = criarBotao("Botão 5", '5');
```

// criação dos botões

```
c.add(botao1);  
c.add(botao2);  
c.add(botao3);  
c.add(botao4);  
c.add(botao5);
```



# Exemplo FlowLayout

// criação dos botões

```
c.add(botao1);  
c.add(botao2);  
c.add(botao3);  
c.add(botao4);  
c.add(botao5);  
setSize(350, 120);  
setVisible(true);  
setLocationRelativeTo(null);  
setDefaultCloseOperation(EXIT_ON_CLOSE);  
}  
public static void main( String args[] )  
{  
new ExFlowLayout();  
}
```



## Exemplo FlowLayout

```
public static JButton criarBotao(String texto, char c)
{
    JButton botao = new JButton(texto);
    botao.setBackground(new Color(0,0,170));
    botao.setForeground(Color.YELLOW);
    botao.setFont(new Font("Helvetica", Font.BOLD, 14));
    botao.setToolTipText("Botao de comando");
    botao.setHorizontalAlignment(SwingConstants.CENTER);
    botao.setVerticalAlignment(SwingConstants.CENTER);
    botao.setMnemonic(c);
    return botao;
}
```



## Layout de Bordas (BorderLayout)

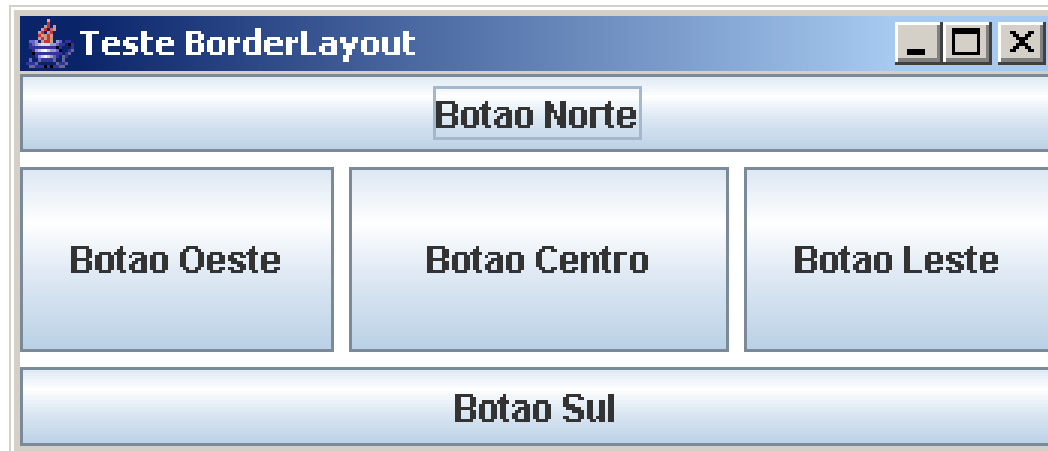
- O leiaute (layout) de bordas é representado pela classe `java.awt.BorderLayout`. Ela posiciona e redimensiona os componentes para ajustarem-se a cinco regiões de seu contêiner: norte, sul, leste, oeste e centro. Cada uma dessas regiões pode conter somente um componente e sua localização é definida por uma constante:
  - ☐ `BorderLayout.NORTH`: representa a região norte (superior)
  - ☐ `BorderLayout.SOUTH`: representa a região sul (inferior)
  - ☐ `BorderLayout.EAST`: representa a região leste (direita)
  - ☐ `BorderLayout.WEST`: representa a região oeste (esquerda)
  - ☐ `BorderLayout.CENTER`: representa a região central





## Layout de Bordas (BorderLayout)

➤ Veja este exemplo:





# Exemplo

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

public class TesteBorderLayout extends JFrame{
    public TesteBorderLayout(){
        setTitle("Teste BorderLayout");
        setSize(350,150);
        Container c = getContentPane();
        c.setBackground(new Color(255,255,255));
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);

        BorderLayout bl = new BorderLayout(5,5);
        c.setLayout(bl);

        JButton b1 = criarBotao("Botao Norte" , BorderLayout.NORTH);
        JButton b2 = criarBotao("Botao Sul" , BorderLayout.SOUTH);
        JButton b3 = criarBotao("Botao Leste", BorderLayout.EAST);
        JButton b4 = criarBotao("Botao Oeste", BorderLayout.WEST);
        JButton b5 = criarBotao("Botao Centro", BorderLayout.CENTER );
        setVisible(true);
    }

    private JButton criarBotao(String texto, String posicao) {
        JButton b1 = new JButton(texto);
        add(b1, posicao);
        return b1;
    }

    public static void main(String [] args){
        new TesteBorderLayout();
    }
}
```



## Layout de Grade (GridLayout)

- Um gerenciador de leiaute (layout) de grade é aquele que dispõe os componentes em um contêiner em forma de uma grade retangular.
- O contêiner é dividido em células retangulares de tamanhos iguais e cada componente é disposto em uma célula diferente.



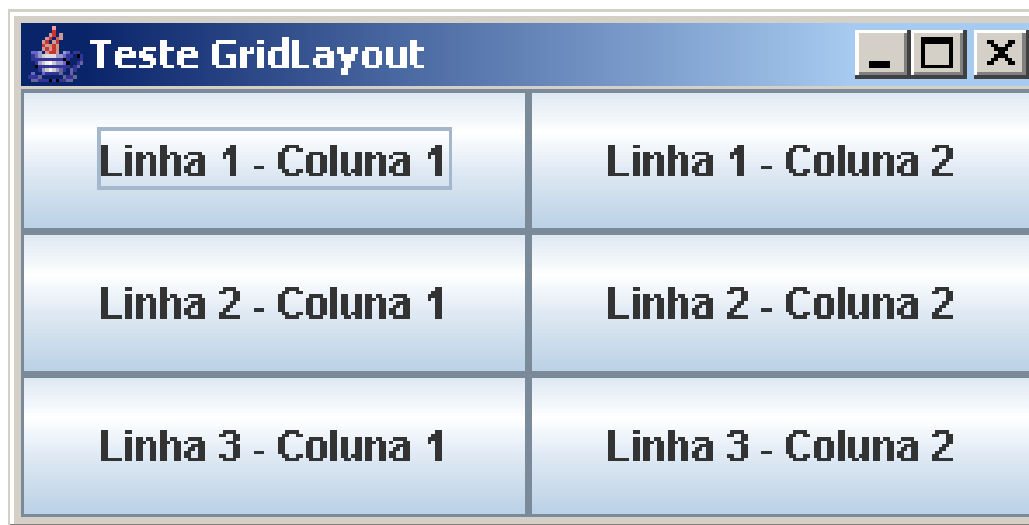
## Layout de Grade (GridLayout)

- O leiaute (layout) de grade (GridLayout) é representado pela classe `java.awt.GridLayout` e, do mesmo modo que no leiaute (layout) de fluxo, os componentes são dispostos da esquerda para a direita, mas ocupam uma célula da grade e ajustam seu tamanho a ela.



## Layout de Grade (GridLayout)

➤ Veja este exemplo no documento escrito!



A screenshot of a Java Swing window titled "Teste GridLayout". The window contains a grid of six rectangular cells arranged in three rows and two columns. Each cell has a light blue gradient background and a thin black border. The text in each cell is black and centered. The top-left cell is highlighted with a white border. The window has a standard title bar with a blue background, the title text, and three control buttons (minimize, maximize, close) on the right.

Linha 1 - Coluna 1	Linha 1 - Coluna 2
Linha 2 - Coluna 1	Linha 2 - Coluna 2
Linha 3 - Coluna 1	Linha 3 - Coluna 2



# Exemplo

```
import java.awt.*;
import javax.swing.*;
public class TesteGridLayout extends JFrame{
    public TesteGridLayout(){
        setTitle("Teste GridLayout");
        setSize(300,150);
        getContentPane().setBackground(new Color(180,189,255));
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);

        GridLayout gl = new GridLayout(3,2,0,0);
        getContentPane().setLayout(gl);

        JButton b1 = criarBotao("Linha 1 - Coluna 1");
        JButton b2 = criarBotao("Linha 1 - Coluna 2");
        JButton b3 = criarBotao("Linha 2 - Coluna 1");
        JButton b4 = criarBotao("Linha 2 - Coluna 2");
        JButton b5 = criarBotao("Linha 3 - Coluna 1");
        JButton b6 = criarBotao("Linha 3 - Coluna 2");
        setVisible(true);
    }
    private JButton criarBotao(String texto) {
        JButton botao = new JButton(texto);
        add(botao);
        return botao;
    }
    public static void main(String [] args){
        new TesteGridLayout();
    }
}
```



## **Painéis (Panel)**

- Os painéis agem como contêineres que servem para dividir a janela. Eles são utilizados para possibilitar maior controle da organização de interfaces gráficas mais complexas e, em conjunto com os gerenciadores de leiaute (layout), permitem mais exatidão no posicionamento dos componentes



## Painéis (Panel)

- A classe `javax.swing.JPanel` representa um contêiner genérico. Pode-se dizer que todo painel, representado como um objeto da classe **JPanel**, é um contêiner e pode abrigar componentes, inclusive outros painéis.





## Painéis (Panel)

➤ Veja este exemplo





## Exemplo - Implementação

```
import java.awt.*;  
import javax.swing.*;  
class TesteJPanel extends JFrame{  
    JButton b1, b2;  
    JPanel p1, p2;  
    public TesteJPanel() {  
        //Titulo da janela  
        setTitle("Teste JPanel");  
        //tamanho da janela  
        setSize(200,200);  
        // anula o layout padrao  
        getContentPane().setLayout(null);  
        //cor de fundo da janela no padrão RGB (Red Green Blue)  
        getContentPane().setBackground(new Color(255,255,255));  
    }  
}
```



## Exemplo - Implementação

```
//provoca o termino da execução (encerra o programa)
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
centralizar();
p1 = new JPanel();
p1.setLayout(null);
p1.setBounds(10, 10, 140, 70);
p1.setBackground(Color.BLUE);

p2 = new JPanel();
p2.setLayout(null);
p2.setBounds(10,90,140,70);
p2.setBackground(Color.YELLOW);

b1 = new JButton("Gravar");
b1.setBounds(20, 20, 100, 30);
b2 = new JButton("Sair");
b2.setBounds(20, 20, 100, 30);
```



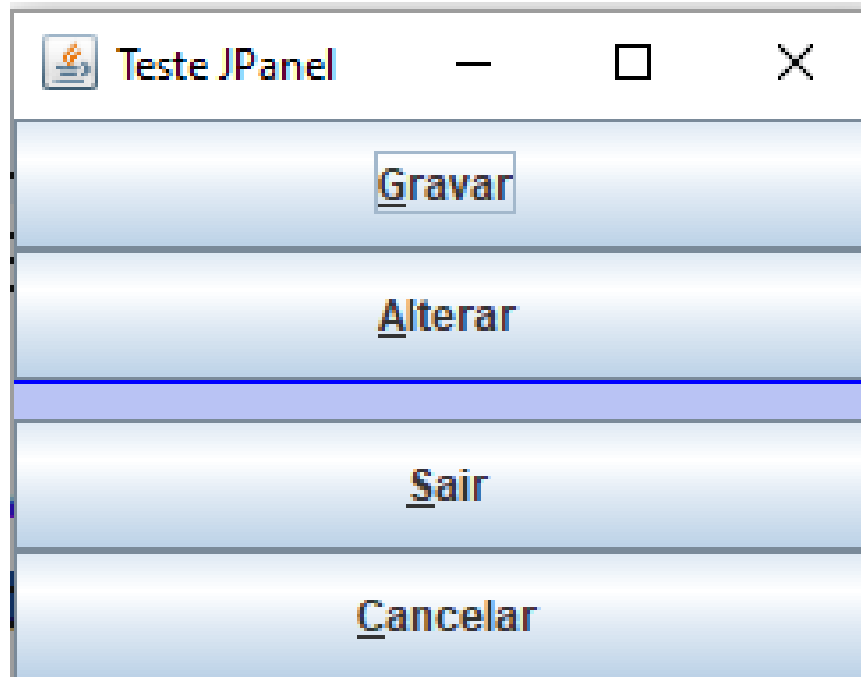
## Exemplo - Implementação

```
p1.add(b1);
    p2.add(b2);
    getContentPane().add(p1);
    getContentPane().add(p2);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    //obtem a altura e largura da resolução vídeo
    Dimension screen = Toolkit.getDefaultToolkit().getScreenSize();
    //obtem a altura e largura da minha janela
    Dimension janela = getSize();
public static void main(String [] args){
    TesteJPanel janela = new TesteJPanel();
    janela.setVisible(true);
}
}
```



## Exercício

- Crie a tela vista abaixo:



- Temos dois painéis o primeiro com os botões gravar e alterar e o Segundo com o sair e cancelar.
- Utilizar gerenciadores de layout na construção da tela e painéis.



# Solução

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

public class ExercicioPaineis extends JFrame{
    JButton b1, b2, b3, b4;    JPanel p1, p2;
    public ExercicioPaineis(){
        setTitle("Teste JPanel");
        setSize(260,200);
        getContentPane().setLayout(new GridLayout(2,1,0,10));
        getContentPane().setBackground(new Color(185,195,244));
        b1 = criarBotao("Gravar", 'G');
        b2 = criarBotao("Sair", 'S');
        b3 = criarBotao("Cancelar", 'C');
        b4 = criarBotao("Alterar", 'A');

        p1 = criarPainel(Color.BLUE, b1, 2);
        p2 = criarPainel(Color.YELLOW, b2, 2);

        setVisible(true);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
    }
    private JButton criarBotao(String texto, char atalho) {
        JButton botao = new JButton(texto);
        botao.setMnemonic(atalho);
        return botao;
    }
}
```



# Solução

```
private JPanel criarPainel(Color corFundo, JButton botao, int linhas) {  
    JPanel painel = new JPanel();  
    painel.setLayout(new GridLayout(linhas,1,0,0));  
    painel.setBackground(corFundo);  
    painel.add(botao);  
    add(painel);  
    return painel;  
}  
public static void main(String [] args){  
    new ExercicioPaineis();  
}  
}
```



## Eventos de Ação

- Eventos de ação são eventos de alto nível gerados por um componente (como um botão) quando a ação especificada por ele ocorrer (como ser pressionado). Quando um evento desse tipo ocorre, o componente gerador envia um objeto da classe `ActionEvent` para cada ouvinte, registrado através do método `addActionListener()`.



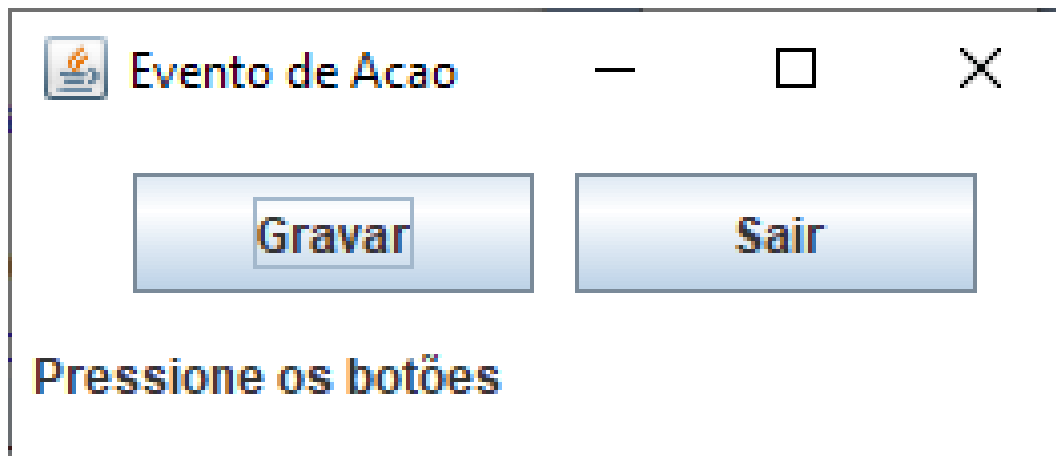


## Eventos de Ação

- Cada vez que um botão é pressionado, o ouvinte será notificado através de um objeto do tipo `ActionEvent`, que contém informações acerca do evento ocorrido.
- Se ele é ouvinte de dois ou mais botões, é preciso determinar qual deles foi pressionado para tomar a decisão correta.
- Isso pode ser feito invocando-se o método `getSource()`, que retornará uma referência ao objeto que gerou o evento



## Exemplo – Evento de Ação





# Eventos de Ação - Implementação

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
class TesteEventoAcao extends JFrame implements ActionListener{
    JLabel l1;
    JButton b1, b2;
    int i1, i2;
    public TesteEventoAcao(){
        setTitle("Teste Evento Acao");
        setSize(240,100); //tamanho da janela
        getContentPane().setLayout(null); // anula o layout padrao
        //cor de fundo da janela no padrão RGB (Red Green Blue)
        getContentPane().setBackground(new Color(255,255,255));
        //provoca o termino da execução (encerra o programa)
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        centralizar();
    }
}
```



## Eventos de Ação - Implementação

```
i1 = i2 = 0;  
b1 = new JButton("Gravar");  
b1.setBounds(10, 10, 100, 30);  
b1.addActionListener(this);  
b2 = new JButton("Sair");  
b2.setBounds(120, 10, 100, 30);  
b2.addActionListener(this);  
l1 = new JLabel("Pressione os botões");  
l1.setBounds(5, 50, 220, 20);  
getContentPane().add(b1);  
getContentPane().add(b2);  
getContentPane().add(l1);
```

```
}
```



# Eventos de Ação - Implementação

```
public void actionPerformed(ActionEvent e){  
    if (e.getSource() == b1)  
        l1.setText("Botão gravar pressionado " + ++i1 + " vez(es)");  
    if (e.getSource() == b2)  
        l1.setText("Botão sair pressionado " + ++i2 + " vez(es)");  
}  
public static void main(String [] args){  
    TesteEventoAcao janela = new TesteEventoAcao();  
    janela.setVisible(true);  
}  
}
```



## Exercício

- Altere o programa anterior criando métodos para geração e tratamento dos components que julgar necessários.
- Caso o usuário clique no botão gravar o texto deve ficar em Vermelho e quando ele clicar no sair deve ficar em azul.



# Solução

```
package Exemplos.interfaceGrafica;
import java.awt.*; import java.awt.event.*; import javax.swing.*;
public class TesteEventoAcao extends JFrame
implements ActionListener{
    JLabel l1; JButton b1, b2;    int i1, i2;
    public TesteEventoAcao(){
        setTitle("Teste Evento Acao");          setSize(300,130); //tamanho da janela
        getContentPane().setLayout(null); // anula o layout padrao
        getContentPane().setBackground(new Color(255,255,255));
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);      setLocationRelativeTo(null);
        i1 = i2 = 0;
        b1 = criarBotao("Gravar", 10, 10, 100, 30);          b2 = criarBotao("Sair", 120, 10, 100, 30);
        l1 = new JLabel("Pressione os botões");              l1.setBounds(5, 50, 220, 20);
        getContentPane().add(l1);          setVisible(true);
    }
    private JButton criarBotao(String texto, int i, int j, int k, int l) {
        JButton botao = new JButton(texto);          botao.setBounds(i, j, k, l);          botao.addActionListener(this);
        add(botao);          return botao;
    }
    public void actionPerformed(ActionEvent e){
        if (e.getSource() == b1) {
            l1.setForeground(Color.BLUE);
            l1.setText("Botão gravar pressionado " + ++i1 + " vez(es)");
        }
        if (e.getSource() == b2) {
            l1.setForeground(Color.RED);
            l1.setText("Botão sair pressionado " + ++i2 + " vez(es)");
        }
    }
    public static void main(String [] args){
        new TesteEventoAcao();    }    }
```



# Exercícios

**1.** Implemente uma classe Java que gere uma tela de login de usuário com os seguintes componentes:

Um rótulo (label) com a descrição "Usuário"

Um rótulo (label) com a descrição "Senha"

Uma caixa de texto (TextField) para o usuário digitar o seu nome

Uma caixa de senha (JPasswordField) para o usuário digitar a sua senha

Dois botões: Um "Ok" e outro "Cancelar"

Quando o usuário clicar no botão "Cancelar" o programa deve ser encerrado (Utilize o comando `System.exit(0)` para isto).

Quando o usuário clicar no botão "Ok" valide o usuário e senha, e caso sejam informações corretas, utilize um `JOptionPane` para informar ao usuário que ele foi logado ao sistema e em seguida encerre o programa (Utilize o comando `System.exit(0)` para isto).

Caso as informações não estejam corretas, utilize um `JOptionPane` para informar ao usuário que as informações de login não estão corretas, e diga para o usuário digitar as informações novamente.





## Exercícios - Interface

Login do Usuário		—	□	×
<b>Usuário:</b>	<input type="text"/>			
<b>Senha:</b>	<input type="password"/>			
<input type="button" value="Ok"/>		<input type="button" value="Cancelar"/>		



# Exercício 1 - Solução

```
package Exemplos.interfaceGrafica.exercicios;
import java.awt.*;
import java.awt.event.ActionEvent;
import Exemplos.interfaceGrafica.TesteEventoAcao;
public class FormularioLogin extends JFrame implements ActionListener{
    JButton btnOk, btnCancel;
    JTextField campoUsuario;
    JPasswordField campoSenha;

    public FormularioLogin(){
        setTitle("Login do Usuário");
        GridLayout gl = new GridLayout(3,2,1,1);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JLabel labelUsuario = criarRotulo(" Usuário: ");
        getContentPane().add(campoUsuario);

        JLabel labelSenha = criarRotulo(" Senha: ");
        getContentPane().add(campoSenha);

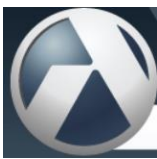
        btnOk = criarBotao("Ok");
        btnCancel = criarBotao("Cancelar");
        setVisible(true);
    }
    private JButton criarBotao(String texto) {
        JButton botao = new JButton(texto);
        botao.addActionListener(this);
        add(botao);
        return botao;
    }
    private JLabel criarRotulo(String texto) {
        JLabel rotulo = new JLabel(texto);
        rotulo.setForeground(Color.blue);
        rotulo.setFont(new Font("Times new Roman", Font.BOLD, 18));
        add(rotulo);
        return rotulo;
    }
    public void actionPerformed(ActionEvent e){
        if (e.getSource() == btnOk){
            String usu = campoUsuario.getText();
            String senha = String.valueOf(campoSenha.getPassword());
            if (usu.equalsIgnoreCase("Marcos") && senha.equals("2023")){
                JOptionPane.showMessageDialog(null, usu.toUpperCase() + ", você foi logado ao Sistema.");
                this.setVisible(false);
                TesteEventoAcao te = new TesteEventoAcao();
            }
        }
        else{
            JOptionPane.showMessageDialog(null, "Usuário ou senha inválidos.\nTente novamente");
        }
        if (e.getSource() == btnCancel){
            System.exit(0);
        }
    }
    public static void main(String[] args) {
        new FormularioLogin();
    }
}
```



## Exercícios

2. Implemente uma classe em Java que gere a seguinte GUI. Ao clicar no Botão OK dê a mensagem do cadastro com sucesso e limpe os campos da tela. Ao clicar no botão Cancelar confirme se deseja realmente e, em caso positivo, encerre o programa.

The image shows a Java Swing window titled "Formulário". The window has a standard title bar with a minimize button, a maximize button (disabled), and a close button. The main content area has a light gray background. On the left, there are five labels in red text: "Nome:", "Endereco:", "Telefone:", "CPF:", and "RG:". To the right of each label is a white text input field. At the bottom of the window, there are two buttons: "Ok" and "Cancelar".



## Exercício 2 - Solução

```
package Exemplos.interfaceGrafica.exercicios;
import java.awt.*; import java.awt.event.ActionEvent; import java.awt.event.ActionListener; import javax.swing.*;
public class FormularioCadastro extends JFrame implements ActionListener {
    JButton btnOk, btnCancel; JTextField txtNome, txtEndereco, txtTelefone, txtCPF, txtRG;
    public FormularioCadastro(){
        setTitle("Formulário"); setSize(350,200); GridLAYOUT gl = new GridLayout(6,2,1,3);
        getContentPane().setLayout(gl); //provoca o termino da execucao (encerra o programa)
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); setLocationRelativeTo(null);

        JLabel labelNome = criarRotulo(" Nome: "); txtNome = new JTextField(); getContentPane().add(txtNome);
        JLabel labelEndereco = criarRotulo(" Endereco: "); txtEndereco = new JTextField(); getContentPane().add(txtEndereco);
        JLabel labelTelefone = criarRotulo(" Telefone: "); txtTelefone = new JTextField(); getContentPane().add(txtTelefone);
        JLabel labelCPF = criarRotulo(" CPF: "); txtCPF = new JTextField(); getContentPane().add(txtCPF);
        JLabel labelRG = criarRotulo(" RG: "); txtRG = new JTextField(); getContentPane().add(txtRG);

        btnOk = new JButton("Ok"); btnOk.addActionListener(this); getContentPane().add(btnOk);
        btnCancel = new JButton("Cancelar"); btnCancel.addActionListener(this); getContentPane().add(btnCancelar);
        setVisible(true);
    }
    public JLabel criarRotulo(String texto) {
        JLabel rotulo = new JLabel(texto); rotulo.setFont(new Font("Times new Roman", Font.BOLD, 18));
        rotulo.setForeground(Color.blue); add(rotulo); return rotulo;
    }
    public void actionPerformed(ActionEvent e){
        if (e.getSource() == btnOk){
            JOptionPane.showMessageDialog(null, "Cadastro realizado com sucesso!"); limparCampos();
        }
        else if (e.getSource() == btnCancel){
            System.exit(0);
        }
    }
    private void limparCampos() {
        txtNome.setText(""); txtEndereco.setText(""); txtTelefone.setText(""); txtCPF.setText("");
        txtRG.setText("");
    }
    public static void main(String[] args) {
        new FormularioCadastro();
    }
}
```



# Painéis

**Vejam a saída gerada por este exemplo que implementa vários painéis e Layouts diferentes.**

Uso de botoes de Radio

Digite um valor

☐ 10% do valor ☐ 20% do valor ☐ 30% do valor

% do Valor :



# Barras de Rolagem

- A barra de rolagem é o objeto que permite mover o conteúdo interno de uma janela para cima ou para baixo, para direita ou para esquerda. A classe que permite a utilização da Barra de Rolagem é a **JScrollbar**. O método abaixo é utilizado para criação de uma barra de rolagem.

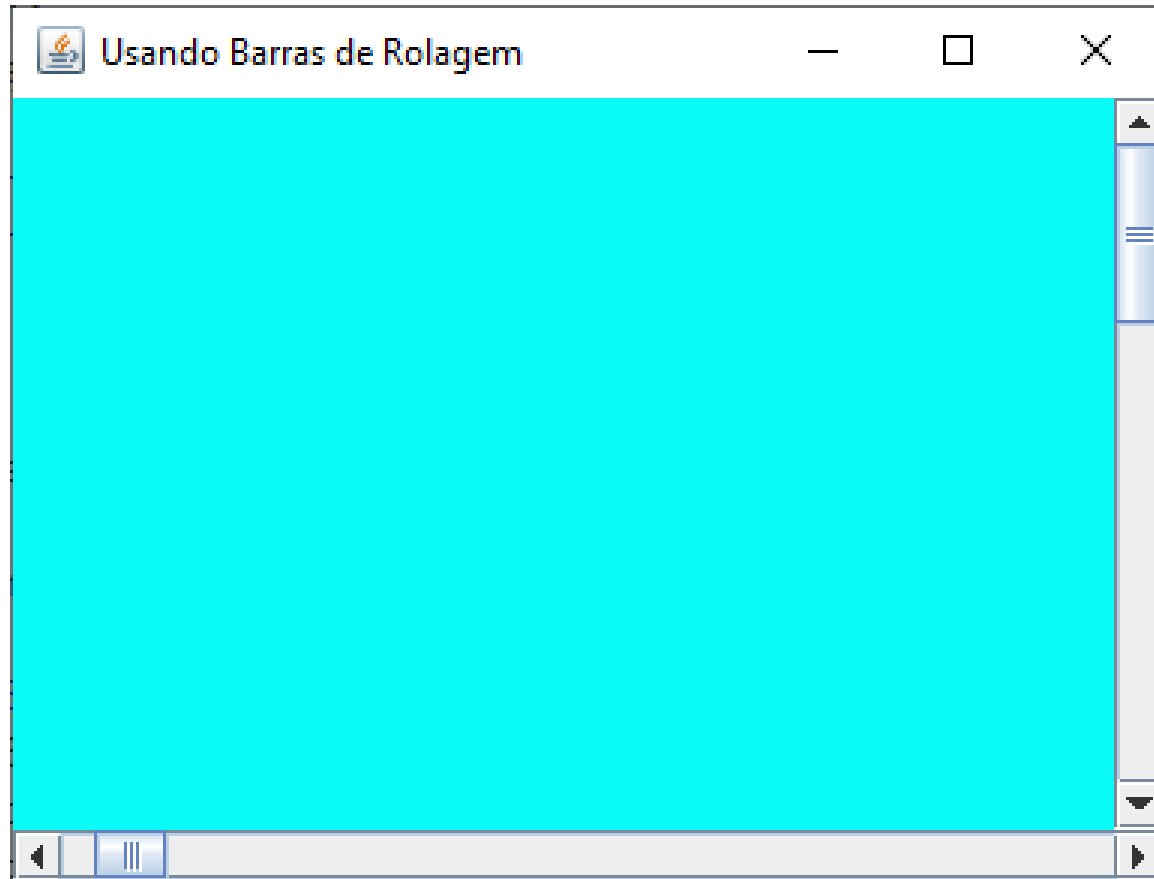
**JScrollbar** (int orientação, int value, int interval, int minimum, int maximum). Onde:

- ☐ **orientação** é um valor que define se a barra será horizontal(0) ou vertical(1), podemos definir também através do método JScrollbar.VERTICAL ou JScrollbar.HORIZONTAL.
- ☐ **Value** define a posição inicial da barra de rolagem que deve estar entre o minimum e maximum definidos;
- ☐ **Interval** define o incremento ou decremento quando o usuário clica no meio da barra e não nas setas;
- ☐ **minimum** e **maximum** definem o valor mínimo e máximo que a barra de rolagem pode atingir.



# Barra de Rolagem

**Vejamos um exemplo do uso da Barra de Rolagem.**





# Áreas de Texto

- Uma área de texto é semelhante a um campo texto com a diferença que a área possui várias linhas, formando assim algo parecido como um documento do Word. É interessante por exemplo para criarmos formulários de ajuda. Veremos a seguir um exemplo do uso de áreas de texto que podem ser criadas com base na classe **JTextArea**.





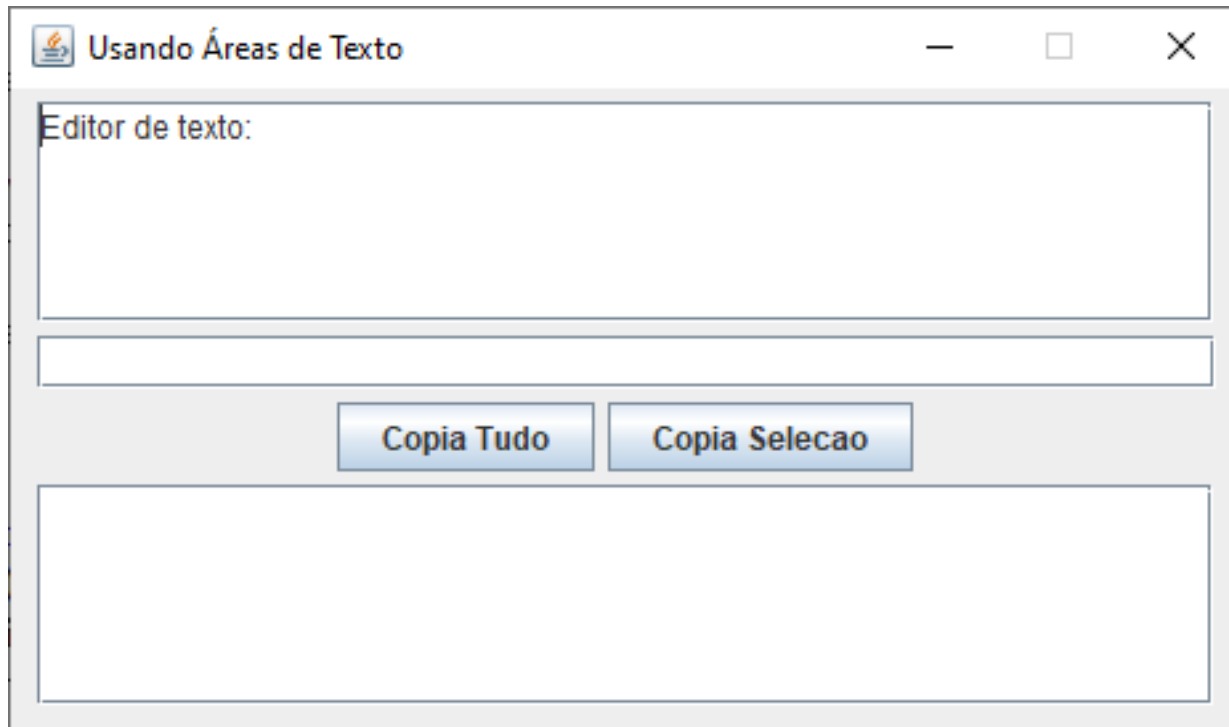
# Áreas de Texto

- Observe na tabela abaixo os métodos utilizados no programa para trabalharmos com Áreas de Texto.
  - ❑ **JTextArea()** Cria uma Área de texto
  - ❑ **JTextArea(int linhas, int colunas)** Cria uma Área de texto com linhas e colunas passadas entre os parênteses
  - ❑ **JTextArea(String Texto)** Cria uma Área de texto com um texto já escrito
  - ❑ **JTextArea(String Texto, int,int)** Cria uma Área de texto indicado entre os parênteses bem como as linhas e colunas determinadas
  - ❑ **getSelectedText()** Retorna com o texto que estiver selecionado
  - ❑ **Insert(String, int)** Insere a String na posição indicada



# Áreas de Texto

Exemplo





# Menu

- Um menu suspenso é algo comum hoje em aplicativos, vemos isto no Word, Excel, Internet, Eclipse, etc, enfim é algo já sedimentado no conhecimento dos usuários de computador.
- Veremos a seguir os passos necessários para criação de um Menu:
  - ❑ Crie um objeto da classe JMenuBar ;
  - ❑ Defina este objeto criado como Barra de menu utilizando para isto o método setJMenuBar();

**Exemplo:** `JMenuBar BarraMenu = new JMenuBar();`  
`setJMenuBar(BarraMenu);`

❑ Crie as opções do Menu através da classe JMenu;

**Exemplo:** JMenu Arquivo = new JMenu ();  
JMenu Editar = new JMenu ();

❑ Adicione os menus a barra de menus através do método `add`.

**Exemplo:** BarraMenu.add(Arquivo);  
BarraMenu.add(Editar);



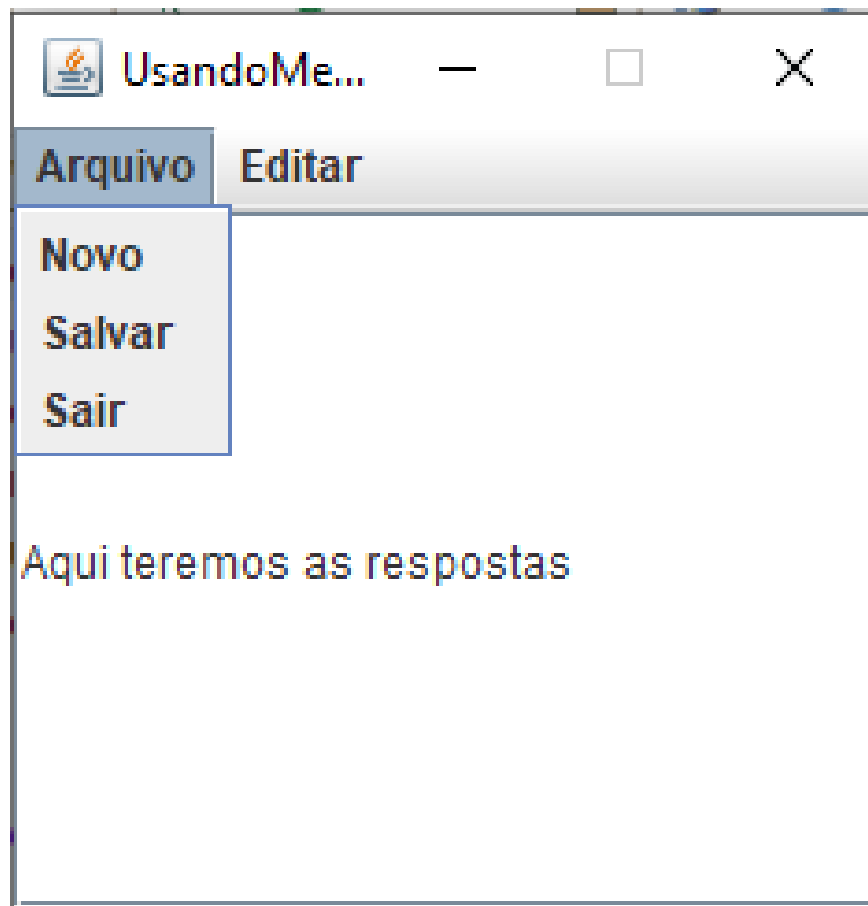
# Menus

- Crie e Adicione os itens do menu utilizando a classe JMenuItem.
  - ❑ **Exemplo:** JMenuItem Novo = new JMenuItem ();  
JMenuItem Salvar = new JMenuItem ();  
Arquivo.add(Novo);  
Arquivo.add(Salvar);
- Pronto é só repetir os passos acima para criar os menus e itens de menu que desejar na barra já criada.



# Menus

## Exemplo:





# Menus

- Observe na tabela abaixo os métodos utilizados no programa para trabalharmos com Menus.
- JMenuBar() Cria uma Barra de Menu que receberá os menus.
- JMenu(String) Cria um menu com texto já definido, que receberá os itens dentro dele.
- JMenuItem(String) Cria uma item de menu com texto definido.
- ObjMenu.add(ObjItemMenu) Adiciona os itens ao objeto de menu correspondente em nosso caso no Menu Arquivo.
- ObjMenuBar.add(ObjMenu) Adiciona o Menu à Barra de Menus.
- SetJMenuBar(ObjMenuBar) Carrega a barra de menus no formulário.



# Eventos

- Eventos são ações que poderão ocorrer sobre nossos objetos como clique do mouse, o clique sobre uma barra de rolagem, o foco sobre um botão, a seleção de um texto, etc.
- Para tratar os eventos que podem ser gerados o Java disponibiliza uma série de classes que chamamos de Receptoras de Eventos.
- É importante lembrar que quando um objeto precisa ter o seu evento tratado ele deve ser registrado na classe que irá tratar o evento.
- **Handler** é a forma como um tratamento de evento é conhecido.
- A seguir vemos uma relação de eventos, o método que o chama e em quais objetos estes eventos podem ser associados.



# Eventos

Evento	Método	Objetos
ActionListener	addActionListener()	JButton, JCheckBox, JComboBox, JTextField e JRadioButton
AdjustementListener	addAdjustementListener ()	JScroolBar
FocusListener	addFocusListener()	Todos
ItemListener	addItemListener()	JButton, JCheckBox, JComboBox, e JRadioButton
KeyListener	addKeyListener()	TextField
MouseListener	addMouseListener()	Mouse
MouseMotionListener	addMouseMotionListener()	Mouse
WindowListener	addWindowListener()	JWindow e JFrame
ComponentListener	addComponentListener()	Todos





# Eventos

- Cabe ressaltar que toda vez que necessitamos tratar os eventos ocorridos fazemos uso das **Interfaces** que contém este tratamento e **obrigatoriamente** temos de implementá-los.
- O Exemplo a seguir ilustra o tratamento de eventos do Mouse.

```
public class ExMouse extends JFrame implements MouseMotionListener,  
MouseListener
```

```
{  
    JLabel mensagem;  
    ExMouse() {  
        super( "Eventos do Mouse " );  
        mensagem = new JLabel();  
        getContentPane().add( mensagem );  
        addMouseListener( this );  
        addMouseMotionListener( this );  
        setSize( 300, 150 );  
        setVisible(true);  
    }  
}
```



# Eventos

```
public void mouseClicked( MouseEvent e )
    {
        mensagem.setText( "Clicado em "+ coords(e) ); }
public void mousePressed( MouseEvent e )
{
    mensagem.setText( "Pressionado em "+ coords(e) ); }
public void mouseReleased( MouseEvent e )
{
    mensagem.setText( "Liberado em "+ coords(e) ); }
public void mouseExited( MouseEvent e )
{
    mensagem.setText( "Mouse saiu da janela" ); }
public void mouseEntered( MouseEvent e )
    {
        mensagem.setText( "Mouse entrou na janela" ); }
public void mouseDragged( MouseEvent e )
{
    mensagem.setText( "Arrastado em "+ coords(e) ); }
public void mouseMoved( MouseEvent e )
{
    mensagem.setText( "Movido em " + coords(e) ); }
```



## Eventos

```
public String coords(MouseEvent e)
{
    return e.getX() + ", " + e.getY() ; }
}
```

```
public static void main( String args[] )
{
    ExMouse(); }
}
```



## Eventos

- O exemplo a seguir foi extraído do **Livro Java Ensino Didático** e demonstra o uso de diversos eventos.

```
import java.awt.*;      import java.awt.event.*;
import javax.swing.*;   import javax.swing.event.*;
public class ExEventos extends JFrame implements MouseListener,
KeyListener, TextListener, FocusListener, MouseMotionListener
{ JButton B1;  JLabel L1,L2;  JTextField T1,T2;  TextField T3;
  static int E=100,T=100;
  public static void main(String[] args)
  {new ExEventos();
  public void windowActivated(WindowEvent e)
  {      // Escreva mensagem A Janela foi ativada  }
```



# Eventos

```
public void windowDeactivated(WindowEvent e)
    {    // A Janela foi desativada    }
public void windowIconified(WindowEvent e)
    {    // A Janela foi minimizada    }
public void windowDeiconified(WindowEvent e)
    {    // A Janela foi restaurada;    }
public void windowOpened(WindowEvent e)
    {    mostraMensagem("A Janela foi aberta");    }
public void windowClosed(WindowEvent e)
    {    // A Janela foi fechada    }    };
Janela.addWindowListener(x);
ComponentListener y = new ComponentAdapter(){
public void componentHidden(ComponentEvent e)
{    mostraMensagem("A janela tornou-se oculta");    }
public void componentMoved(ComponentEvent e)
{ mostraMensagem("A janela foi movida");    }
```



# Eventos

```
public void componentResized(ComponentEvent e)
{
    mostraMensagem("A janela foi redimensionada");
}
public void componentShown(ComponentEvent e)
{
    mostraMensagem("A janela tornou-se visível");
}
Janela.addComponentListener(y); }
```

```
ExEventos() {
    setTitle("Manipulação de Eventos");
    setSize(280,200);
    setLocation(E,T);
    getContentPane().setLayout(new GridLayout(6,1));
    L1 = new JLabel(""); L2 = new JLabel("");
    B1 = new JButton ("Eventos do Botão");
    T1 = new JTextField(); T2 = new JTextField(); T3 = new TextField();
    B1.addMouseListener(this);
    B1.addMouseMotionListener(this);
    B1.setBackground(Color.gray);
}
```



# Eventos

```
T1.addKeyListener(this);
T3.addTextListener(this);
T3.addKeyListener(this);
T3.addFocusListener(this);
getContentPane().add(B1); getContentPane().add(L1);
getContentPane().add(L2);  getContentPane().add(T1);
getContentPane().add(T3); getContentPane().add(T2);
}
// métodos relativos a MouseListener
public void mousePressed(MouseEvent e)
{ L1.setText("O botão do Mouse foi pressionado");
  mostraMensagem(""+e.getClickCount() + " "
    + e.getX() + " " + e.getY());
  mostraMensagem(""+e.isLeftMouseButton() + " "+ e.isControlDown());
}
public void mouseClicked(MouseEvent e)
{ L1.setText("O botão do Mouse foi solto"); }
```



## Eventos

```
public void mouseEntered(MouseEvent e)
{ // o ponteiro do mouse entrou na área
  B1.setBackground(Color.yellow); }
public void mouseExited(MouseEvent e)
{ // o ponteiro do mouse saiu da área
  B1.setBackground(Color.gray); }
public void mouseReleased(MouseEvent e)
{ L1.setText("O ponteiro do Mouse foi arrastado"); }
// métodos relativos a MouseMotionListener
public void mouseMoved(MouseEvent e)
{ L2.setText("Mouse se moveu em "+e.getX()+" ,
"+e.getY());}
public void mouseDragged(MouseEvent e)
{ L2.setText("Mouse foi arrastado em "+e.getX()+" ,
"+e.getY());}
```





# Eventos

// métodos relativos a **KeyListener**

```
public void keyPressed(KeyEvent e) { // uma tecla foi pressionada
    if (e.getSource()==T1)
        {   if(e.getKeyCode()==9)
            { e.setKeyCode(9);
              return; }
            return;    } }
```

```
public void keyReleased(KeyEvent e)
    {   // uma tecla foi solta    }
```

```
public void keyTyped(KeyEvent e)
    {   // uma tecla Unicode foi pressionada    }
```

// método relativo a **TextListener**



# Eventos

```
public void textValueChanged(TextEvent e)    { // o conteúdo alterado
    int n1,n2;
    try
        {n1 = Integer.parseInt(T1.getText());
         n2 = Integer.parseInt(T3.getText());
         T2.setText(""+n1*n2);    }
    catch(NumberFormatException erro)
        { T2.setText("0");
          return; }    }

public void focusGained(FocusEvent e) { // o objeto recebeu o foco
    L1.setText("O objeto T3 recebeu o foco");    }

public void focusLost(FocusEvent e)
    { // o objeto perdeu o foco
      L1.setText("O objeto T3 perdeu o foco");    }

public static void mostraMensagem(String men)    {
    JOptionPane.showMessageDialog(null,men,"Mensagem",
    JOptionPane.INFORMATION_MESSAGE); }    }
```



## Exercício

- Altere o exemplo para que seja tratada a alteração nos campos de texto e seja exibida qual Tecla foi pressionada.
- Altere o tratamento para informar que um valor não foi fornecido corretamente e por isto o resultado será 0.
- Dê uma mensagem de até breve Quando o usuário clicar em Cancelar.



## Exercício

- Abra a sua tela de login e trate os eventos necessários para:
  - ☐ Exibir uma mensagem quando a tela for:
    - ☐ aberta,
    - ☐ Movimentada,
    - ☐ Redimensionada,
    - ☐ Estiver sendo fechada
  - ☐ Exibir uma mensagem quando o campo login receber ou perder o foco.
  - ☐ Exibir uma mensagem quando o no senha for digitado algo:
  - ☐ Exibir uma mensagem quando o botão OK for pressionado e quando o botão Cancelar receber o foco.
  - ☐ Por fim crie uma mensagem que mostre a localização do mouse na tela.



# Exercícios

- Crie a tela de cadastro de Aluno a fim de que ao clicar no botão Gravar seja exibida uma Caixa de Diálogo informando que os dados foram cadastrados e exiba no campo Média a media simples das notas. Ao clicar no botão Sair confirme se deseja encerrar a aplicação e, em caso positivo, encerre.
- Altere o Código necessário sobre as caixas de texto txtID e txtMedia de forma que ao perder o foco do campo txtID o botão Sair seja habilitado e, no caso do campo txtMedia, quando o campo txtNota3 perder o foco ele deve ler o valor das 3 caixas de texto e calcular a media inserindo-a no campo txtMedia.

Cadastro de Alunos

Código  Nome  Telefone

1ª Nota  2ª Nota  3ª Nota  Média



## Exercício

- Habilite o botão **sair**.
- Desabilite o botão **gravar** a fim de que ele somente seja habilitado quando todos os campos estiverem preenchidos. Caso algum campo esteja vazio informe ao usuário que todos os campos são de preenchimento obrigatório.
- Implemente o Código necessário sobre o botão gravar de forma que, ao clicar no botão gravar, os dados da tela sejam gravados na nossa tabela.