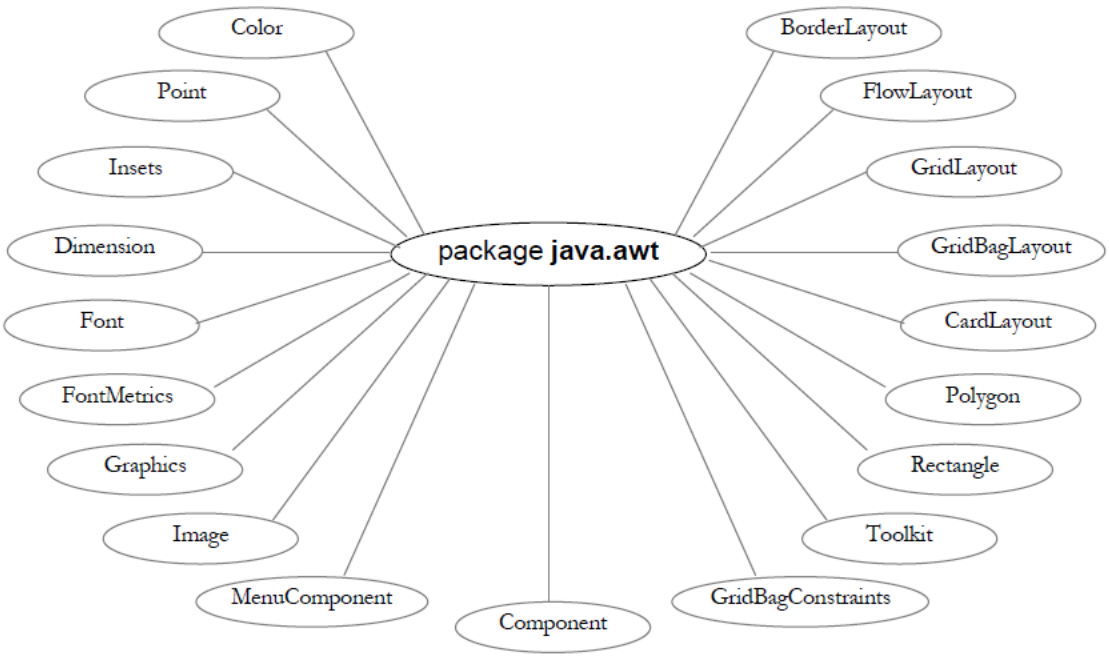
O Java tem dois pacotes que permitem criar interfaces gráficas: **java.awt**, que foi o primeiro a surgir, e **javax.swing**, que o veio "substituir". Todas as classes do pacote javax.swing são extensões das classes do pacote java.awt.

Os objetos das classes que constituem o pacote javax.swing denominam-se **componentes**. Todas as interfaces gráficas se constroem a partir de componentes denominados contentores (janelas e painéis) que, por sua vez, podem conter outros componentes (botões, rótulos, caixas de texto, entre outros).
Os componentes devem ser dispostos de modo a tornar a interface gráfica o mais intuitiva possível e, para isso, o Java disponibiliza vários gestores de layout que se encontram no pacote java.awt.

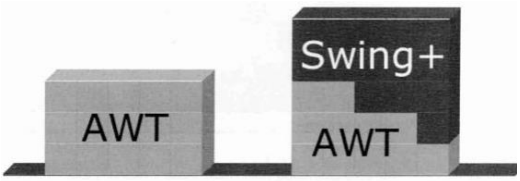
Componentes gráficos são classes que contém membros associados a informações visuais, como cores, dimensões e bordas.
A interface java.AWT usa os componentes de interface gráfica fornecidos pelo sistema operativo subjacente (por exemplo, botões, caixas de texto, menus, etc.), enquanto que o javax.Swing usa seus próprios componentes de interface gráfica que são desenhados em Java.

Aspetos de ergonomia de software são importantes, tais como combinação de cores e fontes utilizadas na interface gráfica. As principais classes do pacote AWT estão representadas na figura abaixo:

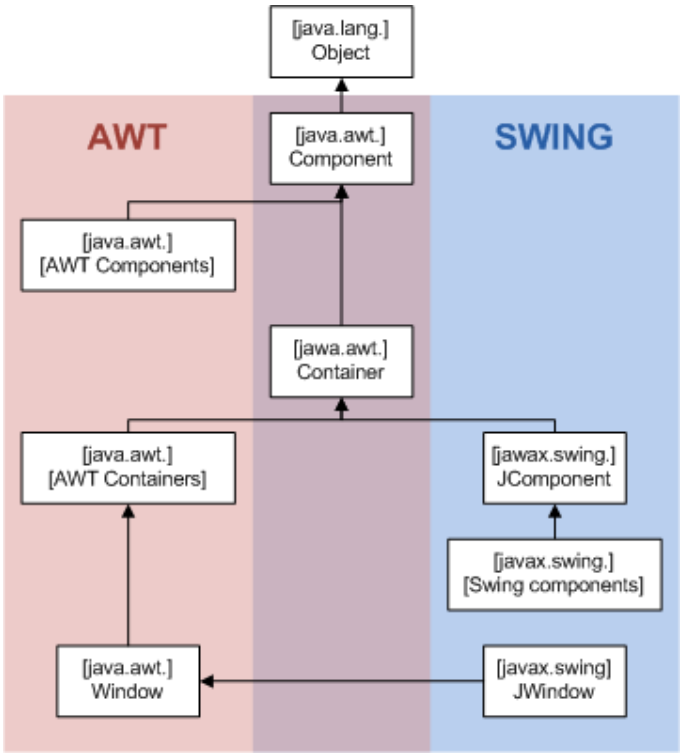
Componentes gráficos – o pacote AWT



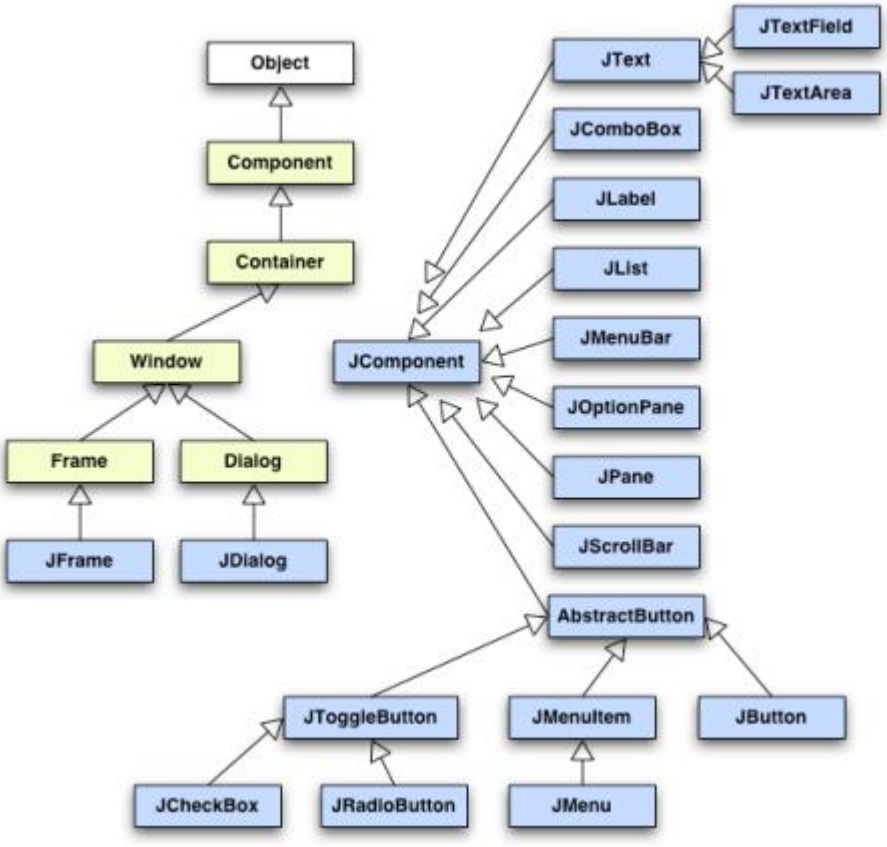
A classe swing do pacote javax oferece mais potencialidades e flexibilidade aos componentes GUI do que a classe awt.



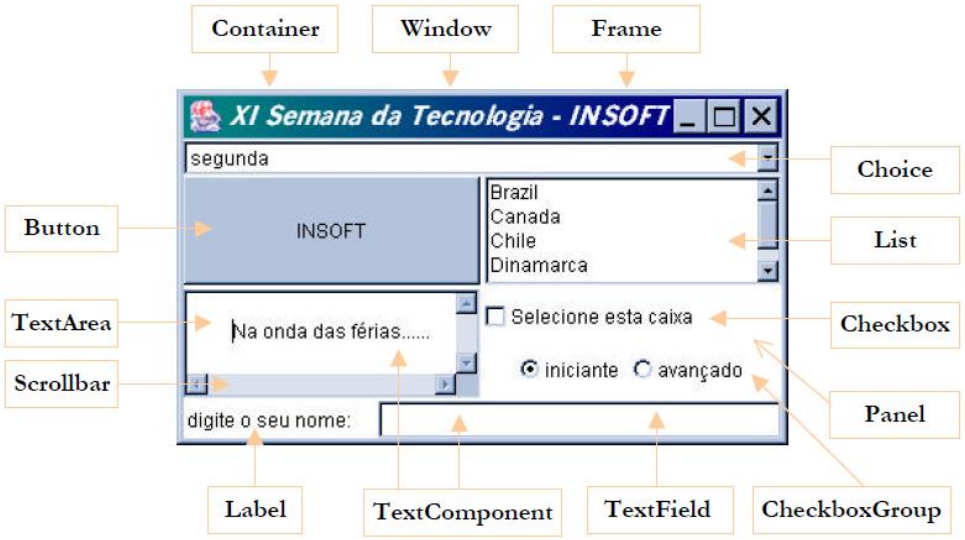
Os componentes de Swing são totalmente personalizáveis e oferecem mais recursos do que os componentes AWT.
Em geral, Swing é considerada uma biblioteca de GUI mais poderosa e flexível em comparação com AWT, mas é também mais complexa e requer mais recursos do sistema.



Componentes gráficos – o pacote Swing

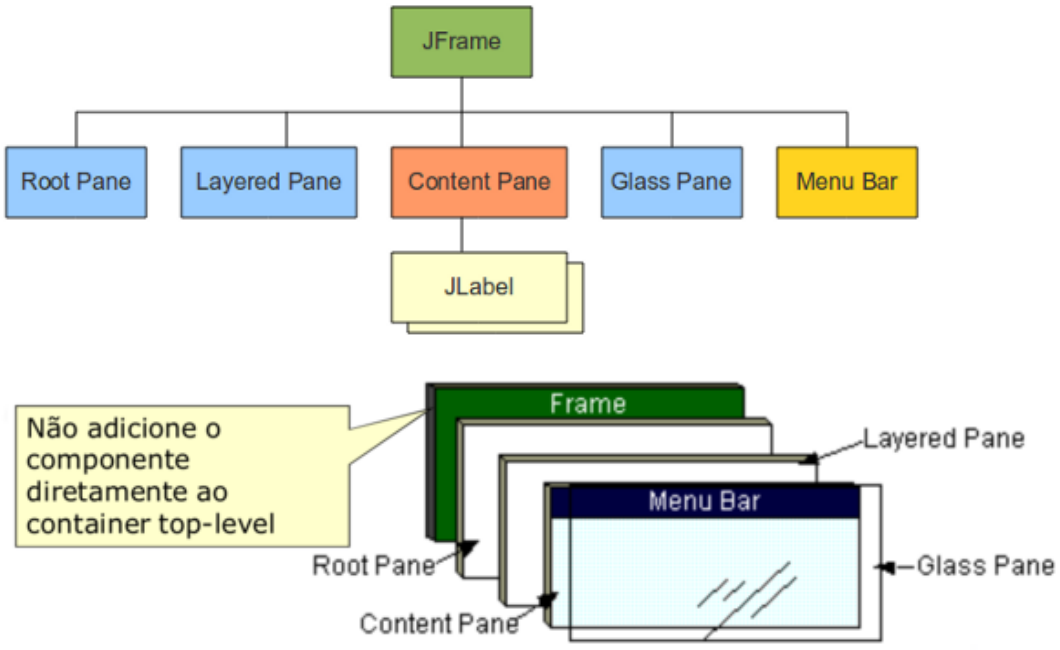


Os detalhes do uso de cada um desses componentes serão apresentados em aula a partir do exemplo abaixo.



AJUDA: Help - Eclipse Platform

Contentores e Componentes



Contentores de nível superior → Usa-se, geralmente, como base, isto é, proporciona um lugar para utilização de outros elementos. Exemplos JFrame, JDialog e JApplet

Contentores de nível intermédio → Usa-se Para agrupar/dispor contentores atómicos Exemplos JPanel, JScrollPane e JTablePane.

Contentores de nível atómico → Não se usa para armazenar outros elementos. É uma entidade autossuficiente, servindo para apresentar informação ao utilizador e recebê-la deste. Exemplos: JButton, JLabel, JComboBox, JTextField e JTable.

AplicationWindow – Corresponde à janela principal de uma aplicação, a partir da qual acedemos a outras janelas.

JFrame

JFrame é um contentor que consiste numa janela à qual se podem adicionar outros componentes ou contentores (figura 2):

Figura 2 – JFrame

O quadro seguinte apresenta os construtores mais usados da classe "JFrame":

Construtor	Descrição
JFrame()	Constrói uma janela sem título.
JFrame(String titulo)	Constrói uma janela com o título indicado no parâmetro.

Exemplo:

```
import java.awt.EventQueue;

import javax.swing.JFrame;

public class Exemplo1 extends JFrame{

private JFrame frame;

/**
* Launch the application.
*/
public static void main(String[] args) {
	EventQueue.invokeLater(new Runnable() {
	public void run() {
	try {
		Exemplo1 window = new Exemplo1();
		window.frame.setVisible(true);
	}
	catch (Exception e) {
e.printStackTrace();
}
}
});
}
```

Importação das classes necessárias

Criação de um objeto da nossa interface G.

Tornar a janela visível

NOTA: Para alterar o tamanho da janela podemos usar o método setSize. Exemplo:
frame.setSize(300, 200);

```
/**
* Initialize the contents of the frame (constructor).
*/

private void initialize() {

frame = new JFrame();

//define que quando a janela é fechada (x100, y100)
//// e o tamanho da janela (450 x 300)

frame.setBounds(100, 100, 450, 300);

//define que quando a janela é fechada a aplicação também é fechada

frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

//atribuir nome à janela

frame.setTitle("Janela");

}
```

JPanel

JPanel é um contentor que consiste num painel ao qual se podem adicionar outros componentes (figura 3). Este, por sua vez, é adicionado a uma janela (JFrame):

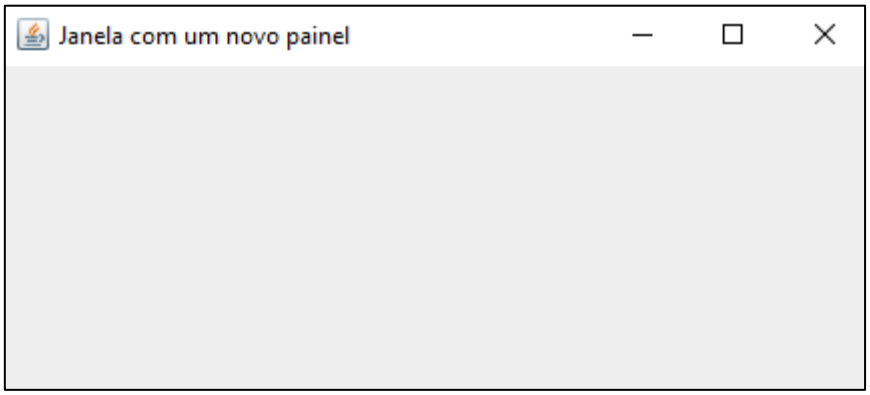


Figura 3 – JPanel

O quadro seguinte apresenta os construtores mais usados da classe "JPanel":

Construtor	Descrição
JPanel()	Constrói um painel.
JPanel(LayoutManager layout)	Constrói um painel associado a um determinado gestor de layout.

Um gestor de layout determina a forma como os componentes são dispostos na janela (JFrame) ou no painel (JPanel).

Exemplo:

```
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;


public class UsarPainel extends JFrame {

    private JPanel contentPane;

    /**
     * Launch the application.
     */
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    UsarPainel janela = new UsarPainel();
                    janela.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    /**
     * Create the frame.
     */
    public UsarPainel() {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 450, 300);
        contentPane = new JPanel();
        setContentPane(contentPane);
    }
```

JLabel

Componente que consta de um rótulo de texto (figura 4). Pode ser adicionado a um contentor (JFrame ou JPanel):

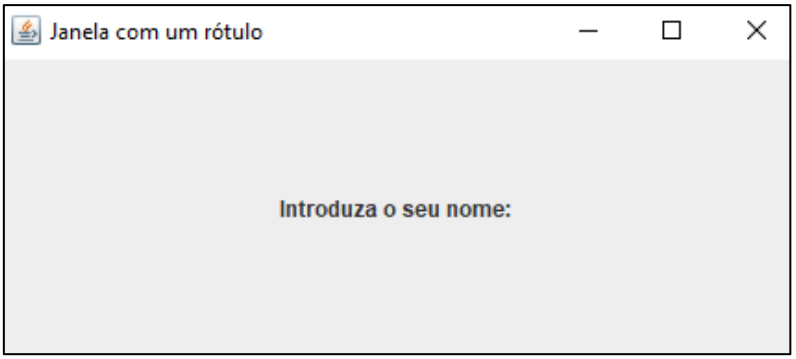


Figura 4 – JLabel

O quadro seguinte apresenta os construtores da classe “JLabel”:

Construtor	Descrição
JLabel()	Constrói um rótulo sem texto.
JLabel(String texto)	Constrói um rótulo com o texto indicado no parâmetro.
JLabel(Icon icone)	Constrói um rótulo com a imagem indicada no parâmetro.
JLabel(String texto, int alinhaHoriz*)	Constrói um rótulo com o texto e o alinhamento do mesmo, indicados nos parâmetros.
JLabel(Icon icone, int alinhaHoriz*)	Constrói um rótulo com a imagem e o alinhamento da mesma, indicados nos parâmetros.
JLabel(String texto, Icon icone, int alinhaHoriz*)	Constrói um rótulo com o texto, a imagem e o alinhamento dos mesmos, indicados nos parâmetros.

O parâmetro “alinhaHoriz” poderá ter um dos valores definidos nas constantes do tipo inteiro: CENTTER (0), LEFT (2), RIGHT (4), LEADING (10) e TRAILING (11). Estas constantes estão definidas na interface SwingConstants.

Exemplo1:

```
package UsarRotulo;

import java.awt.Color;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JLabel;

public class UsarRotulo {

	public static void main(String[] args) {
		JFrame janela=new JFrame();
		janela.setTitle("Janela com um rótulo");
		//tamanho da janela
		janela.setSize(450,400);
		//coordenadas do ponto onde a janela começa a ser desenhada
		janela.setLocation(100, 100);
		//define que quando a janela é fechada a aplicação também é fechada
		janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		JPanel painel=new JPanel();
		painel.setBounds(0,0,450,200); // Define o tamanho do painel.
		painel.setBackground(new Color(255,255,255));
		janela.add(painel);
		JLabel rotulo=new JLabel("Introduza o seu nome: ");
		painel.add(rotulo);
		janela.setVisible(true);
	}
}
```

Exemplo2:

```
package UsarRotulo;
	import javax.swing.JFrame;
	import javax.swing.JLabel;
public class UsarRotulo {

	public static void main(String[] args) {
		JFrame janela=new JFrame();
		janela.setTitle("Janela com um rótulo");
		//tamanho da janela
		janela.setSize(450,400);
		//coordenadas do ponto onde a janela começa a ser desenhada
		janela.setLocation(100, 100);
		//define que quando a janela é fechada a aplicação também é fechada
		janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		JLabel rotulo=new JLabel("Introduza o seu nome: ", 0);
		janela.add(rotulo);
		janela.setVisible(true);
	}
}
```

JTextField

Componente que consta de um campo de texto editável com uma única linha (figura 5). Utiliza-se em conjunto com um rótulo (JLabel). Pode ser adicionado a um contentor (JFrame ou JPanel):

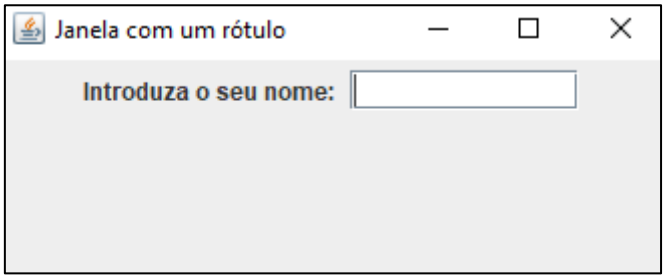


Figura 5 – JTextField

O quadro seguinte apresenta os construtores da classe "JTextField":

Construtor	Descrição
JTextField()	Constrói um campo de texto.
JTextField(int colunas)	Constrói um campo de texto com o número de colunas indicado no parâmetro.
JTextField(String texto)	Constrói um campo de texto que contém, como valor predefinido, o texto indicado no parâmetro.
JTextField(String texto, int colunas)	Constrói um campo de texto que contém, como valor predefinido, o texto indicado no parâmetro. A largura do campo é definida pelo número de colunas também indicadas no parâmetro.

Exemplo:

```
package UsarCaixaTexto;

import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JTextField;
import java.awt.FlowLayout;

public class UsarCaixaTexto {
	public static void main(String[] args) {
		JFrame janela=new JFrame();
		janela.setTitle("Janela com uma caixa de texto");
		//tamanho da janela
		janela.setSize(350,150);
		//coordenadas do ponto onde a janela começa a ser desenhada
		janela.setLocation(50, 50);
		//define que quando a janela é fechada a aplicação também é fechada
		janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		janela.setLayout(new FlowLayout());
		JLabel rotulo=new JLabel("Introduza o seu nome: ", 0);
		janela.add(rotulo);
		JTextField CaixaTexto= new JTextField(10);
		janela.add(CaixaTexto);
		janela.setVisible(true);
	}
}
```

JTextArea

Componente que consta de um campo de texto editável com diversas linhas (figura 6). Utiliza-se em conjunto com um JScrollPane para que as barras de rolamento direita e inferir apareçam quando necessário. Pode ser adicionado a um contentor (JFrame ou JPanel):

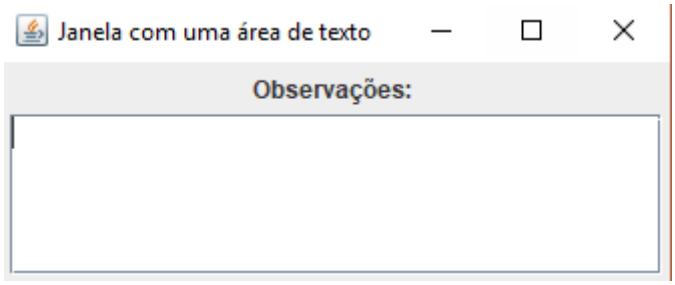


Figura 6 – JTextArea

O quadro seguinte apresenta os construtores da classe "JTextArea":

Construtor	Descrição
JTextArea()	Constrói uma área de texto com dimensão 0.
JTextArea(String texto)	Constrói uma área de texto que contém, como valor por predefinição, o texto indicado no parâmetro.
JTextArea(int linhas, int colunas)	Constrói uma área de texto com o número de linhas e de colunas indicados nos parâmetros.
JTextArea(String texto, int linhas, int colunas)	Constrói uma área de texto que contém, como valor por predefinição, o texto indicado no parâmetro. As dimensões da área são definidas pelo número de linhas e de colunas indicadas nos parâmetros.

Exemplo:

```
package AreaTexto;
import javax.swing.JFrame;
import javax.swing.JTextArea;
import javax.swing.JScrollPane;
import java.awt.Dimension;
import javax.swing.JLabel;
import java.awt.FlowLayout;

public class AreaTexto {

	public static void main(String[] args) {
		JFrame janela=new JFrame();
		janela.setTitle("Janela com uma área de texto");
		//tamanho da janela
		janela.setSize(350,150);
		//coordenadas do ponto onde a janela começa a ser desenhada
		janela.setLocation(50, 50);
		//define que quando a janela é fechada a aplicação também é fechada
		janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		janela.setLayout(new FlowLayout());
		JLabel rotulo=new JLabel("Observações: ");
		JTextArea CaixaTexto= new JTextArea();
		/*Se o texto atingir a margem direita é quebrado e passa para a linha
		seguinte*/
		CaixaTexto.setWrapStyleWord(true);
		/*Se o texto for maior do que a área de texto aparecerão as barras de
		rolagem. Neste caso,só aparecerá a barra vertical uma vez que se utilizou
		o método setLineWrap(true)*/
		JScrollPane areaRolamento=new JScrollPane(CaixaTexto);
		areaRolamento.setPreferredSize(new Dimension(325,80));

		janela.add(rotulo);
		janela.add(areaRolamento);
		janela.setVisible(true);
}}
```

JButton

Componente que consiste num botão em forma retangular (figura 7). Pode ser adicionado a um contentor (JFrame ou JPanel):

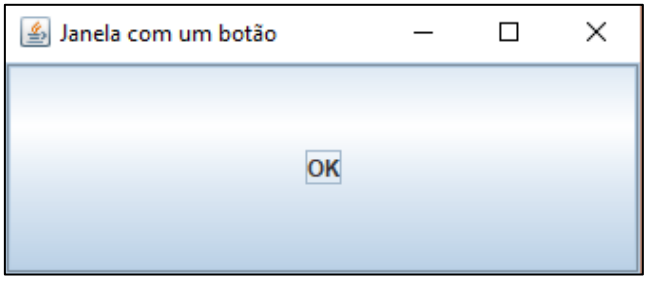


Figura 7 – JButton

O quadro seguinte apresenta os construtores da classe "JButton":

Construtor	Descrição
JButton ()	Constrói um botão sem texto e sem imagem.
JButton (String texto)	Constrói um botão com o texto indicado no parâmetro.
JButton(String texto, Icon icone)	Constrói um botão com o texto e imagem indicados nos parâmetros.

JButton(Icon icone)	Constrói um botão com a imagem indicada nos parâmetros.

Exemplo1:

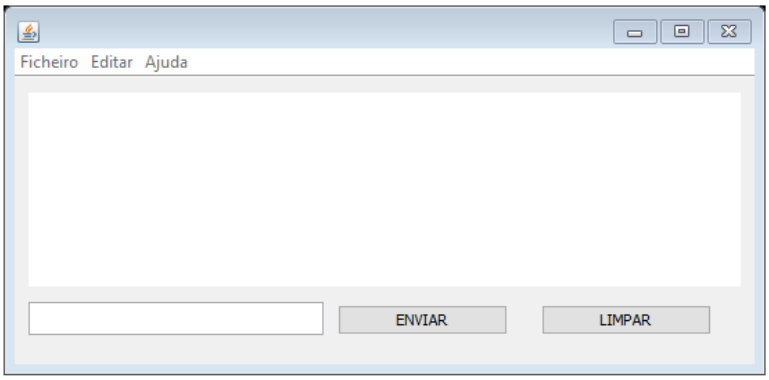
```
package Botao;

import javax.swing.JFrame;
import javax.swing.JButton;

public class Botao {

	public static void main(String[] args) {
		JFrame janela=new JFrame();
		janela.setTitle("Janela com um botão");
		//tamanho da janela
		janela.setSize(350,150);
		//coordenadas do ponto onde a janela começa a ser desenhada
		janela.setLocation(50, 50);
		//define que quando a janela é fechada a aplicação também é fechada
		janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		JButton botoes= new JButton("OK");
		janela.add(botoes);
		janela.setVisible(true);
	}
}
```

EXERCÍCIOS

Reproduz os exemplos desta ficha, compila-os e executa-os (melhora os programas, caso consideres pertinente), inseridos num projeto de nome **F1exemplos.**

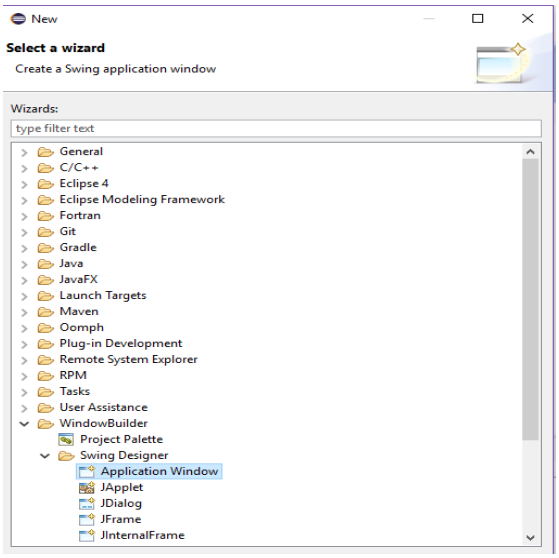
1º criar o projeto e só depois criar os vários contentores e dentro destes os componentes desejados.

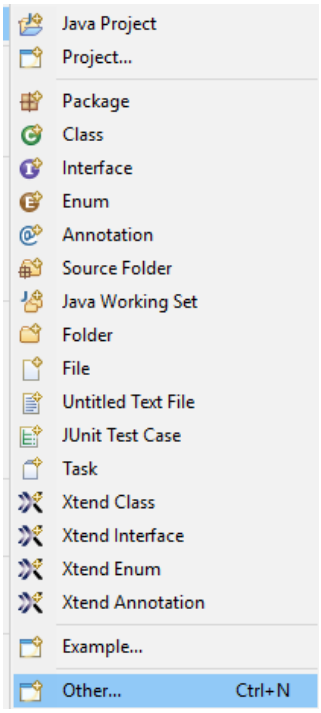
1. Criar um programa que permita criar uma janela que simule um a caixa de mensagens instantâneas, com layout semelhante ao seguinte:



a) No Eclipse cria um novo projeto de nome Java2F1 do seguinte modo:

b) Botão direiro no scr ->new->Other…
->WindowsBuilder->Aplication Window



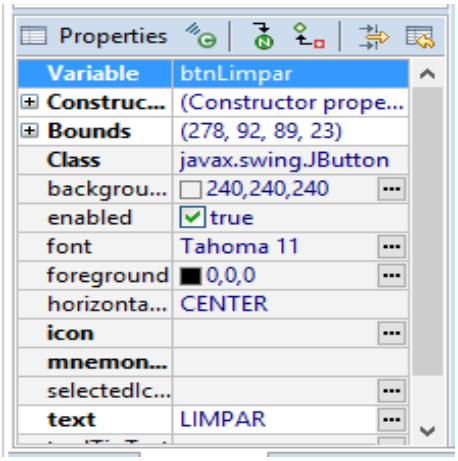


c) Insere na janela um **JPanel**, que encontrarás na secção **containers**, onde irás inserir os outros objetos (seleciona-o e arrasta-o…).

d) Na secção Layouts seleciona o **Absolute layout** (arrasta-o para a janela criada).

e) Da secção **Components** adiciona um painel de texto **(JTextPane)**, uma caixa de texto (**JTextField**) e dois botões **(JButton)**

f) Altera o nome da variável do JTextField para CaixaTexto e do JTextPanel para PainelTexto (ver proprerties ->variable).



g) Atribui o nome ENVIAR e LIMPAR aos botões (ver proprerties ->text).

h) Faz um duplo clique no botão LIMPAR. Surgirá o código:

```
JButton btnLimpar = new JButton("LIMPAR");
btnLimpar.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent arg0) {
                        *

            }
});
```

Acrescenta o seguinte código no local *, isto é, ação a executar ao premir o botão, que permite limpar o JTextPanel:

```
            PainelTexto.setText("");
```

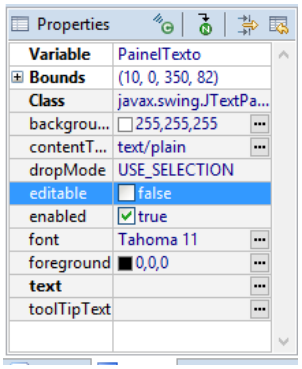
Caso não esteja definido o objeto é necessário passar o seu construtor para dentro da classe e declará-lo como private final.

i) Faz um duplo clique no botão ENVIAR. Surgirá o código:

```
JButton btnEnviar = new JButton("ENVIAR");
btnEnviar.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                  *

            }
});
```

Acrescenta o seguinte código no local *, isto é, ação a executar ao premir o botão, que permite limpar "enviar" para o JTextPanel (PainelTexto) o texto digitado no JTextBox (CaixaTexto) e limpar o texto na caixa de texto:

```
PainelTexto.setText(PainelTexto.getText()+ CaixaTexto.getText()+"\n");
CaixaTexto.setText("");   //para limpar a caixa de texto
CaixaTexto.requestFocus();  // para a caixa de texto receber o foco
                              do cursor
```



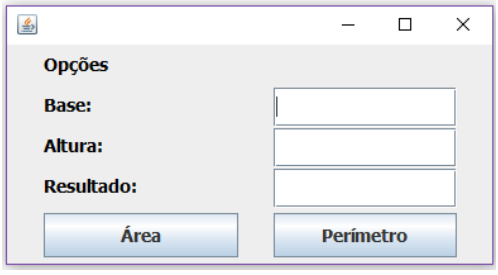
j) Seleciona o Painel de Texto e nas propriedades retira o visto da propriedade *editable* (para que o texto não possa ser aí alterado).

k) Para colocar o foco na caixa de texto logo ao abrir a janela: Escrever o código seguinte, dentro do método inicialize:

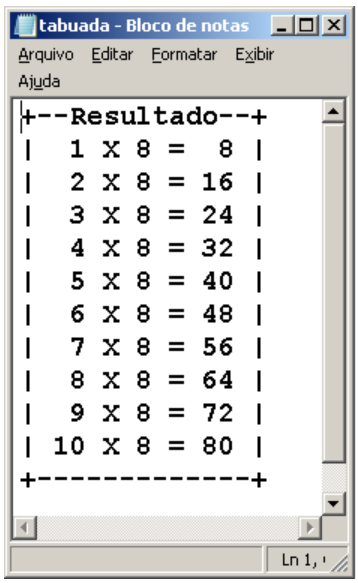
```
javax.swing.SwingUtilities.invokeLater(new Runnable() {
      public void run() {
            CaixaTexto.requestFocusInWindow();
      }
});
```

2. Constrói uma janela semelhante à seguinte (JFrame), de nome **F14EX2**, que te permita introduzir os valores da base e altura de um retângulo e apresentar o resultado da sua área e perímetro.

Após introduzidos os valores nas duas 1.ªs caixas de texto deverá apresentar o resultado na 3.ª caixa de texto, calculando a área ou o perímetro após clicar no botão respetivo.
Acrescenta um botão para limpar os valores das caixas de texto.
Consulta o manual de Swing-Java na ClassRoom (páginas 36 e 37).



3. Copia todo o código relativo ao Exercício1 para um novo projeto, de nome F14EX3 (deves alterar o nome da JFrame e da classe). Altera-o de modo a introduzires na caixa de texto o nº correspondente à tabuada a efetuar e aparecer o resultado, de forma semelhante à seguinte, no painel de texto, quando pressionado o botão enviar.



4. Constrói uma janela semelhante à seguinte (JFrame). De nome F14EX4, que te permita introduzir dois valores e calcular a soma, subtração, produto e divisão desses valores (Calculadora básica)

a. Em modo Design aplica à janela um Layout do tipo **FlowLayout (Na Pallete-> separador Layouts-> FlowLayout,** arrastar e largar na janela).
Verifica as caraterísticas de um layout deste tipo.

b. Coloca o alinhamento do FlowLayout ao centro (CENTER) e distância às bordas da janela, hgap e vgap iguais a 10. Desenha duas caixas de texto (tamanho a definir- Collumns), com nome C1 e C2, respetivamente.

c. Desenha quatro botões com os nomes que se apresentam na figura acima (tamanho de letra 15 e bold).

d. Atribui um nome à janela (**Properties->Title**) e altera a cor de fundo a teu gosto.

e. Adiciona o código necessário aos botões para que o resultado seja exibido numa Caixa de diálogo (objeto de um JOptionPane) e após clicar em OK seja efetuada a "limpeza" das caixas de texto.
No código seguinte altera o argumento (arg0) do ActionEvent para uma variável qualquer, digamos ***e***.

```
JButton Somar = new JButton("Somar");
        Somar.setFont(new Font("Tahoma", Font.BOLD, 15));
        Somar.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent arg0) {

                // * escrever código aqui
}
```

Acrescenta o seguinte código no local *, isto é, ação a executar ao premir o botão, que permite somar os valores das caixas de texto e exibir o resultado numa caixa de diálogo (JOptionPane):

```
try{
      Num1=Double.parseDouble(C1.getText());
      Num2=Double.parseDouble(C1.getText());

      JOptionPane.showMessageDialog(null,"Soma=
       "+(Num1+Num2),"RESULTADO",JOptionPane.PLAIN_MESSAGE);

 // Sintaxe: JOptionPane.showMessageDialog(null, message, title, messageType);
}
```

```
catch(Exception erro){

JOptionPane.showMessageDialog(null, "ERRO! "+erro,"", JOptionPane.ERROR_MESSAGE);
}
```

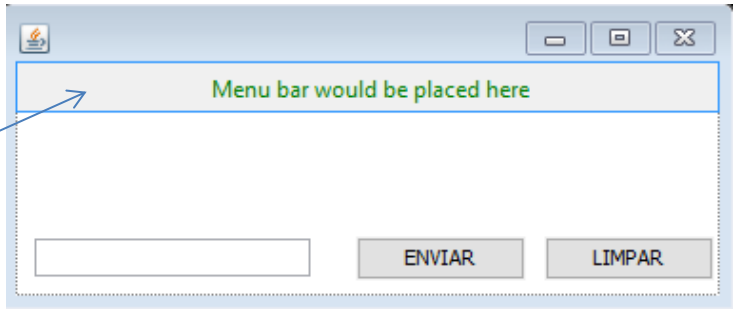
Para **JOptionPane**, consultar: https://www.tutorialspoint.com/swing/swing_joptionpane.htm
https://www.mkyong.com/swing/java-swing-how-to-make-a-simple-dialog/

f. Escreve o código para as outras operações.

g. Com a ajuda do segundo site tenda modificar a caixa de mensagem de erro de modo a que apareça na mesma, um ícone de um smile triste.

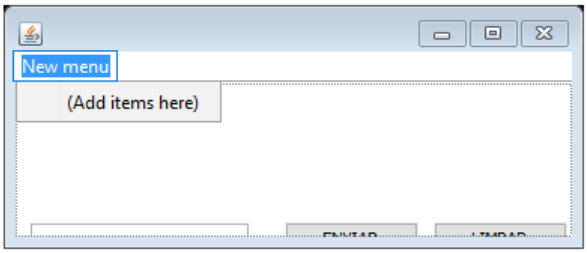
5. Refaz o exercício (copia a frame e dá-lhe o nome EX5, mas agora acrescentando uma caixa de texto que receba o resultado de cada uma das operações (ver exercício 2).

6. Abre o exercício1 e procede às seguintes alterações:
Em modo de Design acrescenta à janela um JMenuBar (Menu->JMenuBar)



Largar aqui

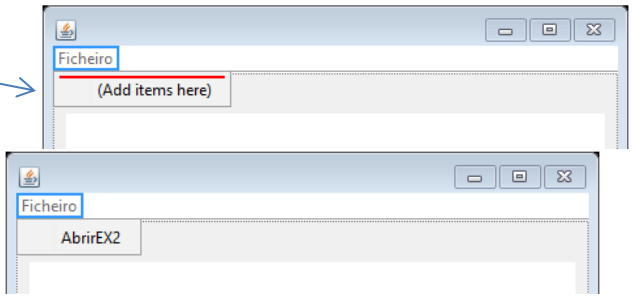
a. Para acrescentar itens de menu, em Menu selecionar um Jmenu a introduzir na barra de menu, com o nome ***Ficheiro***



b. Reajusta a dimensão da janela e adiciona um novo menu com o nome ***Ajuda***

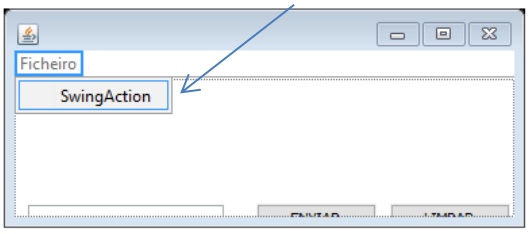
c. Adiciona um JMenuItem (submenu para o menu ficheiro)

d. Atribui-lhe o nome de Abrir EX2



e. Clica com o botão direito do rato em **Abrir EX2->SetAction->New…**

Foi criada uma classe interna, com dois métodos internos com o código seguinte (verifica):



```
private class SwingAction extends AbstractAction {
            public SwingAction() {  //método constructor da classe

                  putValue(NAME, "SwingAction");
                  putValue(SHORT_DESCRIPTION, "Some short description");
            }

            public void actionPerformed(ActionEvent e) { //metodo para responder ao
                                                           clique no menu
      } }
```

f. Substituir o nome da classe *SwingAction* por AbrirEX2 e na ação putValue (para mais facilmente associarmos à classe a ação a efetuar). Deve ser efetuada a substituição na classe interna e também na instrução de criação do construtor (na classe `principal`):

```
private final Action action = new SwingAction();
```

Substituir também, na classe interna, o texto da ação putValue "`"Some short description"` por "`"Calcular área e perímetro de um rectângulo"`

g. No método *actionPerformed* adicionar o seguinte código:

```
new F14EX2().setVisible(true); //abre a janela de nome F3EX2 (verifica se a tua
                                  janela tem o mesmo nome), alternativa a:
```

```
F3EX2 f=new F3EX2();
f.setVisible(true);
```

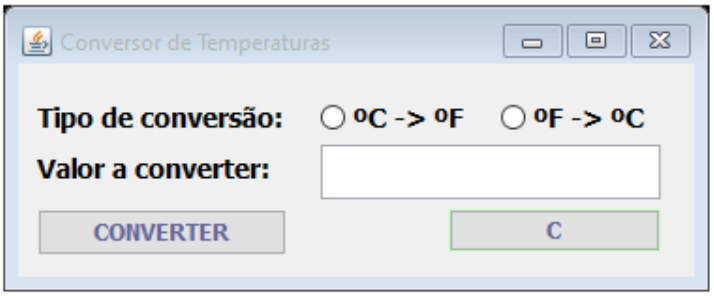
```
frame.dispose(); //fecha a janela atual cujo nome é frame
// por estarmos dentro de outra classe, pode ser necessário, para poder aceder à
instância da classe EX1, atual, fazer referência à mesma, nesse caso , usar a
clausula this: EX1.this.dispose();
```

h. Adiciona um JMenuItem para um JMenu, com o nome Sair, que quando clicado deverá fechar a aplicação.

Ação: ***System.exit(0);*** // sair da aplicação

Substitui o texto da ação putValue "`"Some short description"` por `"Sair da aplicação"`

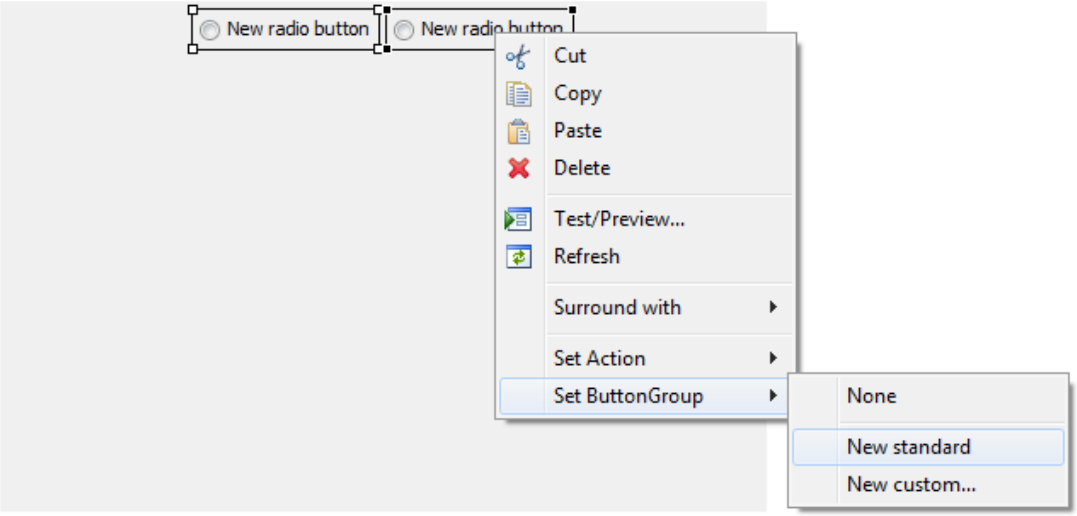
7. Cria uma nova janela de nome F14EX6, com layout semelhante ao seguinte (utiliza um Container JPanel com Absolute Layout):



Letra das labels: Tipo de letra: Tahoma, tamanho 14 e negrito
Esta interface gráfica permitirá receber um valor em ºC, convertendo-o para ºF, ou vice-versa.
O resultado deverá ser apresentado na caixa de texto.
Nota: médodo associado ao RadioButton: `isSelected()`
Se o objeto estiver selecionado deve ser calculado ….
Considera que :
F=C*9/5+32 e **C=(F-32)*5/9**

*Compacta a pasta **Java2F2** e envia-a para a classroom.*

Nota: para que seja permitida apenas a seleção de um único RadioButton/botão de seleção (seleção exclusiva) deves selecionar os Radiobuttons –A clicar com o botão direito e adicioná-los a um ButtonGroup, como mostra a figura. Esta opção também é válida para as CheckBox/caixa de verificação.



Para limpar a seleção do grupo de botões usar o método: clearSelection()

*Compacta a pasta **Java2F2** e envia-a para a classroom*

Bibliografia

http://www.mfbarcell.es/docencia_uned/poo/tema11/capitulo_11.pdf
https://www.aluracursos.com/blog/biblioteca-swing
https://docs.oracle.com/en/java/javase/16/docs/api/java.desktop/java/awt/package-summary.html
https://docs.oracle.com/en/java/javase/16/docs/api/java.desktop/javax/swing/package-summary.html

NÃO DEI AOS ALUNOS A PARTIR DAQUI, PORQUE NÃO DEI FICHEIROS!!!

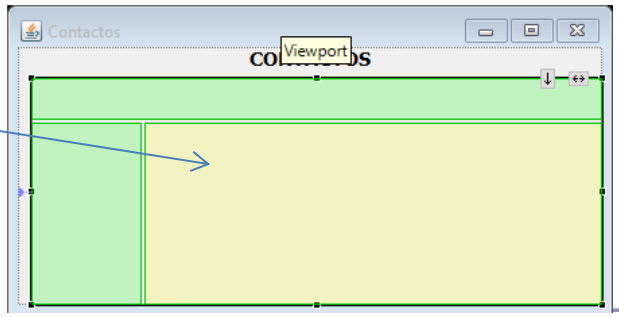
8. Cria uma nova janela de nome F3EX7, com layout semelhante ao seguinte:

   Esta interface permite ler os nomes e os contactos do ficheiro "contactos.txt" e apresenta-los numa tabela incluída ma janela criada.
   O ficheiro "contactos.txt" deve ser criado pelo utilizador, na pasta onde se encontra o projeto, cada nome e contacto correspondente devem ser escritos na mesma linha.

   a. Após criada a janela seleciona um componente Caixa de texto para colocar na parte superior da janela, com o nome – CONTACTOS;

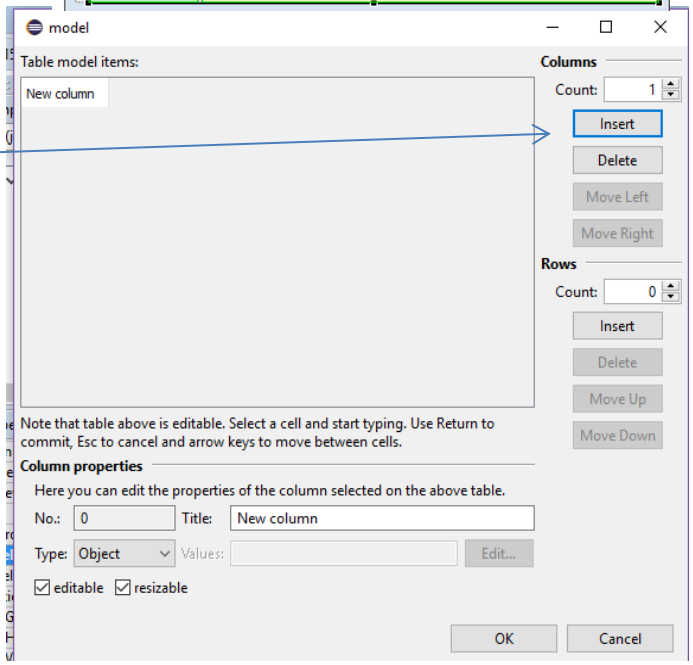
   b. Cria um contentor do tipo ***scrollPane*** para ocupar a área inferior:

   c. Seleciona um componente do tipo JTable a inserir dentro do scrollPane:



   d. Nas propriedades do JTable clicar em **Model;**

   e. Inserir as colunas NOMES e CONTACTOS;



   f. Acrescenta a linha seguinte na classe:

```
public class F3EX7 extends JFrame {
      private JTable table;
      private DefaultTableModel modelo = new DefaultTableModel();
```

g. No construtor onde está o código

```
            …
      table.setModel(new DefaultTableModel(new Object[][] {}, new String[] {"NOMES",
      "CONTACTOS"} )
                  {
                        …
```

elimina todo o código seguinte:

```
table.setModel(new DefaultTableModel(new Object[][] {}, new String[] {"NOMES",
"CONTACTOS"} )
            {
                  Class[] columnTypes = new Class[] {
                        String.class, String.class
                  };
                  public Class getColumnClass(int columnIndex) {
                        return columnTypes[columnIndex];
                  }
            });

                   DefaultTableModel model;
```

e escreve as instruções manualmente:

```
      ...
table = new JTable(modelo);
            modelo.addColumn("NOMES");
                  modelo.addColumn("CONTACTOS");

table.getColumnModel().getColumn(0).setPreferredWidth(131);
            table.getColumnModel().getColumn(1).setPreferredWidth(91);
            scrollPane.setViewportView(table);
            getContentPane().setLayout(groupLayout);

            try{
                  File f=new File("contactos.txt");
                  FileReader fr=new FileReader(f);
                  BufferedReader br=new BufferedReader(fr);
                  int i=0, j=0;
                  String [] registo= new String[] {null,null};  //criar array de
registos
                  while (br.ready()){
                  //     model.setValueAt(aValue, row, column);
                        modelo.addRow(registo); //adicionar nova linha na tabela
                  modelo.setValueAt(br.readLine(),i,0);
                  modelo.setValueAt(br.readLine(),i,1);
                  i++;
                  }
                  br.close();
            }
            catch(Exception e){
                  System.out.println(e.getMessage());
```

```
            e.printStackTrace();
            }
```

...

h. Executa o programa.