Linguagem de Programação II

Michelle Hanne Soares de Andrade
michellehanne@cefetmg.br

CEFET-MG

Sumário:

- ✓ Graphical User Interface (GUI)
- ✓ Formatação
- ✓ Instalando Interface gráfica – Window Builder

Graphical User Interface (GUI)

É onde os resultados são apresentados em modo gráfico.

Essa interface é formada através **de componentes GUI**, conhecidos por controles ou **widgets**. Esses componentes são objetos que fazem a interação com usuário por teclado, mouse ou outros dispositivos que venham a servir para entrada de dados.

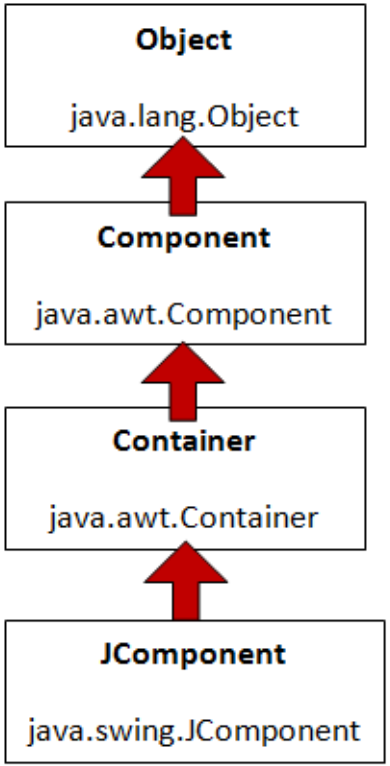
Os elementos básicos necessários para criar um GUI residem em dois pacotes:

- **java.awt**: Abstract Windows Toolkit (classes básicas);
- **javax.swing:** Swing Components - fornece melhores alternativas aos componentes definidos na classe java.awt.

Fonte: https://www.devmedia.com.br/introducao-a-interface-gui-no-java/25646

Graphical User Interface (GUI)

- A diferença entre o **GUI Swing** e **AWT**, é na aparência e comportamento dos componentes, ou seja, quando criado por **AWT**, a aparência e comportamento de seus componentes são diferentes para cada plataforma e enquanto feito por GUI **Swing**, **a aparência e comportamento funcionam da mesma forma para todas as plataformas.**
- Os componentes AWT são mais pesados, pois requerem uma interação direta com o sistema de janela local, podendo restringir na aparência e funcionalidade, ficando menos flexíveis do que os componentes GUI Swing.



Fonte: https://www.devmedia.com.br/introducao-a-interface-gui-no-java/25646

Graphical User Interface (GUI)

Principais componentes:

- **JLabel -** Exibe texto não editável ou ícones.
- **JTextField** – Insere dados do teclado e serve também para exibição do texto editável ou não editável.
- **JButton** – Libera um evento quando o usuário clicar nele com o mouse.
- **JCheckBox** – Especifica uma opção que pode ser ou não selecionada.
- **JComboBox** – Fornece uma lista de itens onde possibilita o usuário selecionar um item ou digitar para procurar.
- **JList** – Lista de itens onde pode ser selecionado vários itens.
- **JPanel** – É a área onde abriga e organiza os componentes inseridos.

Fonte: https://www.devmedia.com.br/introducao-a-interface-gui-no-java/25646

Graphical User Interface (GUI)

- **Classe Container:** Os **Containers** são janelas que podem ser usados para organizar e exibir outros componentes na tela, pelo Container ser um **Component**, podemos colocar **Containers** dentro de outros **Containers** para facilitar a organização.
- **Classe Jcomponent:** Está dentro do pacote **javax.swing** sendo uma subclasse de **Container**, mas é a superclasse que declara os atributos e comportamentos para todos os componentes Swing. Por ser a subclasse de Container, todos os componentes Swing acabam sendo Containers.
- **JOptionPane**: Encontra-se no pacote **javax.swing**, e por ser um componente faz parte da Classe JComponent, permitindo ao usuário inserir informações nas caixas de diálogos, podendo exibir informações ou avisos.

Acesse o module-info.java e altere o conteúdo para:

```
module Exemplos_GUI {
requires java.desktop;
}
```

Fonte: https://www.devmedia.com.br/introducao-a-interface-gui-no-java/25646

Exemplo Primeira Tela

```
package br.com.exemplos.gui;
import javax.swing.JOptionPane;

public class Exemplo_PrimeiraTela {

          public static void main(String[] args) {
                                        String nome = JOptionPane.showInputDialog("Digite o
nome: ");
                                        String                    sobreNome                    =
JOptionPane.showInputDialog("Digite o sobrenome: ");

                                        String nomeCompleto = nome + " " + sobreNome;

                                        JOptionPane.showMessageDialog(null, "Nome Completo:
"+nomeCompleto,"Informação",JOptionPane.INFORMATION_MESSAGE);
          }
}
```

A linha que invoca o método **showMessageDialog**, imprime os resultados armazenados na variável "**nomeCompleto**". Abaixo a lista de argumentos desse método. Argumento que marca a posição que será **exibida da caixa na tela**, como não estamos trabalhando com **frames** o padrão é **null**. Barra de título exibe, "Informação" . O último argumento é o tipo da saída da mensagem que exibe através do diálogo por meio de constante.

Fonte: https://www.devmedia.com.br/introducao-a-interface-gui-no-java/25646

Exemplo Média

```
package br.com.exemplos.gui;
import javax.swing.JOptionPane;

public class Exemplo_Media {

           public static void main(String[] args) {
                     float nota1, nota2, calculaMedia;
                     nota1 = Float.parseFloat(JOptionPane.showInputDialog ("Digite a primeira
nota"));
        nota2 = Float.parseFloat (JOptionPane.showInputDialog("Digite a segunda nota"));
        calculaMedia = (nota1 + nota2) / 2;

        JOptionPane.showMessageDialog(null, "Resultado da Média = "+ calculaMedia,"Resultado",
JOptionPane.WARNING_MESSAGE);
    }

}
```

Fonte: https://www.devmedia.com.br/introducao-a-interface-gui-no-java/25646

Exemplo Média

Mostramos as constantes que representam os **tipos das mensagens** quando são mostradas na saída de um resultado através **JOptionPane**.

INFORMATION_MESSAGE -> Indica uma mensagem Informativa

ERROR _MESSAGE -> Indica um erro ao usuário

QUESTION_MESSAGE -> Mostra uma questão ao usuário

WARNING_MESSAGE -> Alerta o usuário

PLAIN_MESSAGE -> Sem ícone

Fonte: https://www.devmedia.com.br/introducao-a-interface-gui-no-java/25646

Containers e Componentes

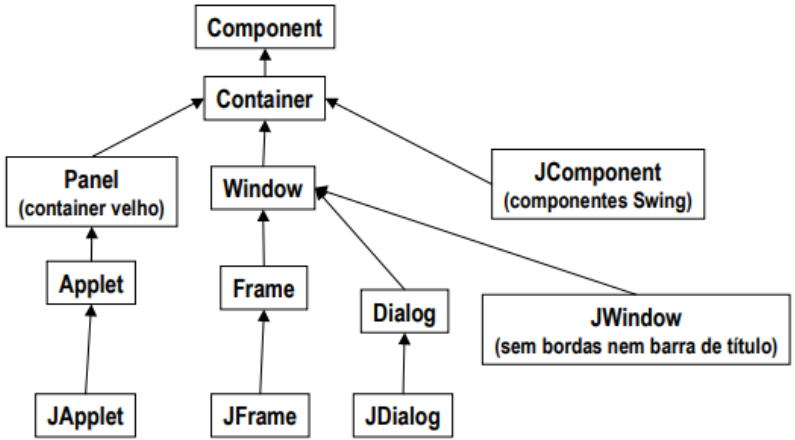
• Uma interface gráfica em Java é baseada em dois elementos:
– containers: servem para agrupar e exibir outros componentes
– componentes: botões, labels, scrollbars, etc.

• Dessa forma, todo programa que ofereça uma interface vai possuir pelo menos um container, que pode ser:
– JFrame: janela principal do programa
– JDialog: janela para diálogos
– JApplet: janela para Applets

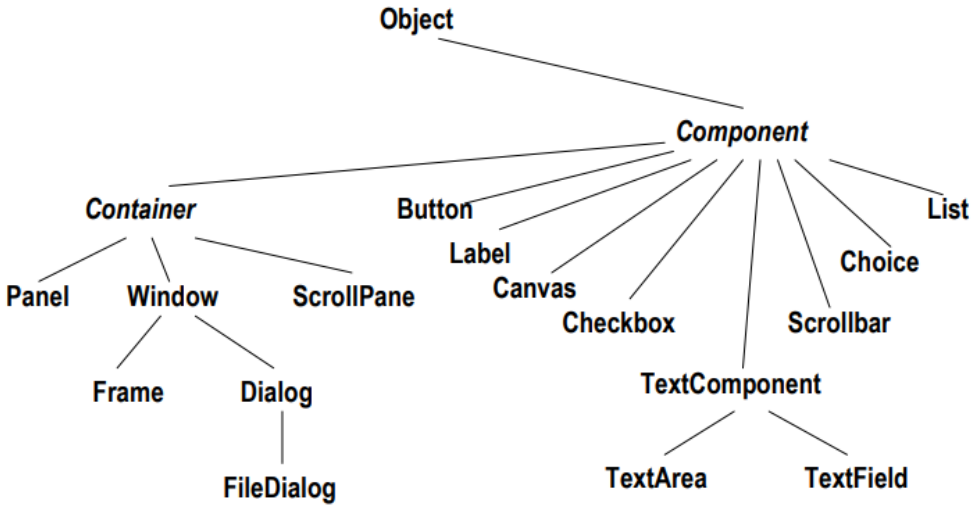
Para **construirmos uma interface gráfica em JAVA**, adicionamos componentes (Botões, Menus, Textos, Tabelas, Listas, etc.) sobre a área da janela.
• Por essa razão a área da janela é um container, ou seja, um elemento capaz de armazenar uma lista de componentes.

Fonte: https://www.inf.pucrs.br/flash/lapro2/lapro2_gui.pdf

Criando uma Janela

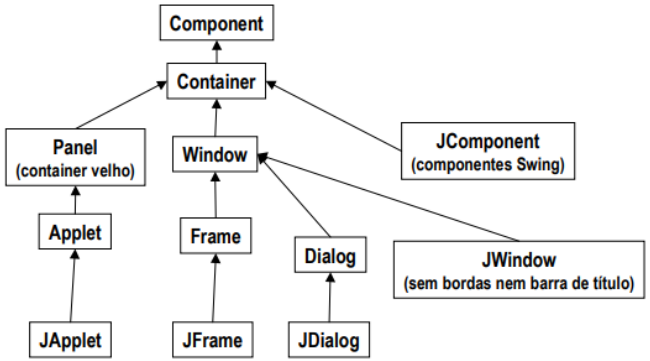


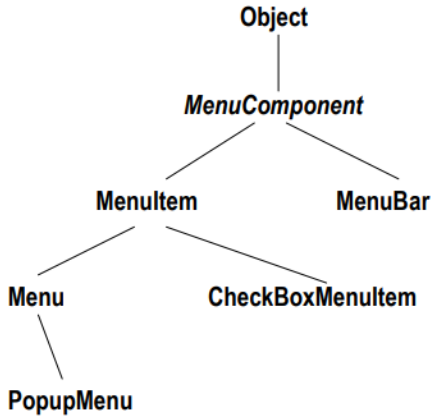
Classes de AWT



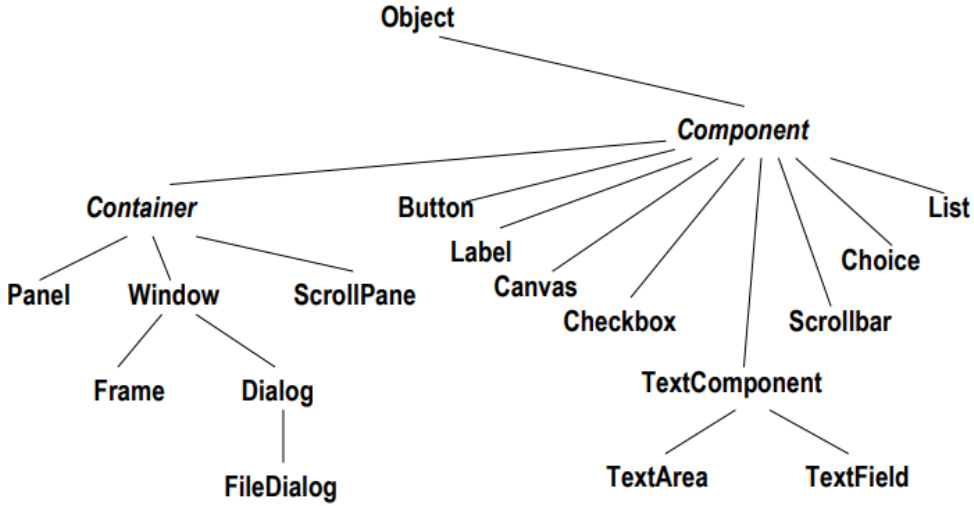
Fonte: https://www.inf.pucrs.br/flash/lapro2/lapro2_gui.pdf

Criando uma Janela





Classes de AWT



Fonte: https://www.inf.pucrs.br/flash/lapro2/lapro2_gui.pdf

Package Swing

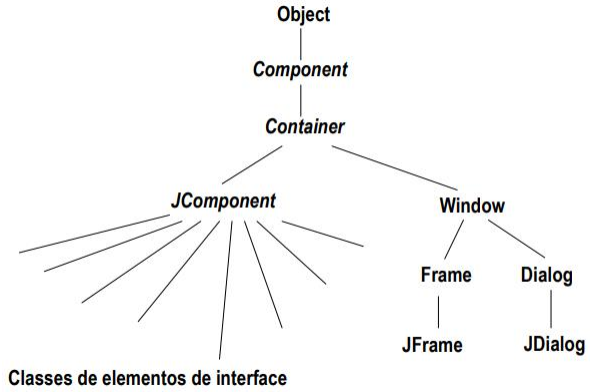
Criado em 1997 é uma extensão da AWT (Abstract Window Toolkit)

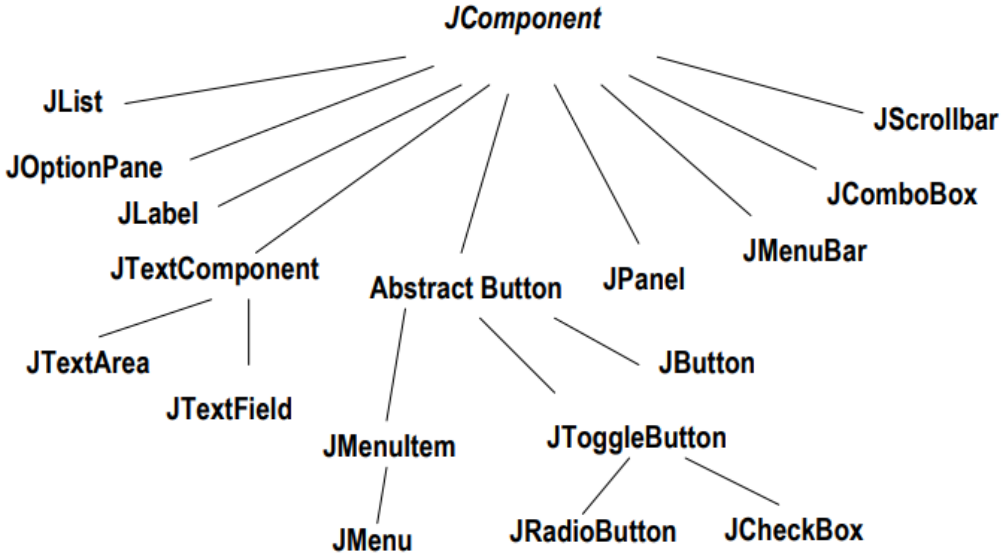
- Classes implementadas inteiramente em Java
- Mesma estrutura que os componentes AWT
- Componentes que fornecem melhores alternativas para a implementação de interfaces gráficas
  - JButton no lugar de Button,
  - JFrame no lugar de Frame, etc.
- As classes de Swing fazem parte de um conjunto mais genérico de classes com capacidades gráficas: **JFC (Java Foundation Classes**)

Fonte: https://www.inf.pucrs.br/flash/lapro2/lapro2_gui.pdf

Package Swing

Visuais:
• botões, menus, barras de ferramentas, etc.
– Não-visuais, de auxílio aos outros:
• root pane, panel, layered pane, etc.
• Pacote javax.swing





Fonte: https://www.inf.pucrs.br/flash/lapro2/lapro2_gui.pdf

Containers e Componentes

Alguns atributos de componentes:

– posição (x,y): posição do objeto em relação ao seu container;
– nome do componente (myWindow.setName(“Teste”););
– tamanho: altura e largura;
– cor do objeto e cor de fundo;
– fonte
– aparência do cursor;
– objeto habilitado ou não (isEnabled(), myWindow.setEnabled);
– objeto visível ou não (isVisible(), myWindow.setVisible);
– objeto válido ou não.

Outros componentes disponíveis:

– Button (JButton)
– Menu (JMenu)
– Text Component (JTextComponent)
– List (JList)
– Table (JTable)
– Container

Exemplos de métodos:

– void setBounds(int x, int y, int width, int height);
– void setBounds(Rectangle rect);
– Rectangle getBounds();
– void setSize(Dimension d);
– Dimension getSize();
– setLocation(int x, int y);
– setLocation(Point p);
– Point getLocation();

Fonte: https://www.inf.pucrs.br/flash/lapro2/lapro2_gui.pdf

Exemplo Swing Demo

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


public class Exemplo_SwingDemo implements ActionListener {
            JLabel jlab;
            Exemplo_SwingDemo() {

                        // Cria um contêiner JFrame.
                        JFrame jfrm = new JFrame("Uma janela simples App");
                        // Fornece um tamanho inicial para o quadro.
                        jfrm.setLayout(new FlowLayout());

                        jfrm.setSize(220, 90);  //Define as dimensões do quadro.
                        // Encerra o programa quando o usuário fecha o aplicativo.
                        jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);//      Encerra quando fechado

                        // Cria dois botões.
                        JButton jbtnUp = new JButton("Up");
                        JButton jbtnDown = new JButton("Down");

                        // Adiciona ouvintes de ação.
                        jbtnUp.addActionListener(this);
                        jbtnDown.addActionListener(this);
```

Fonte: SCHILDT, 2015

Exemplo Swing Demo

```
                // Adiciona os botões ao painel de conteúdo.
                jfrm.add(jbtnUp);
                jfrm.add(jbtnDown);


                // Cria um rótulo.
                jlab = new JLabel("Pressione o botão.");
                jfrm.add(jlab);

                // Exibe o quadro.
                jfrm.setVisible(true);
        }

        // Trata eventos de botão.
        public void actionPerformed(ActionEvent ae) {
                if(ae.getActionCommand().equals("Up"))
                                jlab.setText("Você pressionou o botão Up.");
                else
                                jlab.setText("Você pressionou o botão Down. ");
        }

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                SwingUtilities.invokeLater(new Runnable()  {
                                        public void run() {
                                                new Exemplo_SwingDemo();
                }
                });
}
```

Fonte: SCHILDT, 2015

Formatando campos com JFormattedTextField

As máscaras são muito utilizadas em sistemas comerciais, pois elas ajudam na padronização da visualização de dados. Um exemplo de máscara é o telefone: "(51)3636-0000" ou o CEP: "92-110.310".

Antes de criarmos e utilizarmos um **JFormattedTextField** devemos criar um objeto **MaskFormatter** e configurar uma máscara.

```
MaskFormatter mascaraCpf = new MaskFormatter("###.###.###-##");
```

```
JFormattedTextField cpf = new JFormattedTextField(mascaraCpf);
```

Fonte: https://www.devmedia.com.br/java-swing-conheca-os-componentes-jtextfield-e-jformattedtextfield/30981

Formatando campos com JFormattedTextField

Quando criamos um **MaskFormatter** podemos utilizar ao invés de **“#”**, outros caracteres, dependendo do tipo de restrição que desejamos implementar no **JFormattedTextField**. Esses caracteres são definidos abaixo:

- **"#"** indica que qualquer número poderá ser inserido (0-9);
- **"U"** indica que qualquer letra (a-z) poderá ser inserida. A máscara converterá letras minúsculas em maiúsculas;
- **"L"** indica qualquer letra (a-z) poderá ser inserida. A máscara converterá letras maiúsculas em minúsculas;
- **"?"** indica qualquer letra (a-z) poderá ser inserida. A máscara manterá a letra inserida;
- **"A"** indica qualquer letra ou numero (0-9 e a-z) poderá ser inserido;
- **"H"** indica qualquer caracter hexadecimal (0-9 a-f) poderá ser inserido;
- **"*"** indica qualquer coisa, incluindo caracteres especiais poderão ser inseridos.

Fonte: https://www.devmedia.com.br/java-swing-conheca-os-componentes-jtextfield-e-jformattedtextfield/30981

Formatando campos com JFormattedTextField

Segue abaixo algumas máscaras prontas que podemos utilizar em nossos projetos:

Telefone Internacional: "+##(##)####-####"
Telefone Nacional: "(##)####-####"
CEP: "##.###-###" ou "#####-###"
CPF: "###.###.###-##"
Placa de automóveis: "UUU-####"
CNPJ: "##.###.###/####-##"
Título de eleitor: "#########/##"
Data de nascimento: "##/##/####"

Fonte: https://www.devmedia.com.br/java-swing-conheca-os-componentes-jtextfield-e-jformattedtextfield/30981

Formatando campos com JFormattedTextField

Para exemplificar o uso do JFormattedTextField segue o exemplo presente no exemplo abaixo, onde criamos quatro campos com máscaras, são eles: CEP, Telefone, CPF, Data.

Fonte: https://www.devmedia.com.br/java-swing-conheca-os-componentes-jtextfield-e-jformattedtextfield/30981

Formatando campos com JFormattedTextField

```
package br.com.exemplos.gui;
import java.awt.Container;
import java.text.ParseException;

import javax.swing.JFormattedTextField;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.text.MaskFormatter;

public class TestandoJFormattedTextField extends JFrame {

    private static final long serialVersionUID = 1L;

public static void main(String[] args) {
TestandoJFormattedTextField field = new TestandoJFormattedTextField();
    field.testaJFormattedTextField();
}
```

Fonte: https://www.devmedia.com.br/java-swing-conheca-os-componentes-jtextfield-e-jformattedtextfield/30981

Formatando campos com JFormattedTextField

```
private void testaJFormattedTextField() {
        Container janela = getContentPane();
        setLayout(null);

        //Define os rótulos dos botões
        JLabel labelCep = new JLabel("CEP: ");
        JLabel labelTel = new JLabel("Telefone: ");
        JLabel labelCpf = new JLabel("CPF: ");
        JLabel labelData = new JLabel("Data: ");
        labelCep.setBounds(50,40,100,20);
        labelTel.setBounds(50,80,100,20);
        labelCpf.setBounds(50,120,100,20);
        labelData.setBounds(50,160,100,20);

        //Define as máscaras
        MaskFormatter mascaraCep = null;
        MaskFormatter mascaraTel = null;
        MaskFormatter mascaraCpf = null;
        MaskFormatter mascaraData = null;
```

Fonte: https://www.devmedia.com.br/java-swing-conheca-os-componentes-jtextfield-e-jformattedtextfield/30981

Formatando campos com JFormattedTextField

```
try{
                mascaraCep = new MaskFormatter("#####-###");
                mascaraTel = new MaskFormatter("(##)####-####");
                mascaraCpf = new MaskFormatter("#########-##");
                mascaraData = new MaskFormatter("##/##/####");
                mascaraCep.setPlaceholderCharacter('_');
                mascaraTel.setPlaceholderCharacter('_');
                mascaraCpf.setPlaceholderCharacter('_');
                mascaraData.setPlaceholderCharacter('_');
        }
        catch(ParseException excp) {
                System.err.println("Erro na formatação: " + excp.getMessage());
                System.exit(-1);
        }

        //Seta as máscaras nos objetos JFormattedTextField
        JFormattedTextField jFormattedTextCep = new JFormattedTextField(mascaraCep);
        JFormattedTextField jFormattedTextTel = new JFormattedTextField(mascaraTel);
        JFormattedTextField jFormattedTextCpf = new JFormattedTextField(mascaraCpf);
        JFormattedTextField jFormattedTextData = new JFormattedTextField(mascaraData);
        jFormattedTextCep.setBounds(150,40,100,20);
        jFormattedTextTel.setBounds(150,80,100,20);
```

Fonte: https://www.devmedia.com.br/java-swing-conheca-os-componentes-jtextfield-e-jformattedtextfield/30981

Formatando campos com JFormattedTextField

```
            jFormattedTextCpf.setBounds(150,120,100,20);
        jFormattedTextData.setBounds(150,160,100,20);

        //Adiciona os rótulos e os campos de textos com máscaras na tela
        janela.add(labelCep);
        janela.add(labelTel);
        janela.add(labelCpf);
        janela.add(labelData);
        janela.add(jFormattedTextCep);
        janela.add(jFormattedTextTel);
        janela.add(jFormattedTextCpf);
        janela.add(jFormattedTextData);
        setSize(400, 250);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(true);
}

}
```

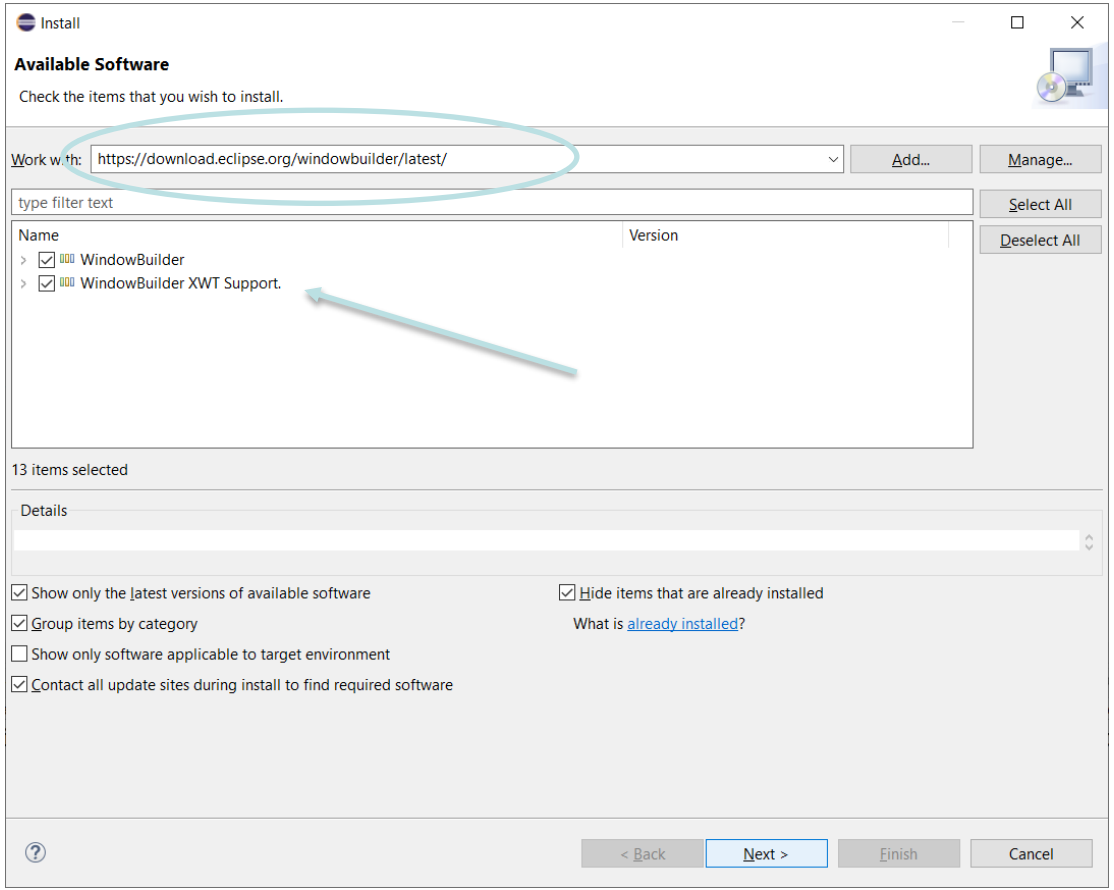
Fonte: https://www.devmedia.com.br/java-swing-conheca-os-componentes-jtextfield-e-jformattedtextfield/30981

Interface Gráfica

Window Builder é um plug-in para o Eclipse que possibilita a criação de interface gráfica Java, responsável pelo design da aplicação, utiliza as bibliotecas como Swing dentre outras.

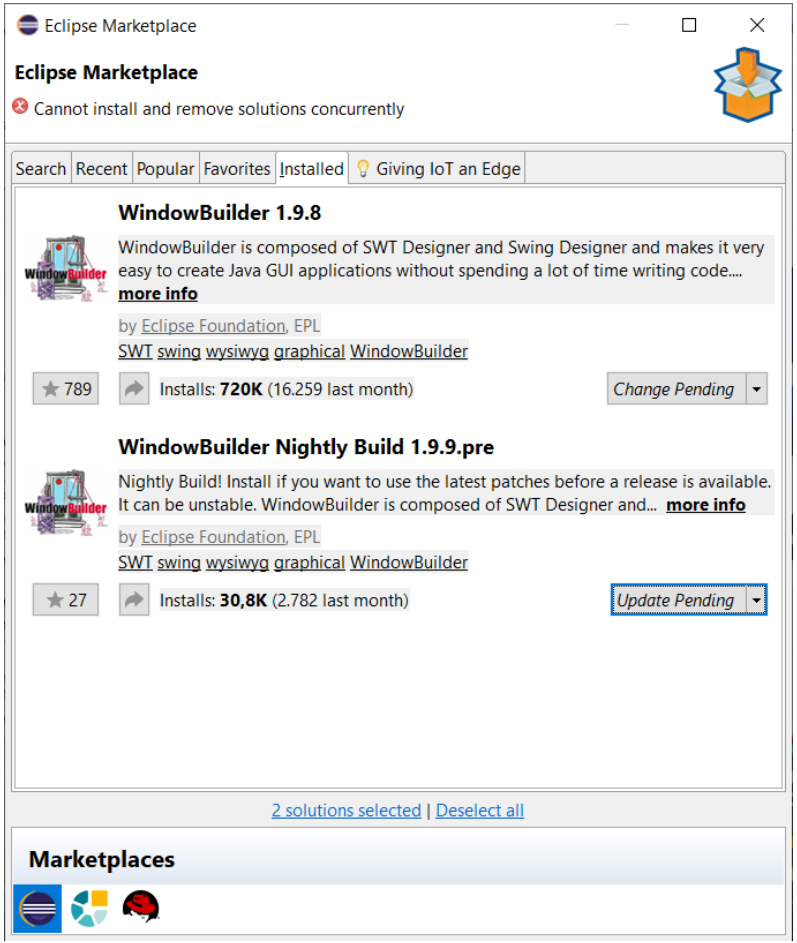
1- Copiar o endereço abaixo
https://download.eclipse.org/windowbuilder/latest/
2- Acessar no Eclipse a opção Help->Install New Sofltware
3- Colar o endereço copiado e pressionar <enter>

Interface Gráfica



4- Selecione as opções e pressione <Next>
5- Em seguida Next e Finish

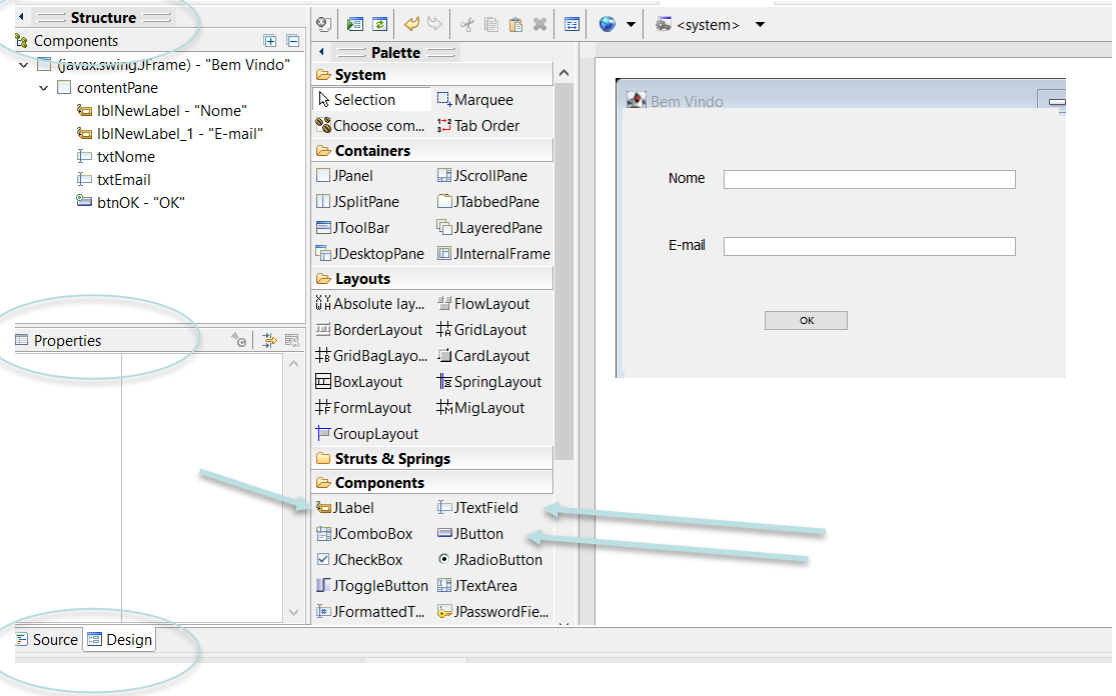
Interface Gráfica



A outra opção é acessar Help-> **Eclipse Marketplace**

Digitar Window Builder e instalar

Projeto com Window Builder



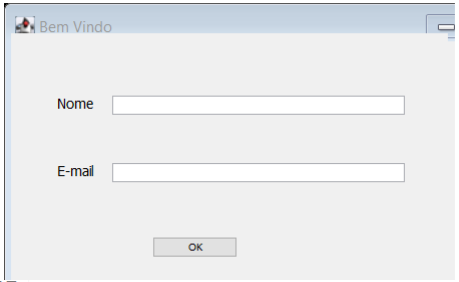
Clique na opção Design será mostrado a interface do Window Builder.

1- Adicionar o “Absolute Layout” que permite criar um contentPane

2- Acionar os elementos Jlabel e JTextField para compor a tela

Todos os elementos possuem propriedades que podem ser alteradas, basta selecionar o objeto que as propriedades serão exibidas abaixo dos “componentes”.

Projeto com Window Builder



Adicione um elemento Jbutton.
Clique duas vezes no botão, será aberto o código fonte, vamos apenas colocar uma mensagem na tela:

```
65
66          txtEmail = new JTextField();
67          txtEmail.setBounds(101, 129, 292, 19);
68          contentPane.add(txtEmail);
69          txtEmail.setColumns(10);
70
71          JButton btnOK = new JButton("OK");
72          btnOK.addActionListener(new ActionListener() {
73              public void actionPerformed(ActionEvent e) {
74                  JOptionPane.showMessageDialog(null, "Cadastro realizado com sucesso");
75              }
76          });
77          btnOK.setBounds(141, 202, 85, 21);
78          contentPane.add(btnOK);
79      }
80
81  }
```

Execute a Aplicação e teste.

Referências

SANTOS, Marcela Gonçalves dos. **Linguagem de programação**. SAGAH, 2018. ISBN digital: 9788595024984.

SEBESTA, Robert W. **Conceitos de Linguagens de Programação**. Bookman, 2018. ISBN digital: 9788582604694.

SCHILDT, Herbert. Java para iniciantes. Disponível em: Minha Biblioteca, (6th edição). Grupo A, 2015.

SILVA, Fabricio Machado da. **Paradigmas de programação**. SAGAH, 2019. ISBN digital: 9788533500426.