



Quem se prepara, não para.



# Linguagens de Programação

3º período

Prof. Dr. João Paulo Aramuni

# Sumário

- Java
  - Formatação
  - Primeiro projeto com o WindowBuilder

# Sumário

- Java
  - **Formatação**
  - Primeiro projeto com o WindowBuilder

- 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: "(31)98888-0000" ou o CEP: "30-100.200".
  - Antes de criarmos e utilizarmos um **JFormattedTextField** devemos criar um objeto **MaskFormatter** e configurar uma máscara.

- Formatando campos com JFormattedTextField

```
MaskFormatter mascaraCpf = new MaskFormatter("###.###.###-##");  
JFormattedTextField cpf = new JFormattedTextField(mascaraCpf);  
  
// Outros exemplos  
MaskFormatter mascaraCep = new MaskFormatter("#####-###");  
MaskFormatter mascaraTel = new MaskFormatter("(##) ####-####");
```

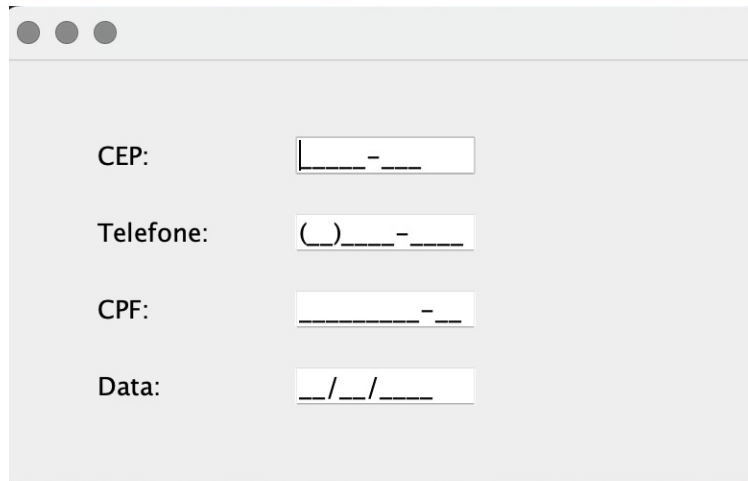
- Formatando campos com `JFormattedTextField`
  - Quando criamos um **MaskFormatter** podemos utilizar ao invés de “#”, outros caracteres, dependendo do tipo de restrição que desejamos implementar no **JFormattedTextField**.
  - Esses caracteres são definidos abaixo:
    - “#” indica que qualquer número 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 número (0-9 e a-z) poderá ser inserido;
    - “H” indica qualquer caractere hexadecimal (0-9 a-f) poderá ser inserido;
    - “\*” indica qualquer coisa, incluindo caracteres especiais poderão ser inseridos.

- Formatando campos com JFormattedTextField
  - 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	##/##/####



- Formatando campos com JFormattedTextField
  - Para exemplificar o uso do JFormattedTextField segue o exemplo abaixo, onde criamos quatro campos com máscaras, são eles: CEP, Telefone, CPF, Data.



A screenshot of a Java Swing window with a light gray background and a standard macOS-style title bar (three gray circles). The window contains four rows of labels and text input fields:

- CEP:** The input field has a mask of two digits, a hyphen, and three digits (e.g., "\_\_\_\_-\_\_\_\_").
- Telefone:** The input field has a mask of an opening parenthesis, a digit, a closing parenthesis, three digits, a hyphen, and four digits (e.g., "(\_\_\_\_)\_\_\_\_-\_\_\_\_").
- CPF:** The input field has a mask of three digits, a hyphen, two digits, a hyphen, and three digits (e.g., "\_\_\_\_-\_\_\_\_-\_\_\_\_").
- Data:** The input field has a mask of two digits, a slash, two digits, a slash, and four digits (e.g., "\_\_\_\_/\_\_\_\_/\_\_\_\_").

```
1 package meuapp;
2
3 import java.awt.Container;
4 import java.text.ParseException;
5
6 import javax.swing.JFormattedTextField;
7 import javax.swing.JFrame;
8 import javax.swing.JLabel;
9 import javax.swing.text.MaskFormatter;
10
11 public class TestandoJFormattedTextField extends JFrame {
12
13     // Método main
14     public static void main(String[] args){
15         new TestandoJFormattedTextField();
16     }
```

```
// Método construtor da classe
public TestandoJFormattedTextField() {
    Container janela = getContentPane();
    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;

    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);
    }

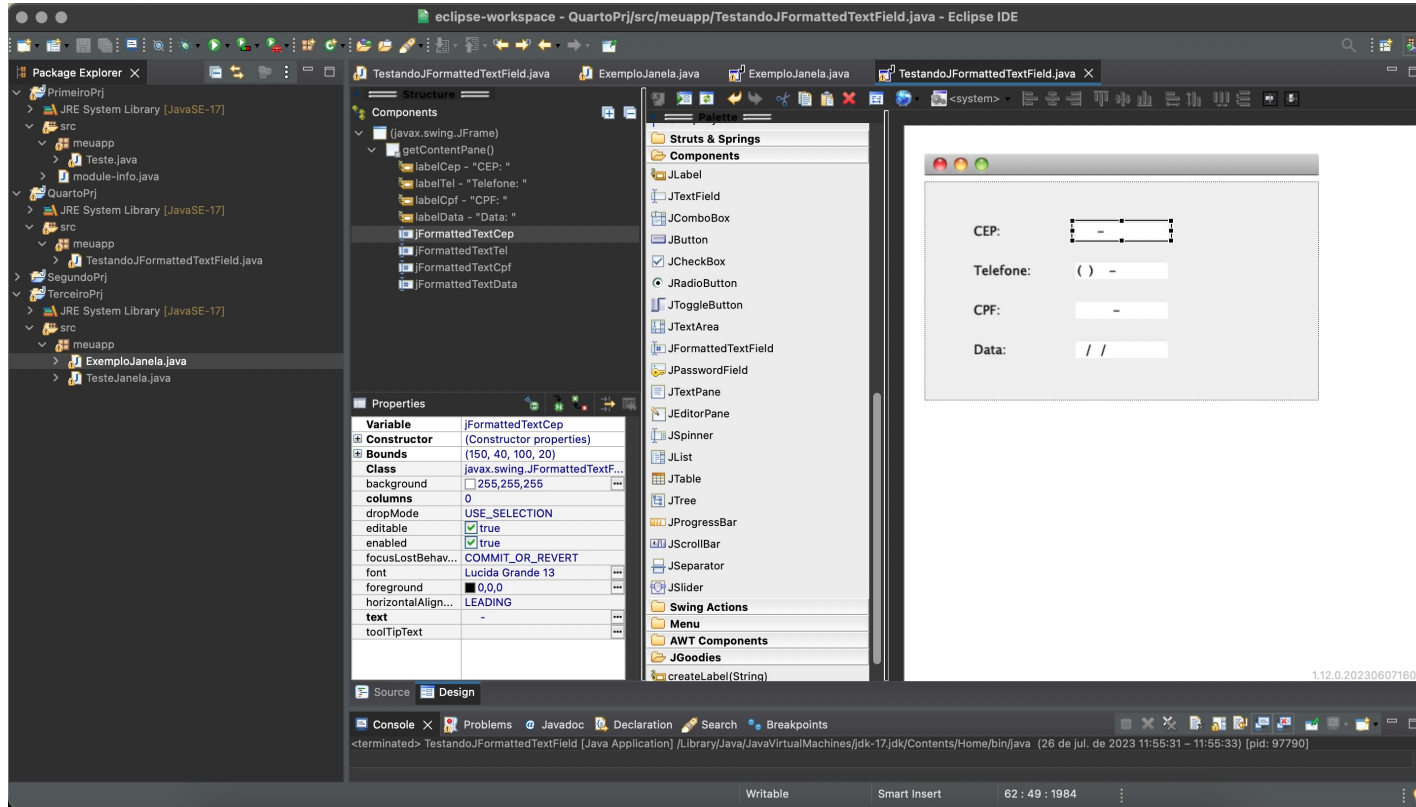
    // Seta as máscaras nos objetos JFormattedTextField
    JFormattedTextField jFormattedTextCep = new JFormattedTextField(mascaraCep);
    JFormattedTextField jFormattedTextTel = new JFormattedTextField(mascaraTel);
    JFormattedTextField jFormattedTextCpf = new JFormattedTextField(mascaraCpf);
    JFormattedTextField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

```
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);
setLocationRelativeTo(null);
setVisible(true);

}

}
```

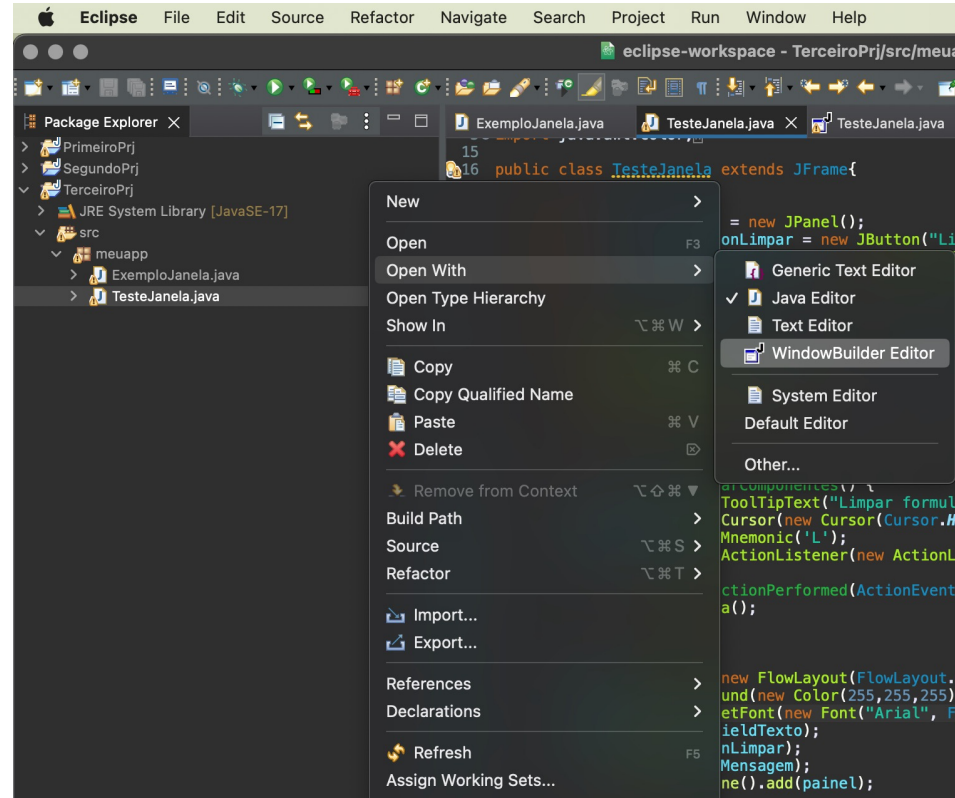


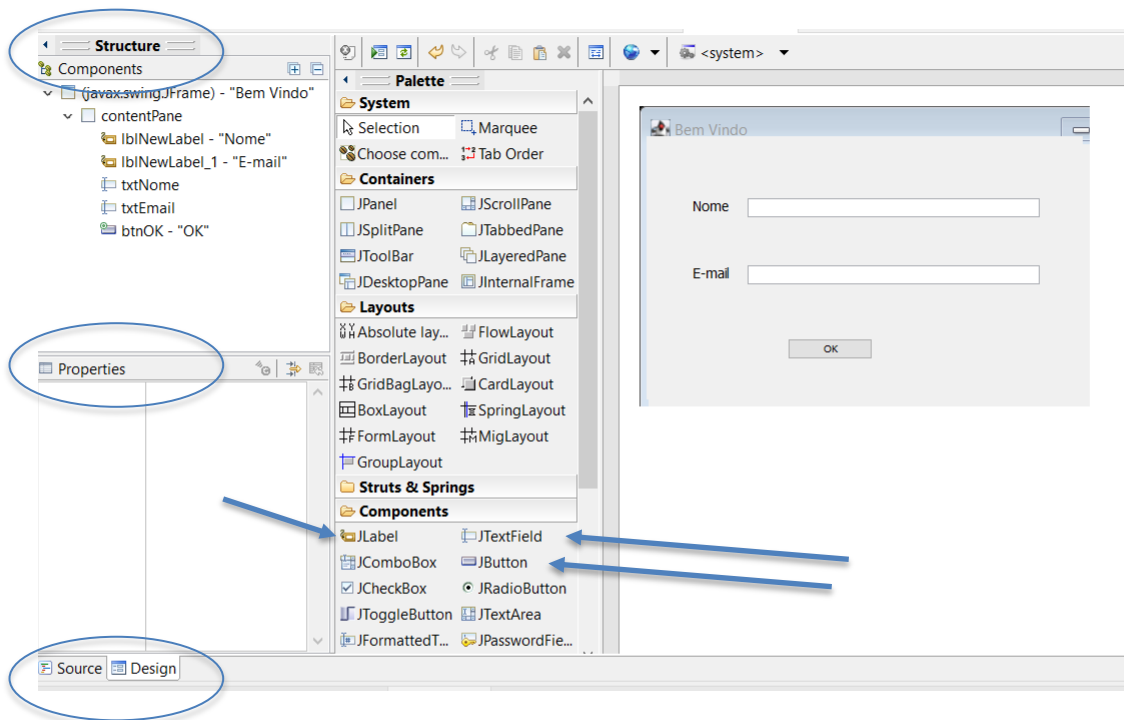
# Sumário

- Java
  - Formatação
  - **Primeiro projeto com o WindowBuilder**

# Java

- Crie um projeto chamado WindowBuilderPrj, um pacote meuapp e uma classe chamada Tela.java.
- Depois clique com o botão direito na classe e abra com o WindowBuilder Editor.



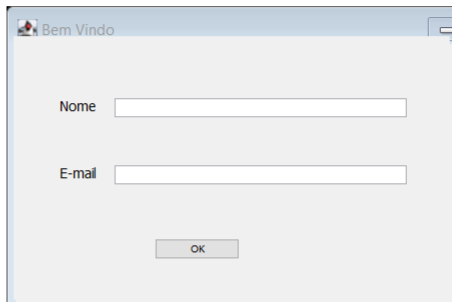


Clique na opção **Design** que será mostrada a interface do Window Builder.

- 1) Adicione o “**Absolute Layout**” que permitirá criar um **contentPane**.
- 2) Acione os elementos **JLabel** e **JTextField** para compor a tela ao lado.

Todos os elementos possuem propriedades que podem ser alteradas, basta selecionar o objeto que as propriedades serão exibidas no menu lateral.





### 3) Adicione um elemento **JButton**.

Clique duas vezes no botão, será aberto o código fonte já com o **evento de click** do botão pronto, vamos colocar uma mensagem na tela:

```
jButtonOk.addActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent e) {  
        /*  
        * No lugar do valor null no método JOptionPane.showMessageDialog(), você pode  
        * passar um componente específico como o "pai" da janela de diálogo. Isso fará  
        * com que a caixa de diálogo apareça no centro do componente pai, em vez de no  
        * centro da tela. Além disso, você pode usar diferentes tipos de componentes  
        * para personalizar a aparência da janela de diálogo.  
        */  
        JOptionPane.showMessageDialog(null, "Cadastro realizado com sucesso.");  
    }  
});
```

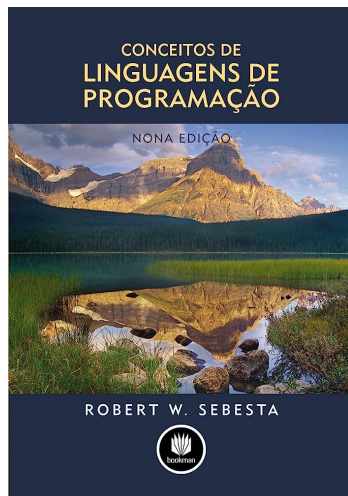
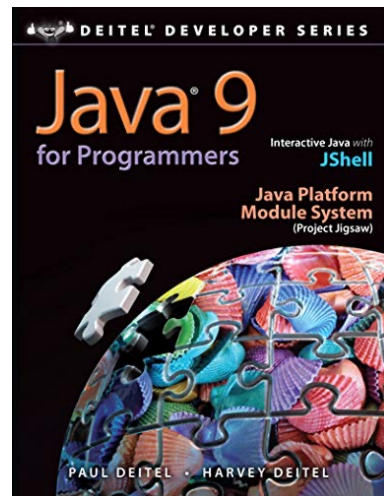
- Primeiro projeto com o WindowBuilder
  - 4) Agora crie um botão de **Limpar**, ao lado do botão OK, e o evento para limpar os campos Nome e Email.

```
jButtonLimpar.addActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent e) {  
        limparTela();  
    }  
});
```

- Primeiro projeto com o WindowBuilder
  - 5) Por fim, crie os campos CEP, Telefone, CPF e Data, formatados por meio da classe **MaskFormatter**.

```
MaskFormatter mascaraCep = new MaskFormatter("#####-###");  
MaskFormatter mascaraTel = new MaskFormatter("(##)####-####");  
MaskFormatter mascaraCpf = new MaskFormatter("#####-##");  
MaskFormatter mascaraData = new MaskFormatter("##/##/####");
```

# Referências





Obrigado!