Programação Orientada e Objetos II

Anhanguera


AVALIE
SUA PROFISSÃO
QUANDO APARECER EM SEU PORTAL UMA AVALIAÇÃO SOBRE SEU CURSO, RESPONDA:
NOTAS
9 OU 10
SIGNIFICA QUE VOCÊ INDICA
NOTAS
7 OU 8
SIGNIFICA QUE VOCÊ NÃO INDICA
Anhanguera

Anhanguera

Ementa

Anhanguera

Unidade 1 - Programação orientada a eventos com interfaces gráfi cas e banco de dados relacional

Seção 1.1 Desenvolvimento de interfaces gráficas na linguagem Java

Seção 1.2 Programação em Java usando orientação a eventos

Seção 1.3 Programação em Java usando banco de dados relacional

Unidade 2- Programação concorrente orientada a objetos

Seção 2.1 Programação em Java usando threads

Seção 2.2 Definição e tratamento de exceções para sistemas com threads

Seção 2.3 Programação em Java utilizando elementos para sincronização em Java

Ementa

Anhanguera

Unidade 3 - Padrões de projeto, ferramentas e métodos ágeis

Seção 3.1 Ferramentas para programação em linguagens orientadas a objetos

Seção 3.2 Padrões de projetos em orientação a objetos

Seção 3.3 Métodos ágeis em orientação a objetos

Unidade 4 - Novas tecnologias para programação em banco de dados

Seção 4.1 Banco de dados NoSQL

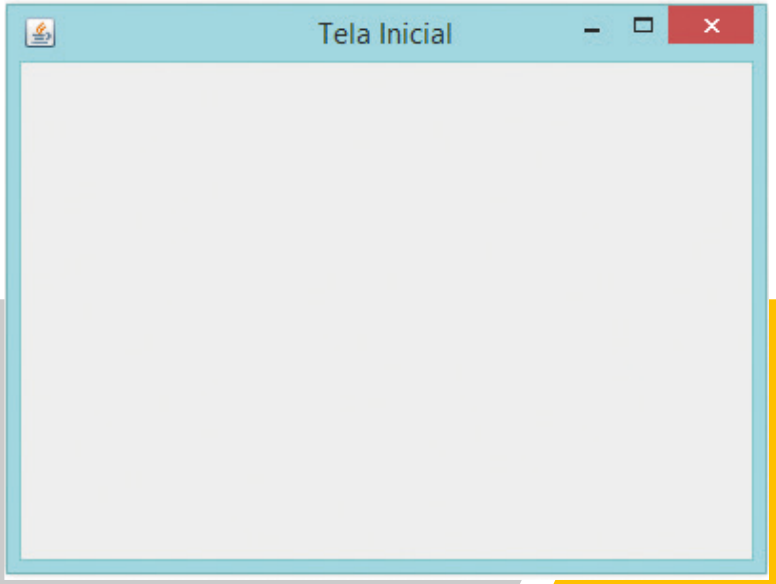
Seção 4.2 Introdução ao desenvolvimento em Java usando MongoDB

Seção 4.3 Desenvolvimento em Java usando MongoDB

Desenvolvimento de interfaces gráficas na linguagem Java

A criação de interfaces gráficas, também chamado de Graphical User Interface (GUI), é um processo que necessita de grande dedicação, pois por meio delas o usuário percebe e interage com o sistema. Dessa forma, em todo o processo de criação de uma interface é necessário seguir as orientações de boas práticas na utilização de componentes, inclusive quanto ao posicionamento e agrupamento (DEITEL; DEITEL, 2016). Portanto, nessa etapa de estudo você aprenderá sobre a criação inicial de uma interface gráfica utilizando as bibliotecas do Java.

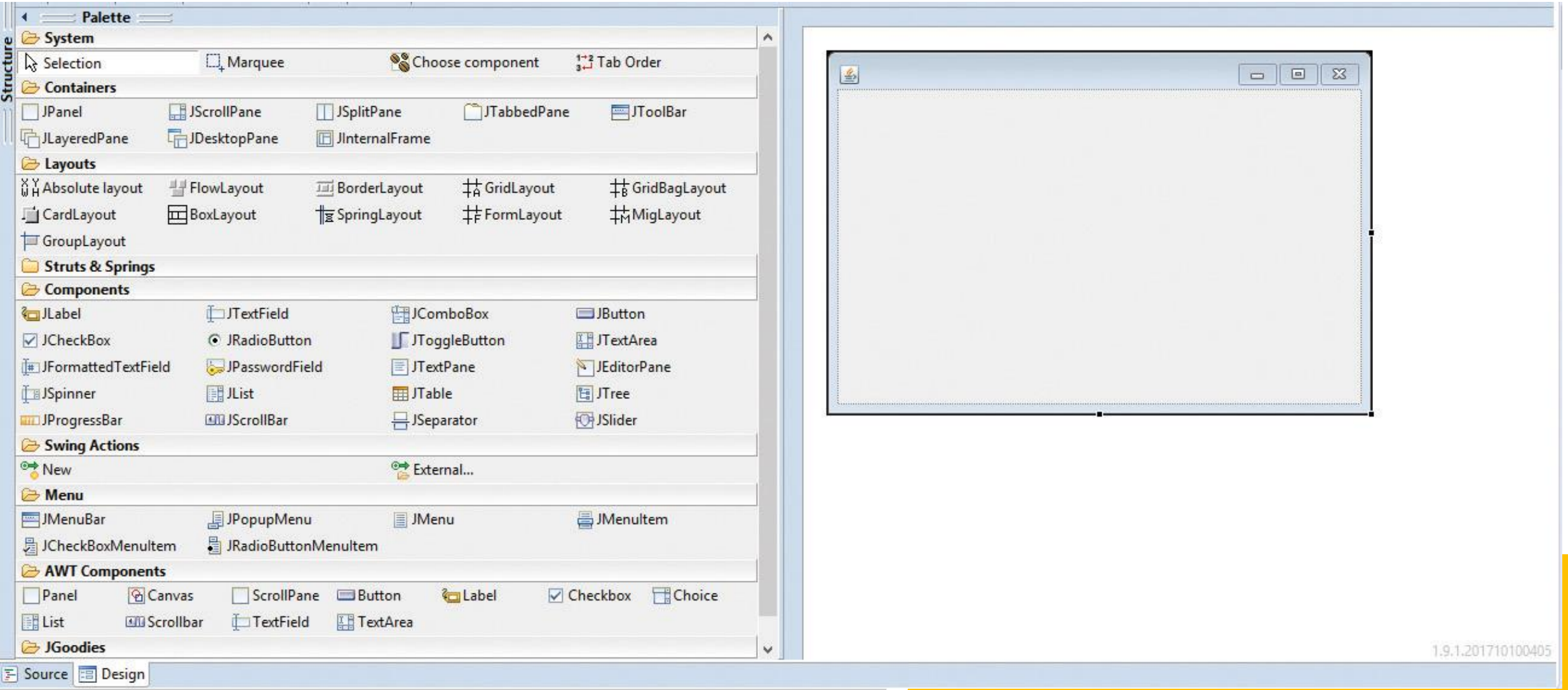
No caso do Java, utilizaremos o Swing (FURGERI, 2015). O Swing é uma biblioteca que vem incorporada no Java Development Kit, dispondo de diversos elementos para a produção dessas telas (HORSTMANN, 2016). O primeiro elemento a ser utilizado em uma interface gráfica é a representação de uma área que apresenta uma barra de título e um espaço reservado para se adicionar componentes. No caso do Java com Swing, a classe que faz essa representação é o JFrame (MANZANO; COSTA, 2014). A Figura 1.1 exibe um exemplo da interface criada por essa classe.



Na Figura 1.1 é possível reparar uma barra no topo da tela com o texto “Tela Inicial”, os botões para minimizar, maximizar e fechar e uma área cinza para adicionar os componentes. Todos esses elementos de tela são passíveis de alteração e todas essas modificações podem ser feitas por código em Java sem a necessidade de utilizar algum editor gráfico, todavia, existem editores para a criação de telas utilizando o Swing. No caso do Integrated Development Environment (IDE) Eclipse, existem plugins e extensões que propiciam esse editor (WINDER, 2009).

Existem editores “what you see is what you get” (WYSIWYG) para a produção de interfaces gráficas. Esses editores apresentam uma interface gráfica na qual existe a possibilidade de selecionar um componente gráfico com o mouse e arrastar para a posição desejada na interface gráfica que está sendo montada. Esse tipo de editor propicia um dinamismo maior na produção de uma interface, porém é necessário atentar-se ao fato de que ao fazer a simples ação de colocar um botão em uma interface, inicia-se um processo automático de criação de código pelo editor. Com isso, como é possível mensurar os ganhos de tempo comparados à possibilidade de manutenção mais complexa no código?

Um dos editores gráficos acoplados ao Eclipse é o Windows Builder, que fornece diversas ferramentas e formas de criar a janela (WINDER, 2009). A Figura 1.2 apresenta uma visão desse editor, no qual os itens do lado esquerdo, “Palette”, são todos os componentes disponíveis para inserção na tela, e a direita é a representação da própria tela que receberá os componentes. Uma das grandes vantagens é a capacidade de, apenas com o cursor do mouse, fazer a inclusão de componentes na tela, o que proporciona aumento de velocidade no desenvolvimento.

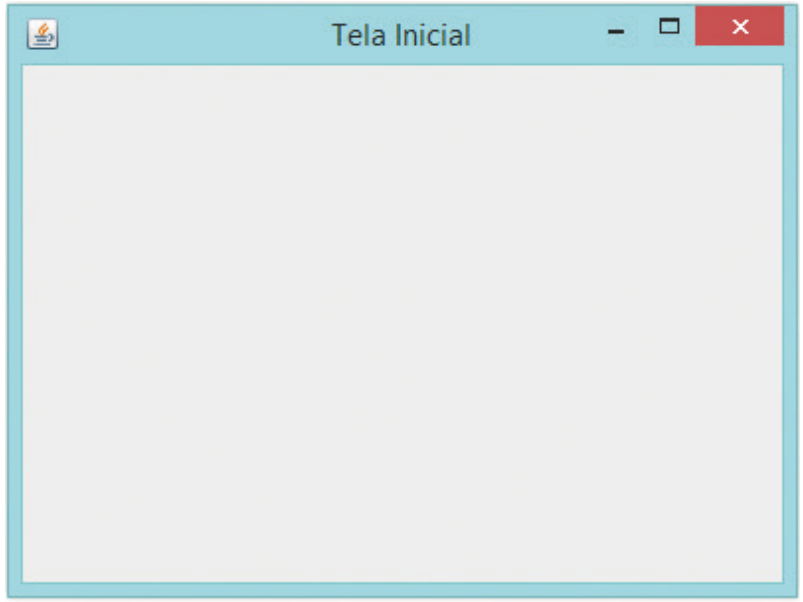

Palette
System
Selection
Marquee
Choose component
Tab Order
Containers
JPanel
JScrollPane
JSplitPane
JTabbedPane
JToolBar
JLayeredPane
JDesktopPane
JInternalFrame
Layouts
Absolute layout
FlowLayout
BorderLayout
GridLayout
GridBagLayout
CardLayout
BoxLayout
SpringLayout
FormLayout
MigLayout
GroupLayout
Struts & Springs
Components
JLabel
JTextField
JComboBox
JButton
JCheckBox
JRadioButton
JToggleButton
JTextArea
JFormattedTextField
JPasswordField
JTextPane
JEditorPane
JSpinner
JList
JTable
JTree
JProgressBar
JScrollBar
JSeparator
JSlider
Swing Actions
New
External...
Menu
JMenuBar
JPopupMenu
JMenu
JMenuItem
JCheckBoxMenuItem
JRadioButtonMenuItem
AWT Components
Panel
Canvas
ScrollPane
Button
Label
Checkbox
Choice
List
Scrollbar
TextField
TextArea
JGoodies
Source
Design
1.9.1.201710100405

https://www.online-java.com/

https://eclipse.dev/windowbuilder/

Anhanguera

```
1. package view;
2. import javax.swing.JFrame;

3. public class InterfaceGrafica extends JFrame{
4.      public InterfaceGrafica(){
5.              setSize(400,500);
6.              setTitle(“Tela Inicial”);
7.              setVisible(true);
8.      }
9.      public static void main(String[] args) {
10.             InterfaceGrafica telaInicial = new InterfaceGrafica();
11.     }
12. }
```



A linha 1 do Quadro 1.1 faz referência ao pacote (conjunto de classes que têm funções semelhantes). Sobre os package é sempre interessante manter as classes de interface no mesmo pacote para manter a alta coesão e baixo acoplamento. A coesão demonstra que as classes, pacotes e métodos têm funções delimitadas, relacionando apenas a um propósito, e o alto acoplamento refere-se ao funcionamento independente de outras classes para executar sua função (DEITEL; DEITEL, 2016). A linha 2 apresenta quais pacotes terão de ser incluídos no seu código para que se possa utilizar as classes do Swing; nesse caso foi necessário incluir somente a JFrame, classe à qual pertence a tela da Figura 1.1. A linha 3 apresenta a classe que foi criada para representar a tela, sendo que ela especializa (extends) a classe JFrame. A linha 4 é o construtor da classe InterfaceGrafica, no qual são feitas as configurações da tela que é criada nessa classe. O comando setSize(), na linha 5, define a largura e altura da janela, o comando setTitle() define o título da tela e, por fim, o comando setVisible() define que
a tela é visível. As linhas de 9 a 11 são as funções iniciais (método main) do programa que utiliza a classe que representa a tela.

Para se criar uma interface gráfica são necessários cinco passos (HORSTMANN, 2016, DEITEL; DEITEL, 2016):

1. Criar uma relação de especialização com a classe que representa sua tela.

2. Declarar como atributos os elementos que serão adicionados à tela.

3. Definir a forma de alocação dos elementos gráficos na tela.

4. No construtor, instanciar, configurar e posicionar os itens na tela.

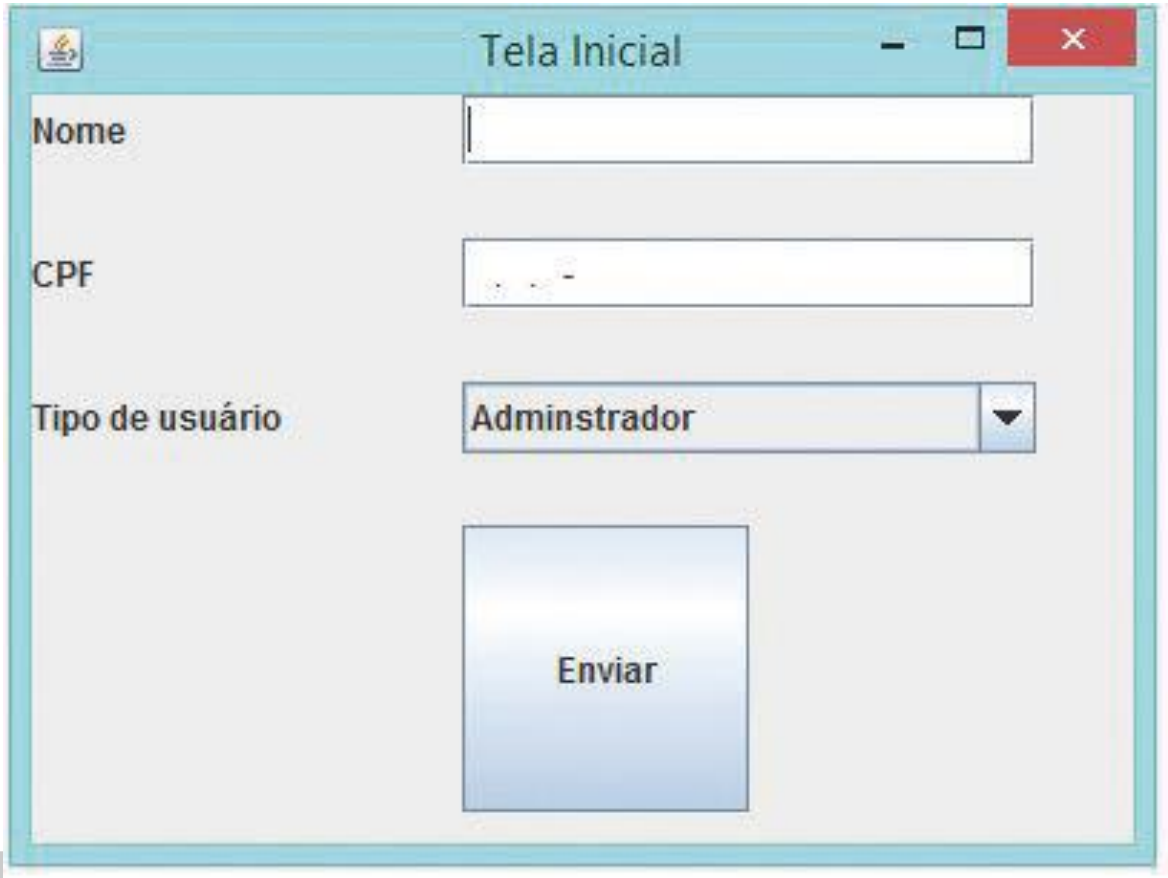
5. Tratar os eventos dos componentes para tratar as ações do usuário com a interface gráfica.

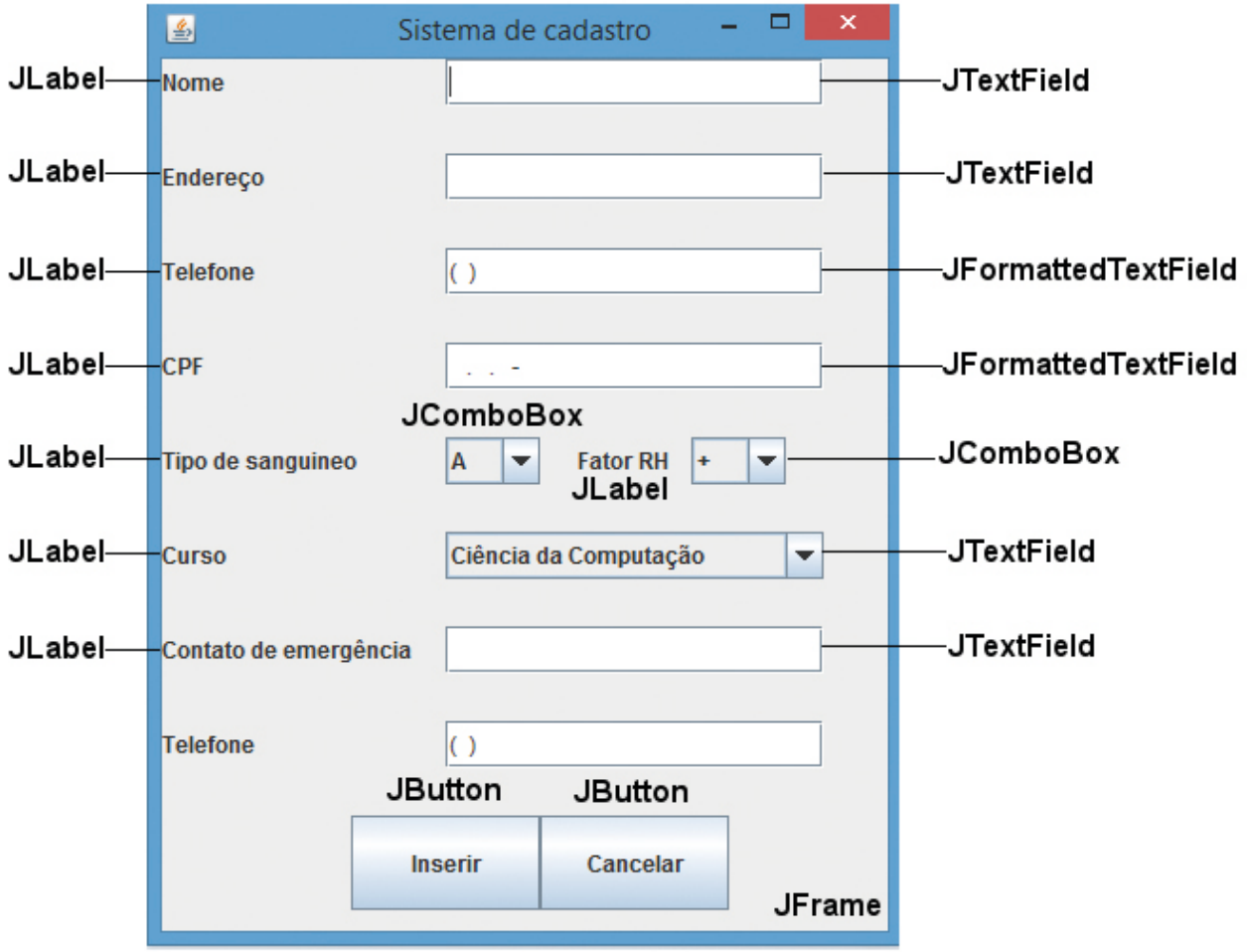
```
1. package view;
2. import javax.swing.JFrame;
3. import javax.swing.JLabel;
4. import javax.swing.JTextField;
5.       public class PrimeiraTela extends JFrame {
6.                private JLabel lblNome;
7.                private JTextField txtNome;
8.       public PrimeiraTela() {
9.                lblNome = new JLabel("Nome");
10.               txtNome = new JTextField();
11.               setSize(400,200);
12.               setTitle("Tela Inicial");
13.               setVisible(true);
14.               setLayout(null);
15.               lblNome.setBounds(10,10,100,25);
16.               txtNome.setBounds(50,10,200,25);

17.               getContentPane().add(lblNome);
18.               getContentPane().add(txtNome);
19.               }
20.      public static void main(String[] args) {
21.               PrimeiraTela t1 = new PrimeiraTela();
22.               }
23.      }
```

Componente	Descrição
`JButton`	Objeto usado para criar botões.
`JCheckBox`	Objeto usado para oferecer uma opção para o usuário. Normalmente é representado por uma caixa de seleção, que quando está com "check" representa "sim" e quando está com "não" representa "não".
`JComboBox`	Objeto usado para oferecer mais de uma opção para o usuário em forma de lista *drop-down*. Para cada lista somente um item pode ser selecionado.
`JList`	Objeto usado para oferecer mais de uma opção para o usuário, mas, diferentemente do JComboBox, esse componente permite a seleção de mais que uma opção.
`JPanel`	Objeto usado para organizar diversos componentes.

Para adicionar um frame foi preciso incluir a biblioteca javax.swing. JFrame, para adicionar um label foi necessário importar javax.swing. JLabel, e para adicionar uma caixa de texto foi necessário importar javax.swing.JTextField. Diante disso, para cada componente será necessário incluir uma biblioteca? E se houver dez, vinte ou mais componentes diferentes serão necessárias todas as linhas de inclusão?

Anhanguera




Sistema de cadastro
JLabel
Nome
JTextField
JLabel
Endereço
JTextField
JLabel
Telefone
( )
JFormattedTextField
JLabel
CPF
. . -
JFormattedTextField
JComboBox
JLabel
Tipo de sanguineo
A
Fator RH
+
JComboBox
JLabel
Curso
Ciência da Computação
JTextField
JLabel
Contato de emergência
JTextField
Telefone
( )
JButton
JButton
Inserir
Cancelar
JFrame

Anhanguera

Página 20 e 21 do livro

A interface gráfica representa o primeiro contato do usuário com o sistema. Essa interface garante que o usuário utilize todos os aspectos que o sistema pode fornecer, e no caso do Java é possível empregar diversos componentes para fazer a construção das janelas de interface.

Dos componentes descritos a seguir, quais deles pertencem ao Java Swing?

a) JButton, JFrame e JText.

b) JTextField, JSwing e JFrame.

c) JFrame, JButton e JTextField.

d) Container, JButton e JWindowsP.

e) JFrame, JButton, JTextField, JMask.

Existem diversos elementos visuais no Java Swing, e cada um deles apresenta funções bem definidas, que podem ser utilizados diversas vezes na mesma interface. Alguns elementos têm funções semelhantes e alguns aspectos peculiares de configuração e parâmetros.

Um campo de texto deve receber apenas letras, tanto em caixa baixa como em caixa alta, na quantidade máxima de 10 letras. Qual componente deve ser utilizado?

a) JFormattedTextField.

b) JTextField.

c) JTextArea.

d) JComboBox.

e) JText.