



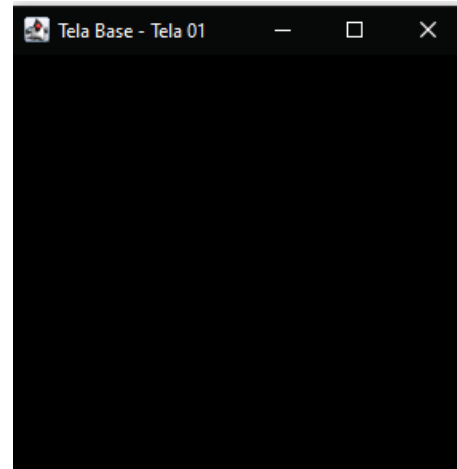
UNIDADE I

Aplicações de Linguagem de Programação Orientadas a Objetos

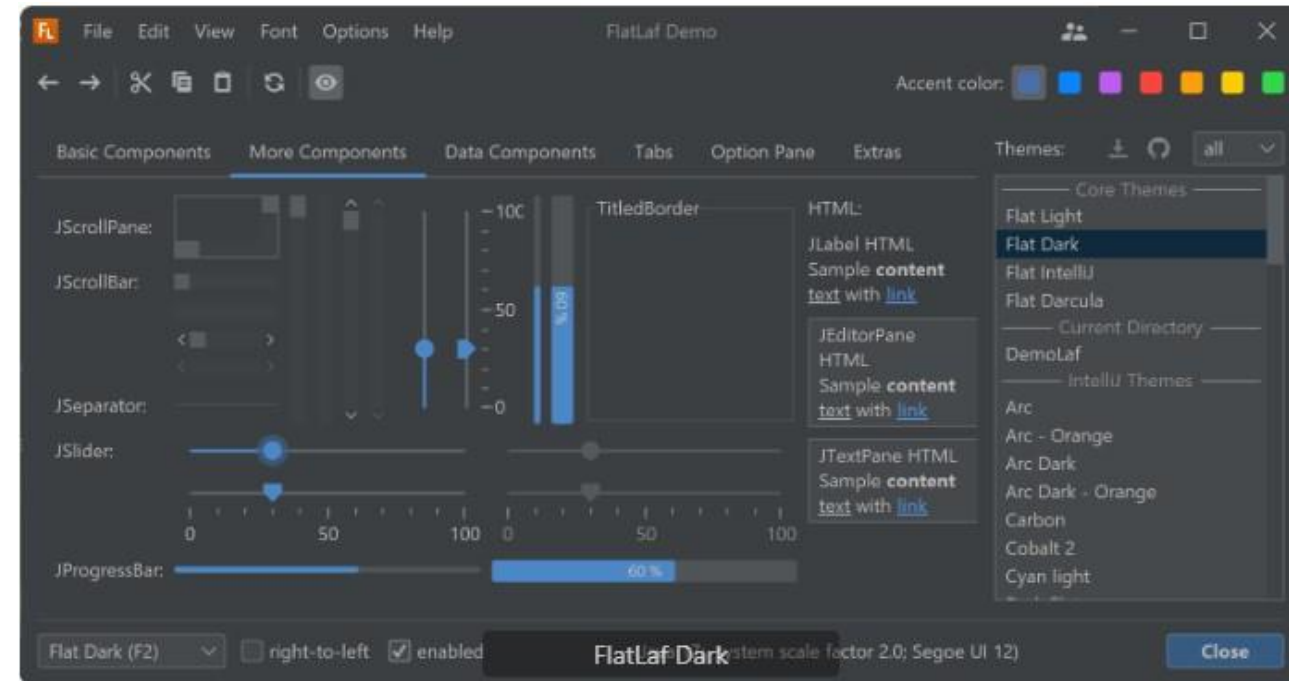
Prof. Me. Lauro Tomiatti

Graphic User Interface

- O que é interface?
- *Graphic User Interface.*



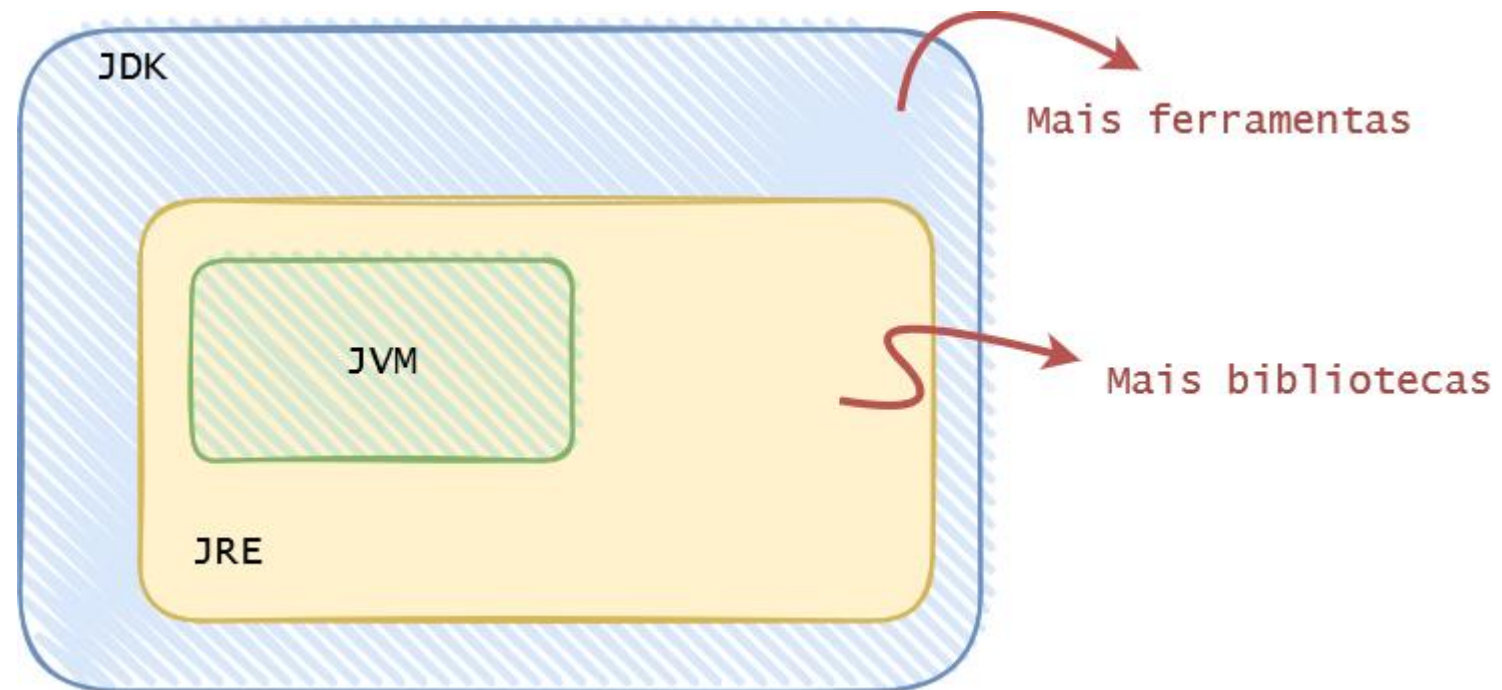
Fonte: Acervo do autor.



Fonte: <https://www.formdev.com/flatlaf/>

Rodando sem GUI

- Como funciona o Java?
- JDK, JRE, JVM.
- 8 bilhões de dispositivos com Java.



Fonte: Autoria própria.

Compilar

- O que é compilar?
- Rodar pela IDE ou pelo terminal.
- Compilando o projeto.
- Rodar através do terminal.

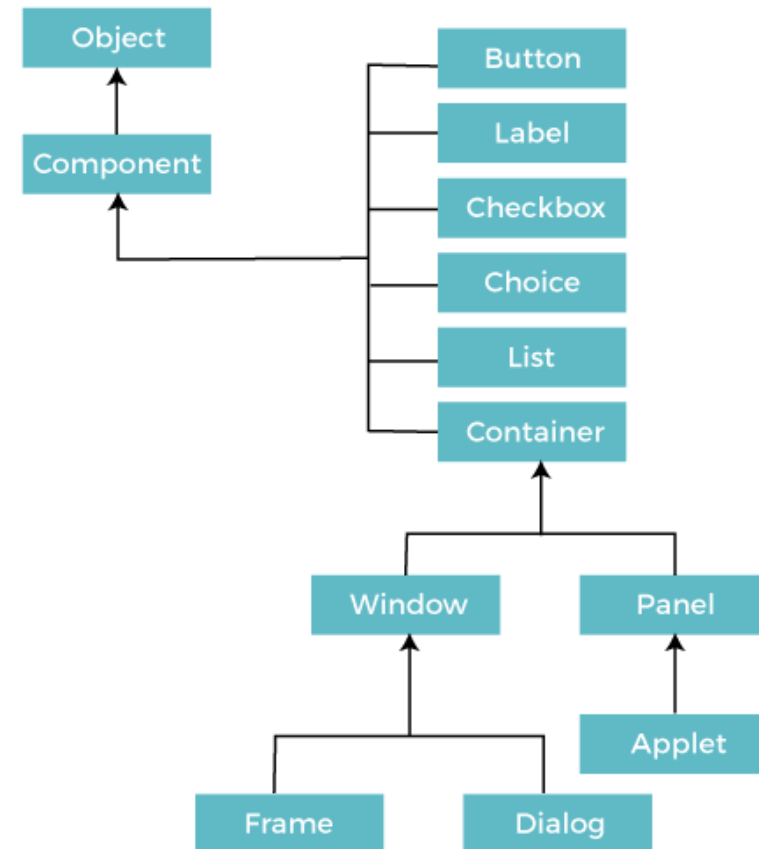
```
javac arquivo.java
```

```
java arquivo
```

Fonte: Autoria própria.

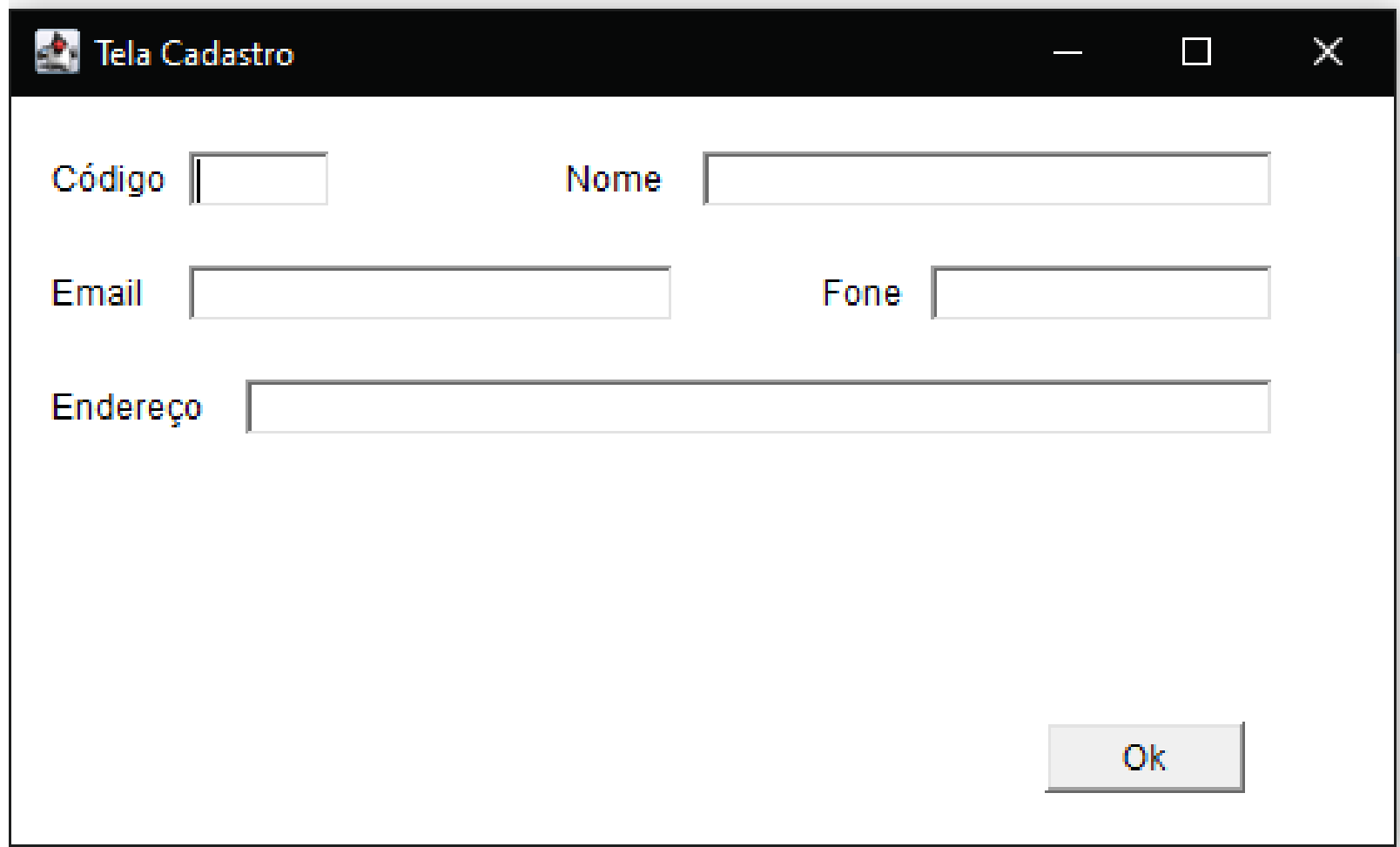
Programação Orientada a Objetos

- AWT (*Abstract Window Toolkit*) é uma API para criar aplicativos GUI em Java. É uma estrutura dependente da plataforma, ou seja, os componentes GUI pertencentes ao AWT não são os mesmos em todas as plataformas. De acordo com a aparência nativa da plataforma, a aparência dos componentes do AWT também muda.
- Conjunto de sub-rotinas.
- Foi a base para o *Java Swing*.



AWT

- Aplicação *Desktop*.
- Aplicações *Web*.
- Aplicações *Mobile*.



The image shows a screenshot of a Java AWT window titled "Tela Cadastro". The window has a standard title bar with a close button (X), a maximize button (square), and a minimize button (dash). The main content area contains five text input fields arranged in three rows. The first row has "Código" and "Nome". The second row has "Email" and "Fone". The third row has "Endereço". All fields are empty. An "Ok" button is located in the bottom right corner of the window.

Código	<input type="text"/>	Nome	<input type="text"/>
Email	<input type="text"/>	Fone	<input type="text"/>
Endereço	<input type="text"/>		

Ok

Fonte: Acervo do autor.

Componentes

- *Frame* ou janela – uma janela com seus elementos básicos (bordas, espaçamento, barra, os botões do sistema).
- *Panel* ou painel – como uma janela, mas apenas com seus componentes internos (não possui a barra nem os botões do sistema).
- Ambos são *containers*.
- O *panel* é colocado dentro do *frame*.

Componentes

- *Label* ou rótulo – todo texto que antecede um campo de texto.
- Utilizamos os rótulos para indicar para o usuário do que se trata aquele componente; podemos pedir por uma informação, indicar qual mapeamento da aplicação que ele está ou até mesmo criar definições para imagens.

Usuário:

Senha:

Fonte: Autoria própria.

Componentes

- *TextField* ou campo de texto – simples, utilizado para adquirir informações do usuário.
- Os campos de texto atualmente são maiores, mais largos e tentam acompanhar o reconhecimento cognitivo das aplicações Mobile.



Fonte: Autoria própria.

Componentes

- *Button* ou botão – forma de interação com o usuário para acionamento de algum evento.
- Os botões costumam seguir o *design* da empresa e expressar sua visão, missão e valores.



Fonte: Autoria própria.

Interatividade

Se podemos criar janelas normalmente e inserir nossos componentes padrões nela, para que utilizar painéis?

- a) Para criar novas janelas menores que podem ser usadas de aviso.
- b) Para dividirmos nossa janela em frações menores ou mudarmos a tela.
- c) Para gastar menos recurso dos computadores.
- d) Para criar contraste em nossa aplicação.
- e) Para inserir imagens.

Resposta

Se podemos criar janelas normalmente e inserir nossos componentes padrões nela, para que utilizar painéis?

- a) Para criar novas janelas menores que podem ser usadas de aviso.
- b) Para dividirmos nossa janela em frações menores ou mudarmos a tela.
- c) Para gastar menos recurso dos computadores.
- d) Para criar contraste em nossa aplicação.
- e) Para inserir imagens.

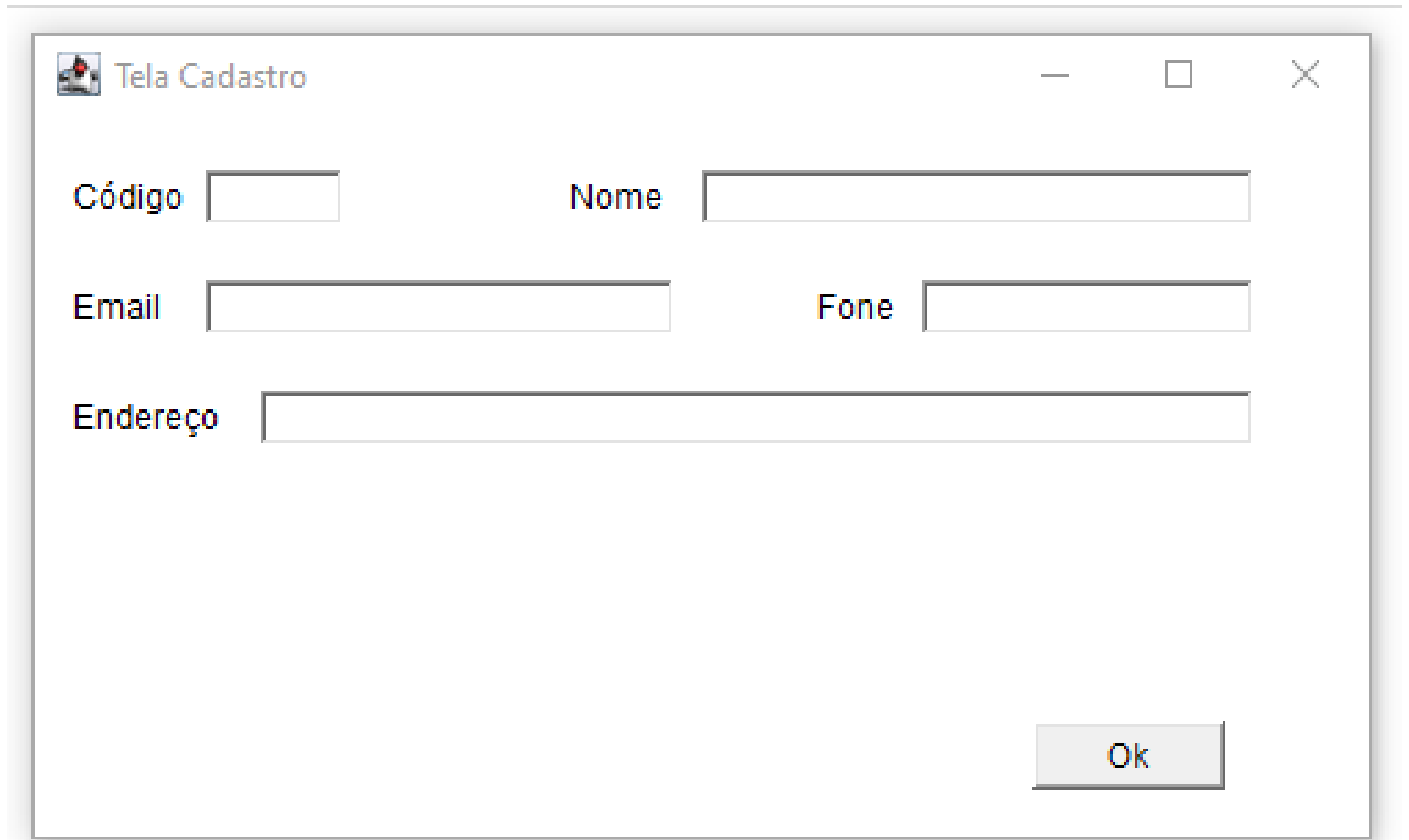
A constante evolução da tecnologia

- O desenvolvimento de *software* está em constante mudança, seja em *design* ou em mudança de pensamento ou ferramentas.
- Martin Fowler é um autor, palestrante, consultor e autointitulado falastrão geral sobre desenvolvimento de *software*. Em seu site pessoal, ele explica como o desenvolvimento de *software* é uma profissão jovem, e ainda estamos aprendendo as técnicas e criando as ferramentas para fazê-la de modo efetivo.

Java *Swing*

- Uma biblioteca com componentes específicos, com a aparência advinda do próprio Java.
- Diferente do AWT, ele não utiliza apenas componentes nativos do sistema operacional, sendo assim, os componentes não devem divergir entre os dispositivos.

Análise comparativa - AWT



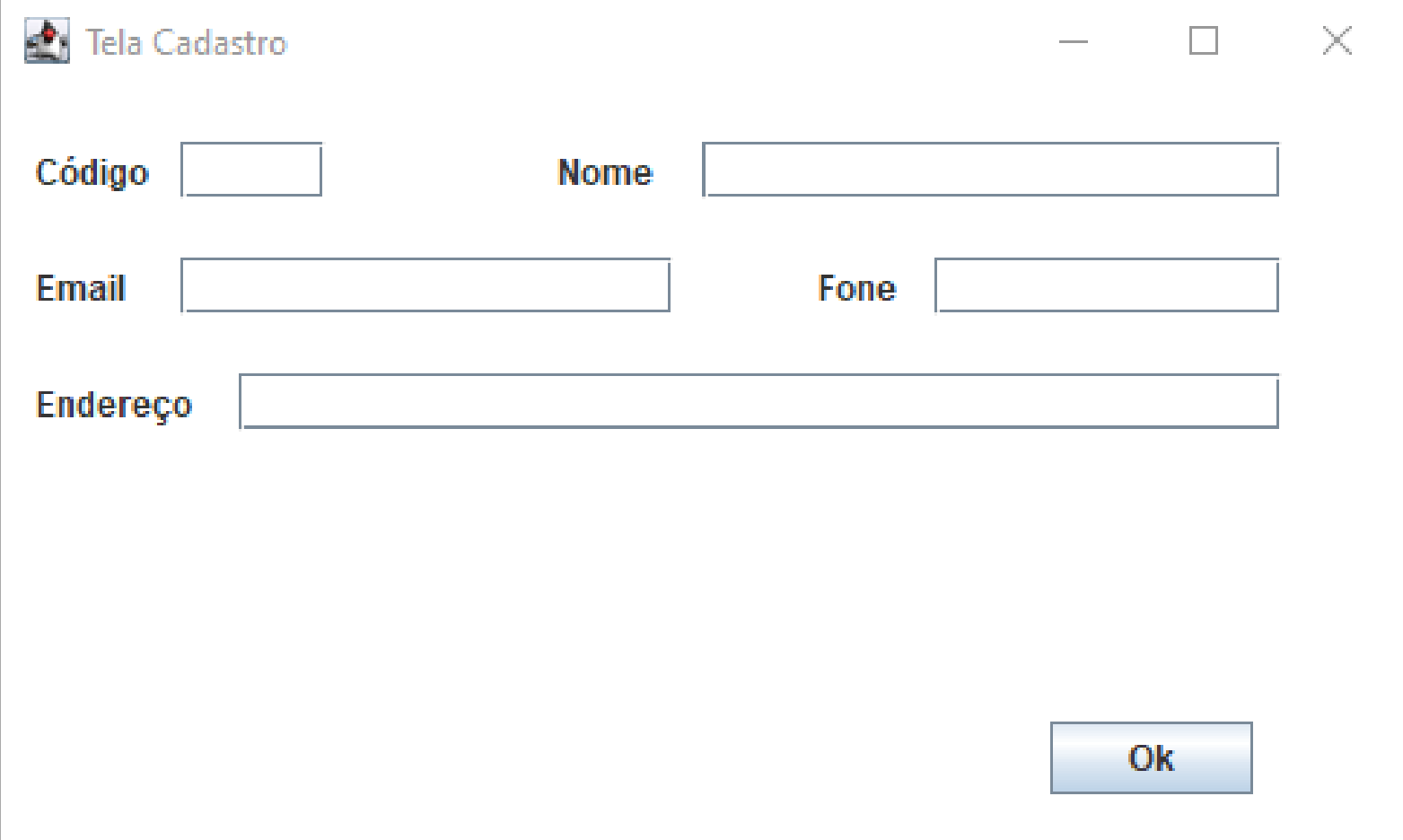
The image shows a screenshot of a Java AWT window titled "Tela Cadastro". The window has a standard title bar with a minimize button, a maximize button, and a close button. Inside the window, there are five text input fields arranged in three rows. The first row contains "Código" and "Nome". The second row contains "Email" and "Fone". The third row contains "Endereço". All fields are empty. At the bottom right of the window, there is an "Ok" button.

Código	<input type="text"/>	Nome	<input type="text"/>
Email	<input type="text"/>	Fone	<input type="text"/>
Endereço	<input type="text"/>		

Ok

Fonte: Autoria própria.

Análise comparativa - Swing



A Swing window titled "Tela Cadastro" with a standard Windows-style title bar (minimize, maximize, close buttons). The window contains five text input fields and one button. The fields are arranged in three rows: the first row has "Código" and "Nome"; the second row has "Email" and "Fone"; the third row has "Endereço". The "Ok" button is located at the bottom right of the window.

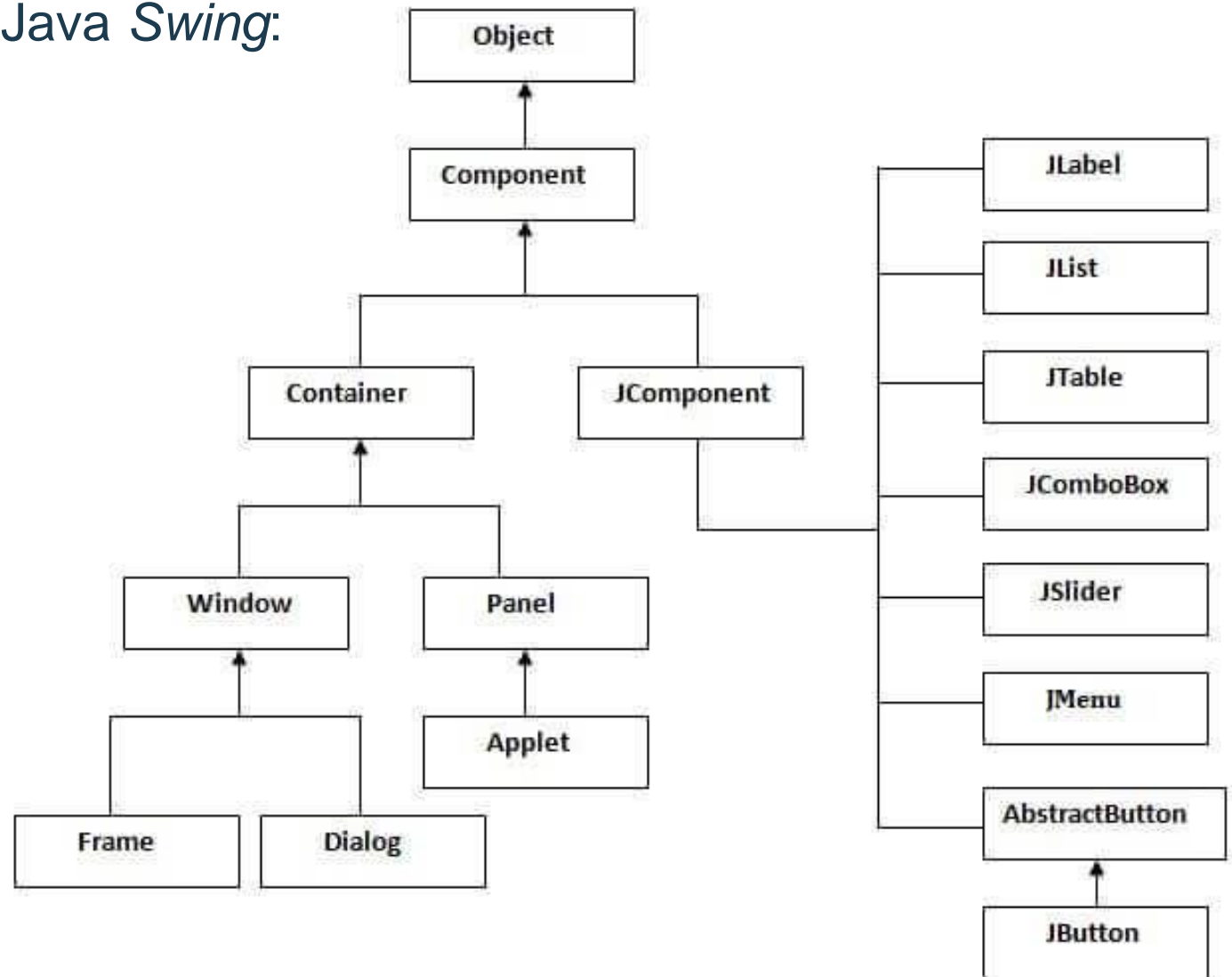
Código	<input type="text"/>	Nome	<input type="text"/>
Email	<input type="text"/>	Fone	<input type="text"/>
Endereço	<input type="text"/>		

Ok

Fonte: Autoria própria.

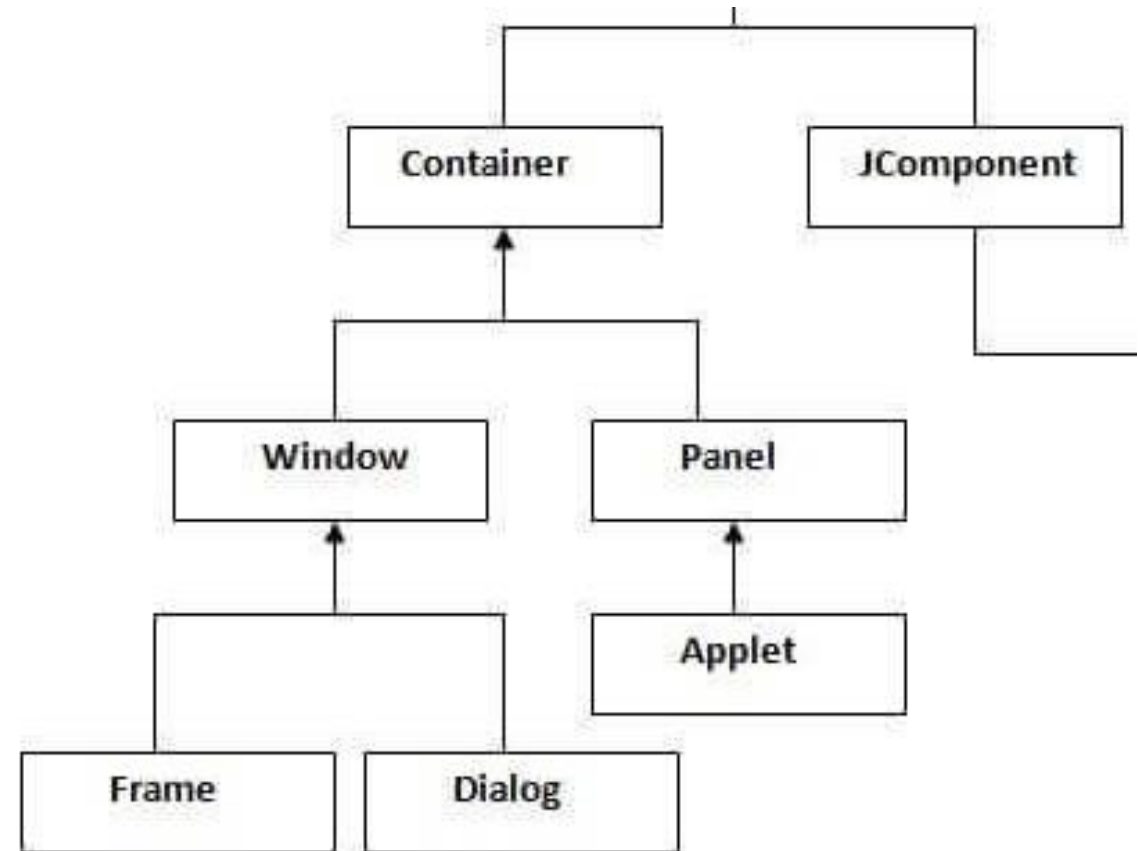
Estrutura do Swing

- Essas são as classes pertencentes ao Java *Swing*:



E sobre os contêineres?

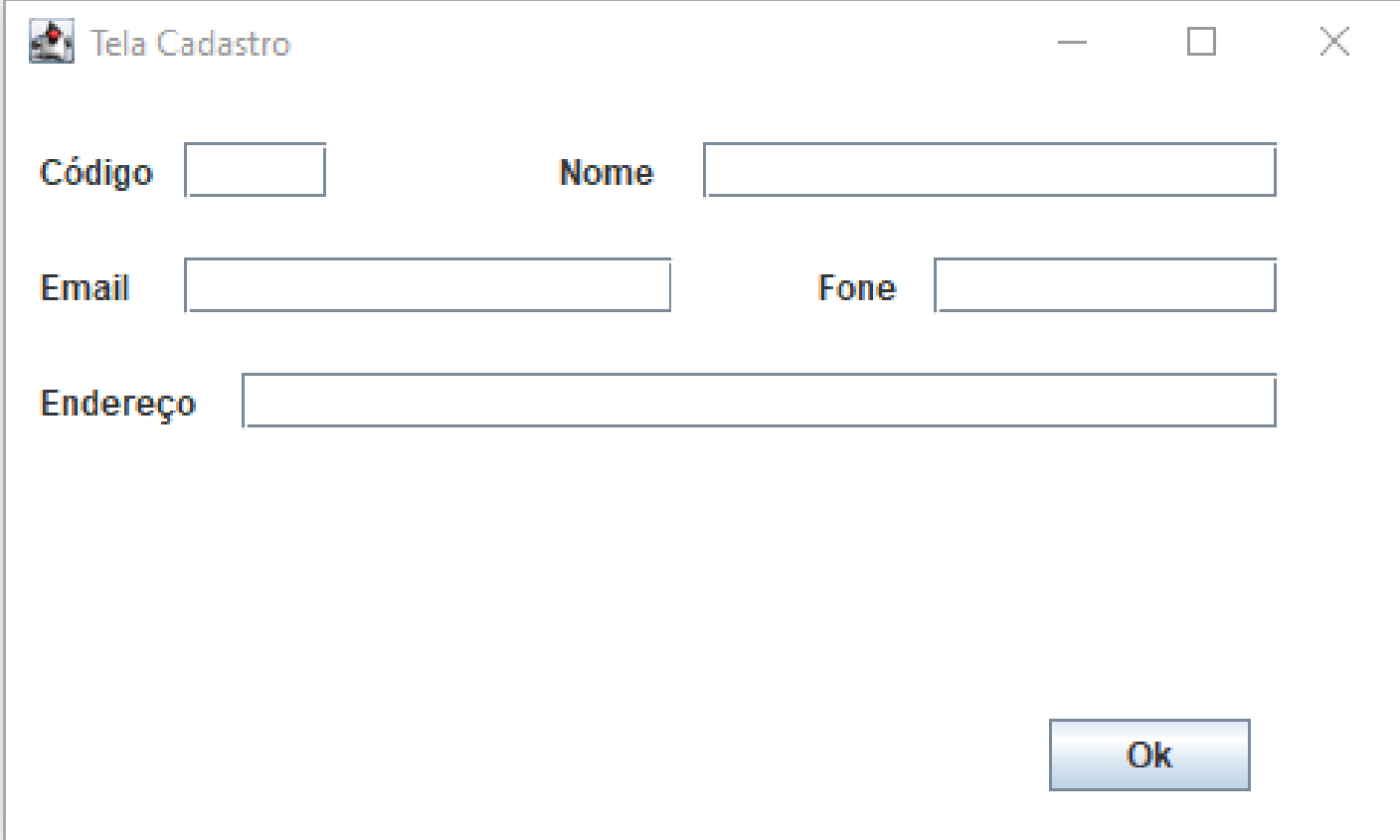
- Se for um pouco metodoso, é possível notar a árvore de componentes de uso como o *container*.



Trabalhando com os objetos

O que podemos alterar em nossos componentes?

- Atributos e métodos.
- *setForeground(...)*.



The image shows a Java Swing window titled "Tela Cadastro". It contains five text input fields arranged in three rows: "Código" and "Nome" in the first row, "Email" and "Fone" in the second row, and "Endereço" in the third row. An "Ok" button is located at the bottom right of the window.

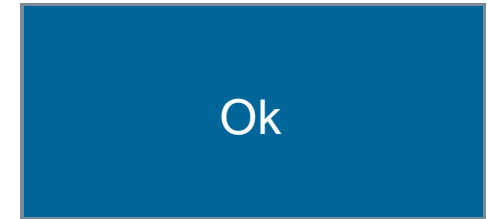
Fonte: Autoria própria.

O que é possível mudar?

- Posição no componente pai.
- Fonte.
- Cor da fonte.
- Cor do fundo.
- Tamanho.
- Borda.
- Cursor.
- Entre outros.

Exemplo de alteração pelo código

```
JButton botaoOk = new JButton("Ok");  
botaoOk.setLocation(300, 200);  
botaoOk.setSize(150, 70);  
botaoOk.setForeground(Color.WHITE);  
botaoOk.setBackground(new Color(0, 102, 153));
```



Fonte: Autoria própria.

Método *setVisible(boolean visible);*

- O método *setVisible* permite que o componente seja devidamente iniciado, através da habilitação da visualização da janela.
- Também devemos prestar atenção ao método *repaint()*;

Interatividade

Sabendo que é possível desenvolver telas robustas com componentes básicos, qual das alternativas a seguir não é válida?

- a) Os contêineres são dispostos para que se aloquem outros componentes.
- b) Para se criar uma tela não é necessário o uso de todos os componentes (botões, campos de texto, etc.).
- c) *Swing* é menos acoplado que AWT, possuindo também mais consistência.
- d) Para a alteração de um objeto, o Java está alterando os atributos do sistema operacional.
- e) Tanto os componentes do AWT como os do *Swing* utilizam conceitos consolidados de *design* e usabilidade.

Resposta

Sabendo que é possível desenvolver telas robustas com componentes básicos, qual das alternativas a seguir não é válida?

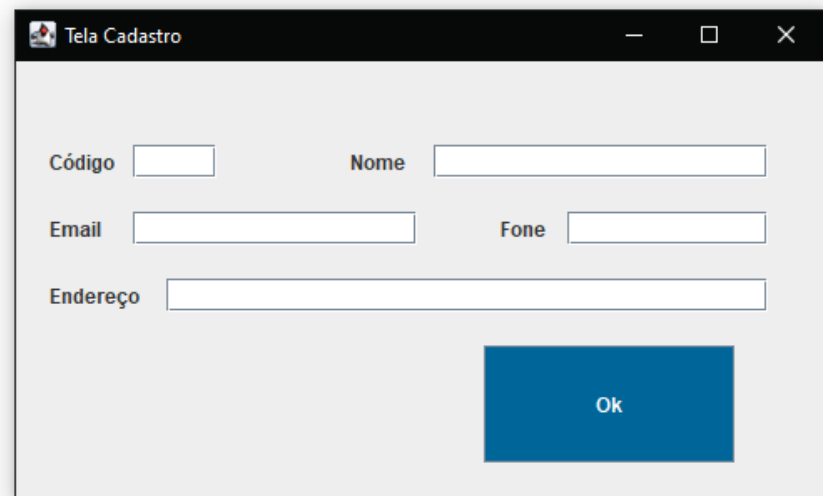
- a) Os contêineres são dispostos para que se aloquem outros componentes.
- b) Para se criar uma tela não é necessário o uso de todos os componentes (botões, campos de texto, etc.).
- c) *Swing* é menos acoplado que AWT, possuindo também mais consistência.
- d) Para a alteração de um objeto, o Java está alterando os atributos do sistema operacional.
- e) Tanto os componentes do AWT como os do *Swing* utilizam conceitos consolidados de *design* e usabilidade.

Código limpo

- Programas mais separados e organizados facilitam na manutenção de código, boas nomenclaturas permitem uma melhor legibilidade e maior encontro de discrepâncias.
- Robert C. Martin (2009), aclamado autor e líder da indústria de *software*, diz em seu livro *Código limpo* que “passamos mais tempo lendo nosso código do que o digitando”.

Desenvolvendo uma tela

- Para que os atributos gráficos existam, é necessário apenas desenvolver uma tela e aplicar os componentes à ela.



```
JFrame t01 = new JFrame();  
t01.setSize(new Dimension(300, 300));  
t01.setTitle("Tela Cadastro");  
t01.setLayout(null);  
t01.setSize(500, 300);
```

```
JLabel lb01 = new JLabel("Código");  
lb01.setSize(50, 20);  
lb01.setLocation(20, 50);  
t01.add(lb01);
```

```
t01.setVisible(true);
```

Fonte: Autoria própria.

Nome dos objetos

- Devemos sempre trabalhar a nosso favor.
- Muitos componentes costumam confundir o desenvolvedor se não forem devidamente referenciados.

```
JFrame frameDeCadastroDeUsuario = new JFrame();
```

Fonte: Autoria própria.

O uso dos painéis

- Os painéis permitem flexibilidade no desenvolvimento para conseguirmos alcançar o resultado esperado.
- Mas nem sempre ele nos permite criarmos a aparência desejada pela equipe de *design*.
- Para resolver esse problema temos os *Layouts*.

Layouts

- Contêineres podem receber outros componentes (inclusive outros contêineres).
- O *Layout* define como os componentes estarão dispostos dentro do contêiner.
- Como objetos, já existem diversos *Layouts* com características específicas para utilizarmos, a maioria da biblioteca AWT.

Layouts padrões

- O *Frame* / *JFrame* possui o *Layout* padrão *BorderLayout*.
- O *Panel* / *JPanel* possui o *Layout* padrão *FlowLayout*.

Os *Layouts* mais utilizados

- Nulo.
- *BorderLayout*.
- *FlowLayout*.
- *GridLayout*.
- *CardLayout*.

Adicionando *Layouts*

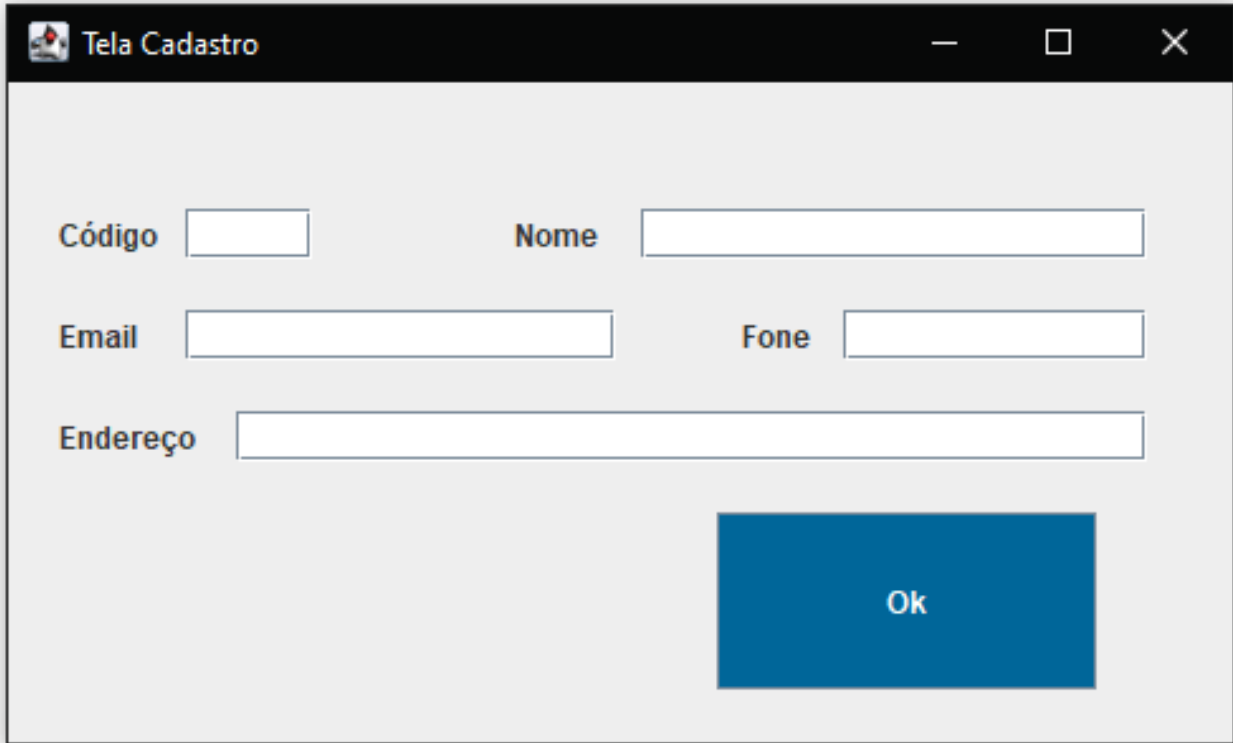
- Para isso, os contêineres possuem um método chamado *setLayout()*.
- Essa é uma modificação ao atributo *Layout*.

```
frameDeCadastroDeUsuario.setLayout(null);
```

Fonte: Autoria própria.

Layout Nulo

- Cada *Layout* segue um padrão de disposição dos componentes nele inseridos.
- Suas configurações também são específicas.
- Ao utilizar o *Layout* nulo, podemos inserir componentes em qualquer posição da tela, com qualquer tamanho.



The image shows a screenshot of a software window titled "Tela Cadastro". The window has a standard title bar with a minimize button, a maximize button, and a close button. The main content area is light gray and contains a registration form. The form consists of five text input fields arranged in three rows. The first row has "Código" and "Nome". The second row has "Email" and "Fone". The third row has "Endereço". All fields are empty. Below the fields, on the right side, is a large blue button with the text "Ok" in white.

Fonte: Autoria própria.

Layout Nulo

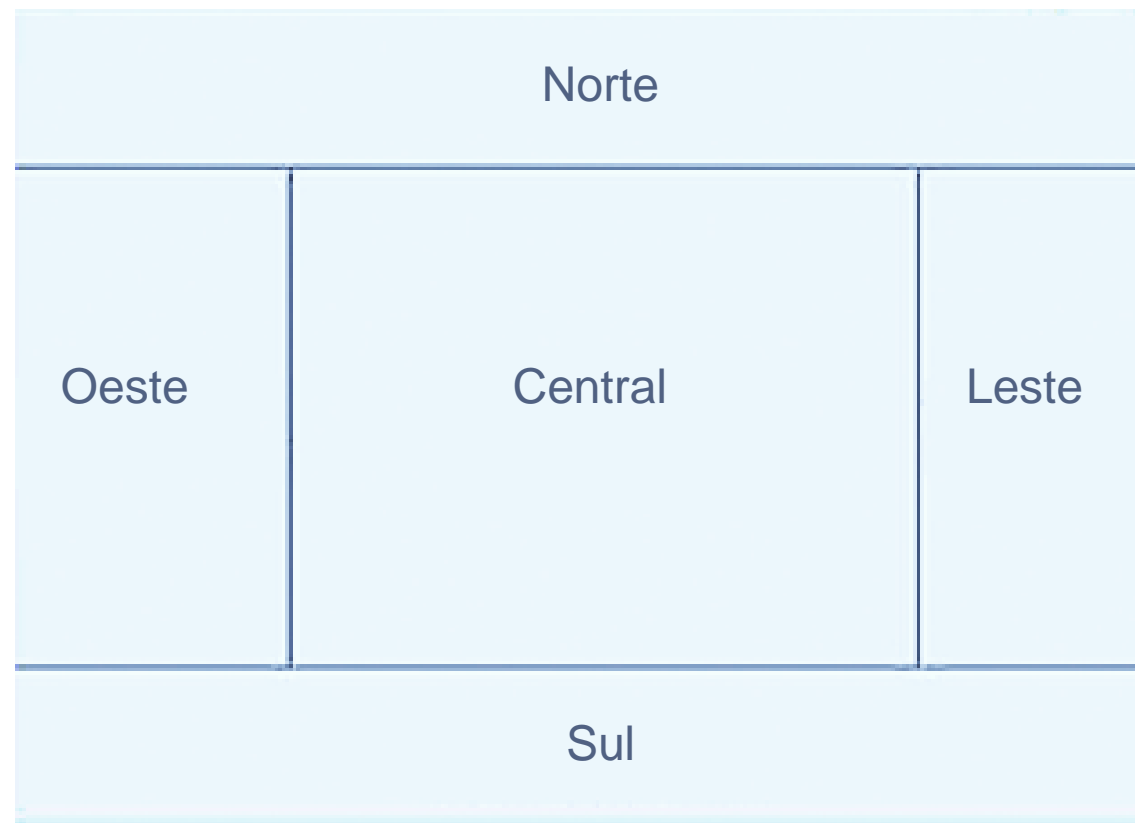
- Para modificar o local de aparecimento do componente, antes de adicioná-lo ao container, utilizamos o método *setLocation*.
- Para modificar seu tamanho, utilizamos o método *setSize*.
- Devido a essas especificações, muitas vezes temos que denotar o local ao rodar o programa ou fazer cálculos de posicionamento, mesmo se o *designer* informou o local e o tamanho.

```
JButton botaoOk = new JButton("Ok");  
botaoOk.setLocation(280, 170);  
botaoOk.setSize(150, 70);  
botaoOk.setForeground(Color.WHITE);  
botaoOk.setBackground(new Color(0, 102, 153));
```

Fonte: Autoria própria.

BorderLayout

- É o *Layout* padrão do *Frame* (AWT) ou *JFrame* (Swing).
- Esse *Layout* divide o componente em 5 partes.



BorderLayout

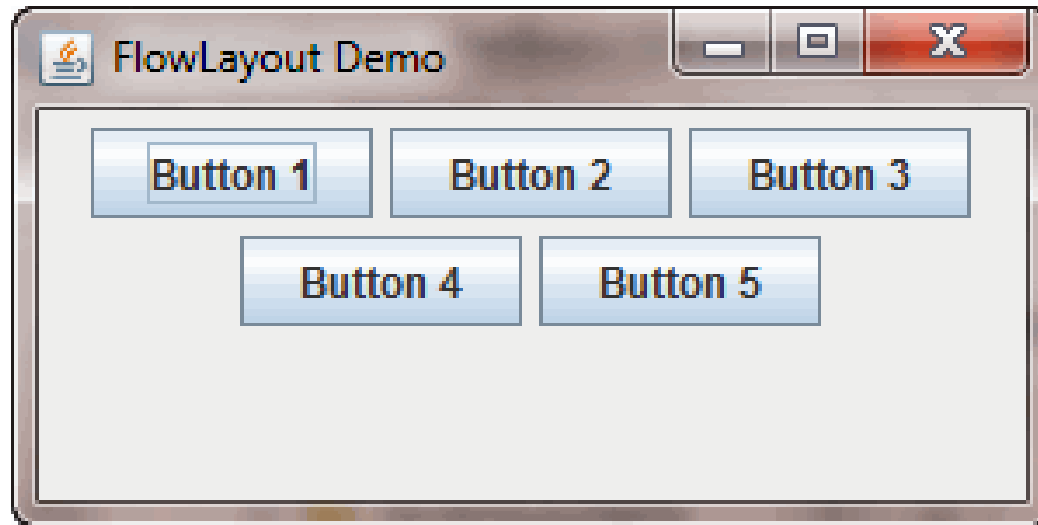
- Em cada divisão só é possível adicionar um componente (mas esse componente pode ser um *container*).
- Para adicionarmos um componente ao *BorderLayout*, informamos a disposição junto ao componente nos parâmetros do método *add* do *container*.
- Também é possível notar o nível de abstração do método *add* por permitir isso.

```
frameDeCadastroDeUsuario.add(botaoOk, BorderLayout.NORTH);
```

Fonte: Autoria própria.

FlowLayout

- É o *Layout* padrão do *Panel* (AWT) e do *JPanel* (Swing).
- O *FlowLayout* é usado para dispor os componentes um após o outro (um *flow*).



Fonte: <https://www.zentut.com>

FlowLayout

- Os componentes do *FlowLayout* são alocados centralizados, e conforme mais componentes são adicionados, eles tomam mais espaço da linha.
- Quando não sobra mais espaço, uma nova linha é criada para o novo componente adicionado, o que vai depender do tamanho do componente.
- Pode ser usado para guardar um outro *container*, com a certeza de que ele estará devidamente centralizado.

```
frameDeCadastroDeUsuario.setLayout(new FlowLayout());
```

Fonte: Autoria própria.

Interatividade

Por que temos diversos *Layouts* em contrapartida a um único e prático?

- a) Porque isso gastaria mais recursos do computador, visto que teríamos que alterar os objetos com mais frequência.
- b) Para se digitar menos, visto que teríamos que adicionar os componentes separadamente.
- c) O *Swing* foi criado pensando nos *Layouts* e isso invalidaria sua existência.
- d) Para que os desenvolvedores tivessem à disposição a aparência certa ou desejada com mais facilidade.
- e) Para aumentar o nível de complexidade da aplicação e tornar a programação mais nichada.

Resposta

Por que temos diversos *Layouts* em contrapartida a um único e prático?

- a) Porque isso gastaria mais recursos do computador, visto que teríamos que alterar os objetos com mais frequência.
- b) Para se digitar menos, visto que teríamos que adicionar os componentes separadamente.
- c) O *Swing* foi criado pensando nos *Layouts* e isso invalidaria sua existência.
- d) Para que os desenvolvedores tivessem à disposição a aparência certa ou desejada com mais facilidade.
- e) Para aumentar o nível de complexidade da aplicação e tornar a programação mais nichada.

A facilidade é extremamente recompensada

- A cada ano que passa, mais ferramentas, bibliotecas e *frameworks* são criados para auxiliar no desenvolvimento e permitir uma nova janela ao mercado para preencher um vazio de pessoas especializadas.
- Em 2022, houve a criação do Manifesto Tech, possível de encontrar em seu próprio site, que explica a necessidade de mais pessoas no mercado, de modo que as empresas procurassem também especializar rostos novos na área.

GridLayout

- Esse *Layout* divide a tela em partes de igual tamanho, dispostas em linhas e colunas.
- Cada componente adicionado ocupa aquela posição. Por exemplo, se um Grid de dimensão 3 por 3 for criado, ele irá adicionar seu primeiro componente na posição 1x1, como uma matriz.

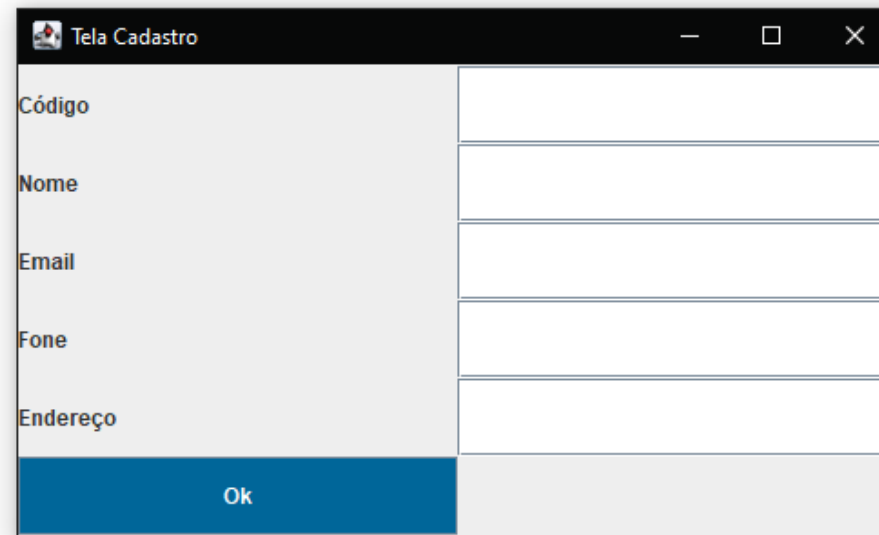


Fonte: Autoria própria.

GridLayout

- A sequência de adição ao *GridLayout* é da direita para a esquerda, de cima para baixo.

```
frameDeCadastroDeUsuario.setLayout(new GridLayout(6, 2));
```

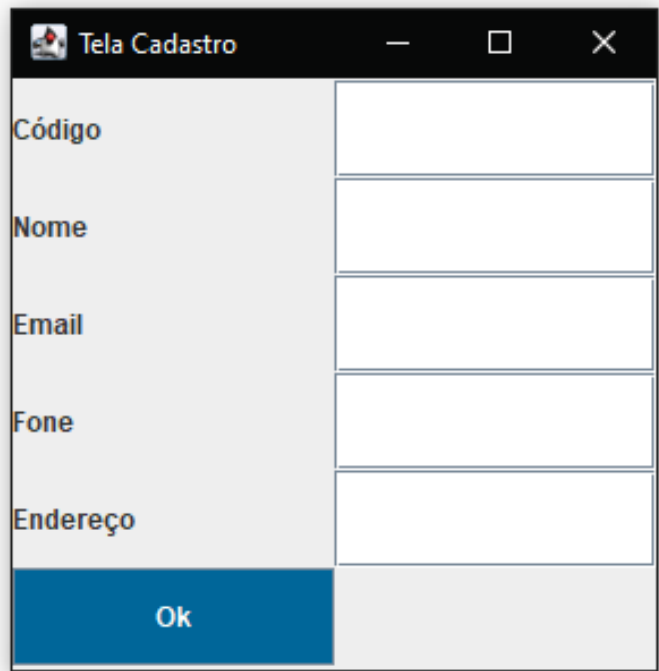


The screenshot shows a Java Swing window titled "Tela Cadastro" with a standard Mac OS-style title bar (minimize, maximize, close buttons). The window contains a registration form. On the left side, there are five labels: "Código", "Nome", "Email", "Fone", and "Endereço". To the right of these labels are five corresponding text input fields, arranged vertically. At the bottom left of the window is a blue button labeled "Ok". The entire form is organized using a GridLayout with 6 rows and 2 columns. The first five rows contain the labels and input fields, and the sixth row contains the "Ok" button and an empty space.

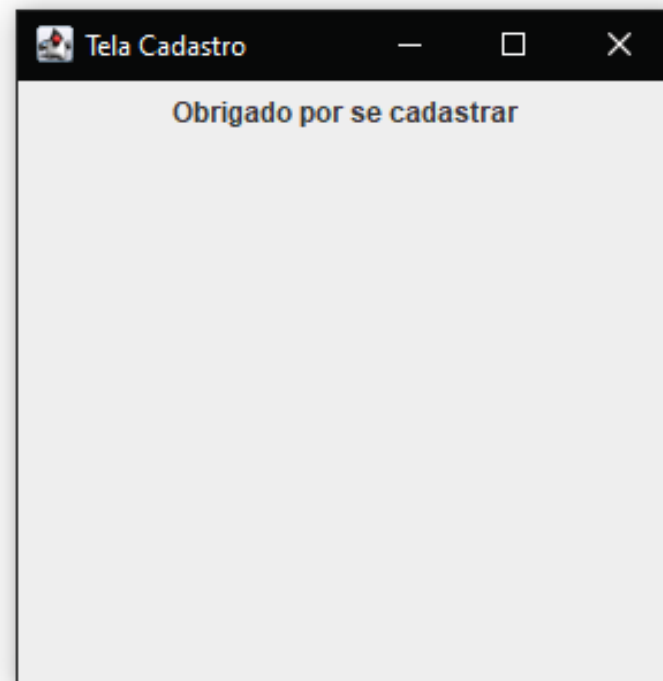
Fonte: Autoria própria.

CardLayout

- O *CardLayout* permite gerar várias telas, mostrando a cada momento apenas uma das telas.
- Ele viabiliza a troca de telas facilmente através de um *Layout*.



A screenshot of a Java Swing window titled "Tela Cadastro". The window contains a registration form with the following fields: "Código", "Nome", "Email", "Fone", and "Endereço". Each field is represented by a text label on the left and an empty text input box on the right. At the bottom left of the form is a blue button labeled "Ok". The window has standard Mac OS X window controls (red, yellow, and green buttons) in the title bar.



A screenshot of a Java Swing window titled "Tela Cadastro". The window displays a confirmation message: "Obrigado por se cadastrar". The text is centered in a bold, black font. The window has standard Mac OS X window controls (red, yellow, and green buttons) in the title bar.

Fonte: Autoria própria.

CardLayout

- Ele será criado com o primeiro item adicionado.
- Itens posteriores podem ser chamados através do método *show*.

```
CardLayout layoutDoFrameDeCadastro = new CardLayout();  
frameDeCadastroDeUsuario.setLayout(layoutDoFrameDeCadastro);  
layoutDoFrameDeCadastro.show(painel2, "segundo painel");
```

Fonte: Autoria própria.

Estilos de interação

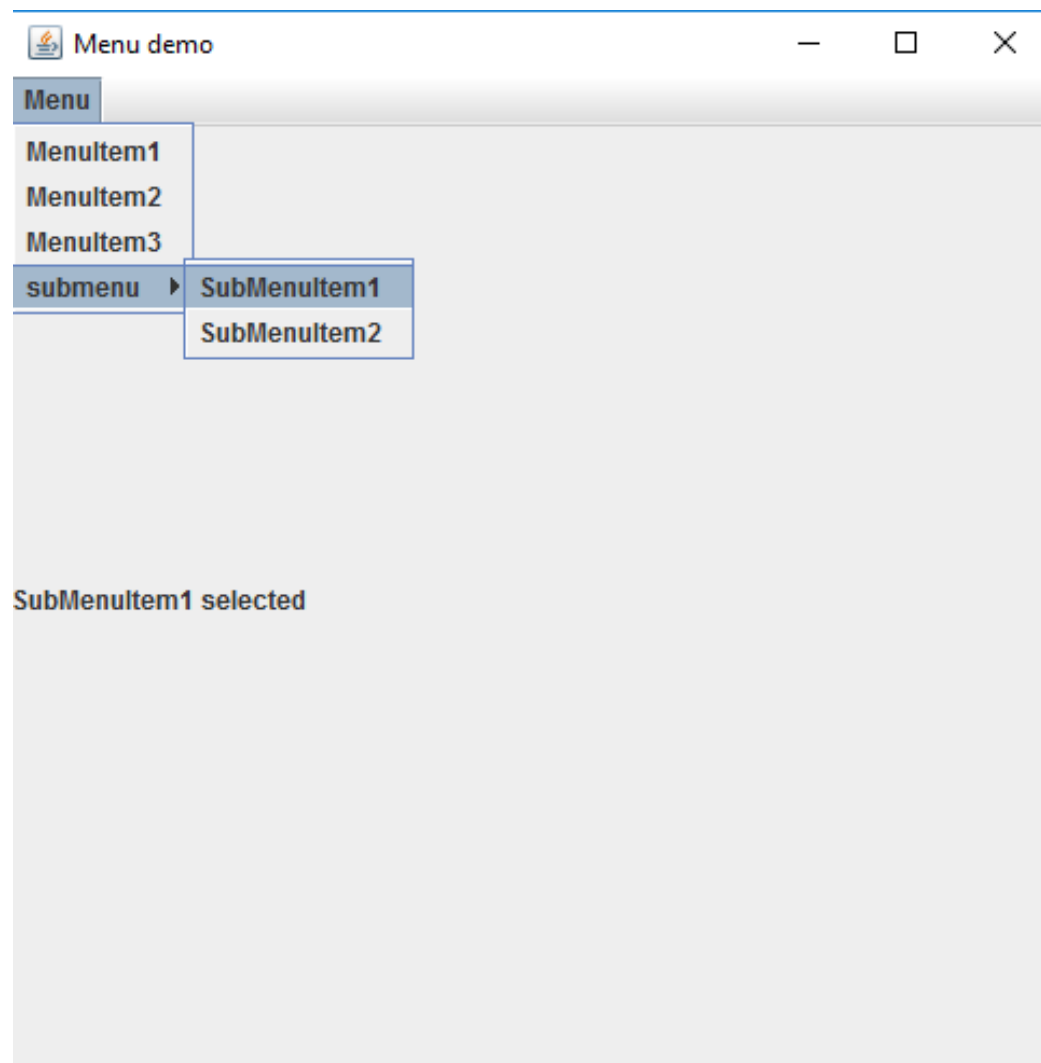
- O conceito de estilos de interação refere-se a todas as maneiras pelas quais o usuário pode se comunicar ou interagir com o sistema de computador. Esses conceitos, no entanto, mantêm alguns de seus poderes descritivos fora do meio computacional. Por exemplo, você pode falar sobre a seleção de menu em telefones celulares.

Estilos de interação

- Linguagem de comando.
- Preenchimento de formulário.
- Seleção de menu.
- Manipulação direta.

Menu

- *MenuBar / JMenuBar*
- *Menu / JMenu*
- *MenuItem / JMenuItem*
- *JSeparator*



Fonte: <https://www.geeksforgeeks.org/java-swing-jmenubar/>

MenuBar

- Cria a barra superior para adição de menus, os componentes do tipo menu devem ser adicionados nela.
- Pode conter diversos menus.



Fonte: Autoria própria.

Menu

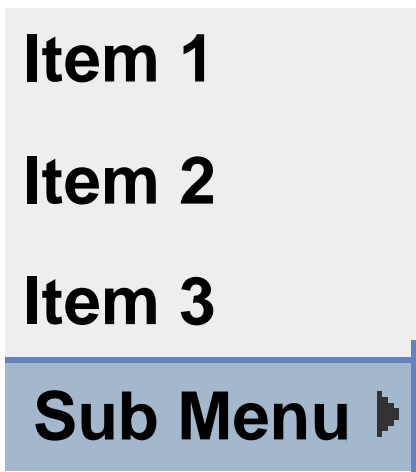
- Aqui é criado um menu, cuja utilização é igual a de componentes visuais normais, mas é adicionado na *MenuBar*.
- O menu deve conter os itens do menu, ou *MenuItems*. Ele pode conter também submenus, dos quais possuem outras opções, esses submenus também são do componente Menu.
- Os itens do menu podem ser separados por um *JSeparator*.



Fonte: Autoria própria.

MenuItem

- Esses são os itens clicáveis de um menu, estamos acostumados com itens como Salvar ou Abrir.
- Divide-se o espaço com os submenus dentro de um Menu.



Fonte: Autoria própria.

Usando o que foi aprendido

- Com esse conhecimento, é possível criar uma tela com menu, que pode mudar os painéis através de um *CardLayout* quando são clicados.

Interatividade

Ao final desse bloco, em qual nível está a aplicabilidade de estilos de interação?

- a) Adentramos a área de linhas de comando.
- b) Estamos prontos para mexer com formulários.
- c) Podemos implementar seleção de menus.
- d) Aplicamos a manipulação direta.
- e) Podemos criar *touchscreens*.

Resposta

Ao final desse bloco, em qual nível está a aplicabilidade de estilos de interação?

- a) Adentramos a área de linhas de comando.
- b) Estamos prontos para mexer com formulários.
- c) Podemos implementar seleção de menus.
- d) Aplicamos a manipulação direta.
- e) Podemos criar *touchscreens*.

Referências

- MARTIN, C. R. *Código limpo: habilidades práticas do Agile Software*. São Paulo: Alta Books, 2009.

ATÉ A PRÓXIMA!