VOLTAR

Conceitos Básicos de GUI em Java

De que adianta desenvolver aplicações poderosíssimas em Java se o usuário deve opera-la via terminal? Bem, atualmente softwares de terminal estão quase extintos! Chegou a hora de começarmos a programar além do console do Java, ou seja, chegou a hora de começarmos a desenvolver interfaces gráficas de usuário (GUI - Graphical User Interface), para que seus programas sejam mais amigáveis, robustos e funcionais. Bem-vindo ao mundo Java orientado a objetos e com interfaces gráficas.

NESTE TÓPICO

O que é uma GUI?

GUI significa *Graphical User Interface* ou, em português, interfaces gráficas de usuário. Uma interface, por definição, é um dispositivo físico ou lógico que permite a comunicação entre dois meios que são distintos.

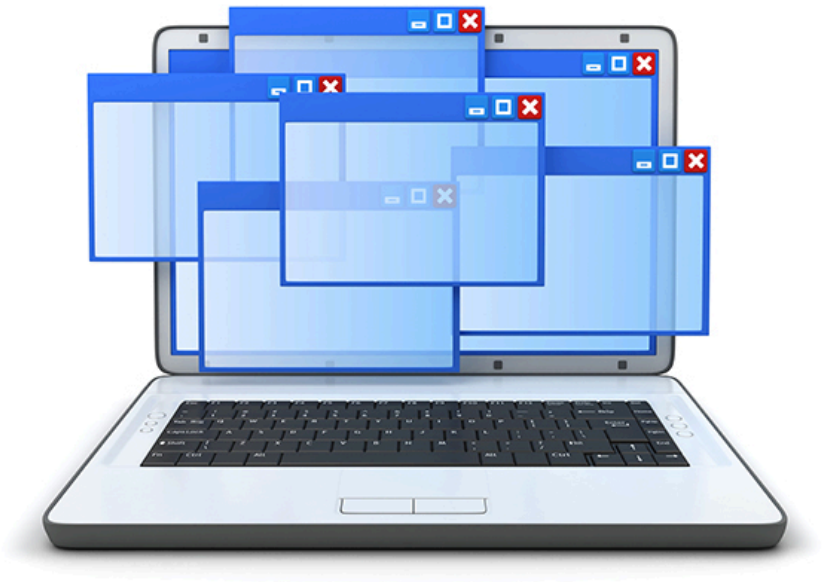
Isso quer dizer que uma interface é um mecanismo de comunicação. Tudo que está a sua volta possui uma interface de comunicação: O celular possui a interface física que exibe uma interface lógica que, por sua vez, promove acesso aos recursos do aparelho.

É importante ressaltar que interfaces podem ser físicas, também. O aparelho de celular é uma interface física. O computador é uma interface física, a cabine de um avião é uma interface física, assim por diante.

Contudo, como estamos trabalhando com Java, não vamos nos preocupar com interfaces físicas, mas sim com interfaces lógicas, ou seja, interfaces de que o usuário opera em um computador.

É importante ressaltar, ainda, que estamos trabalhando com interfaces para sistemas desktop. Interfaces para sistemas mobile, web e demais dispositivos (relógios, máquinas de ultrassom, aviões etc.) serão vistas

em disciplinas específicas para este contexto pois, embora Java permita que programemos para praticamente qualquer dispositivo, há diferenças crucias na organização e administração do projeto para cada tipo de interface. Resumindo, trabalharemos aqui com interfaces para desktop.



Interfaces de usuário no computador

GUI em Java

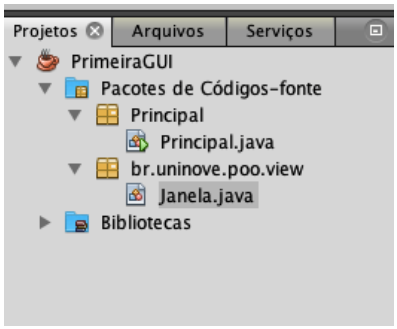
Temos, em Java, duas APIs para uso gráfico: a **AWT** (Abstract Window Toolkit) e a **Swing**.

A **AWT** ainda pode ser utilizada, mas foi substituída pela **Swing** a partir do Java 1.2. A biblioteca Swing provê melhor aparência aos componentes, melhor tratamento de eventos e recursos estendidos. As classes do **Swing** são extensões do pacote **AWT**. Em outras palavras, vamos usar o pacote **Swing** pois é mais recente, enquanto pacote **AWT** é antigo e não foi é atualizado desde a versão 2 do Java.

As classes da **AWT** fazem parte do pacote **java.awt** e as classes do **Swing** são encontradas no pacote **javax.swing**. Uma vantagem do uso do pacote **Swing** é que uma aplicação terá a mesma forma, aparência e comportamento, independente do sistema operacional em que rodar. Mas agora, é hora de programar.

Primeiro programa gráfico em Java

Para esta primeira aplicação, vamos criar uma janela vazia. Teremos uma classe “Janela” e outra classe para aciona-la, camada principal. Cada uma das classes será colocada em um pacote, por boas práticas de programação. Nossa estrutura de projeto ficará assim:



Estrutura do projeto de GUI

Vamos, agora, a codificação das classes. Primeiro criamos a classe ?Janela", pois, sem ela, não conseguiremos criar a classe principal (que faz referência a ela):

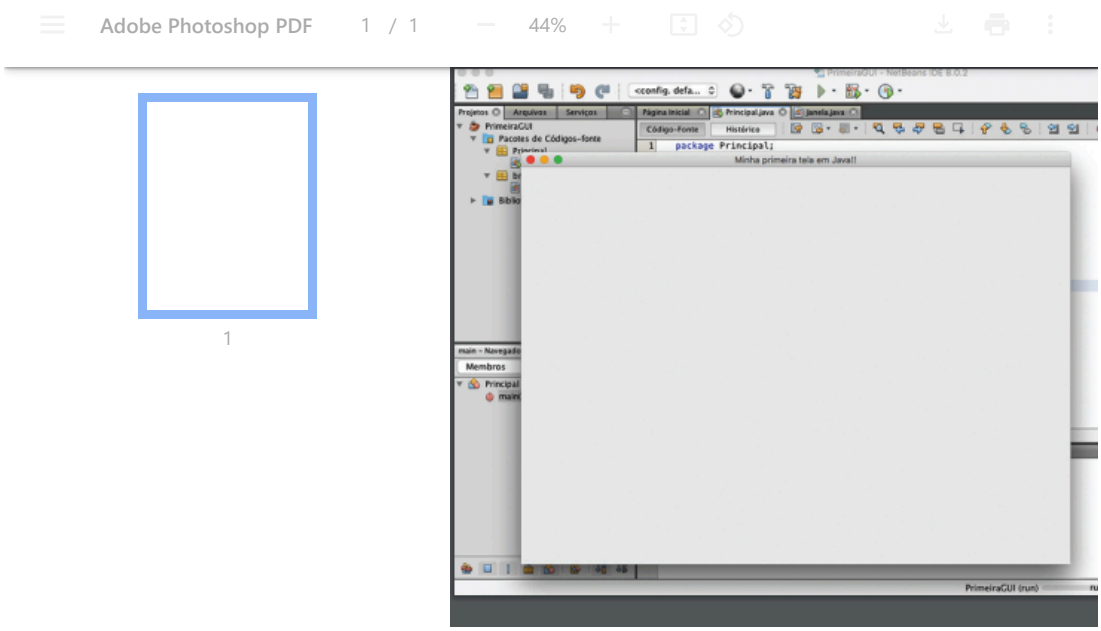
```
1.  package br.uninove.poo.view;
2.
3.  //É preciso importar o API do Swing:
4.  import javax.swing.JFrame;
5.
6.  public class Janela {
7.
8.      JFrame janela = new JFrame(); //criar um objeto JFrame chamado 'janela1'
9.
10.     //Construtor da classe, que gera um objeto de tela (jFrame)
11.     public Janela() {
12.         janela.setTitle("Minha primeira tela em Java!!");  //Titulo da janela
13.         janela.setSize(800, 600); //largura e a altura da janela, em pixels
14.         janela.setLocation(100, 150); //define a posição da janela coluna e linha
15.          janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  //ao fechar a janel
    a, encerra a aplicação
16.     }
17.
18.     //Método para mostrar a Janela
19.     public void mostraJanela() {
20.         janela.setVisible(true); //torna a janea visível
21.     }
22. }
```

E agora o código da classe “Principal”.

```
1.  package Principal;
2.
3.  import br.uninove.poo.view.Janela;
4.
5.  public class Principal {
6.
7.      public static void main(String args[]) {
8.          //Cria uma instância da classe Janela
9.          //Este trecho de código aciona o construtor da classe Janela:
10.         Janela j = new Janela();
11.
12.         //Dispara o método que trona a Janela vidível
13.         j.mostraJanela();
14.     }
15. }
```

É importante ressaltar que, mesmo trabalhando com interfaces gráficas, ainda precisamos de uma classe que contenha o método ***main*** pois este é o primeiro método chamado pela máquina virtual Java. Em outras palavras, todas as aplicações precisam de um método ***main***, com a assinatura como sempre fizemos (public static void main(String args[]).

Veja o resultado da execução do código acima:



Resultado da execução do código acima - Primeira interface gráfica em Java

É importante notar que essa janela possui 800x600 pixels de tamanho e foi iniciada na posição 100, 150 do monitor. Essas informações, de tamanho e posição, são sentadas no construtor da classe "Janela".

O tamanho é definido em pixels e a posição refere-se a posição x, y do monitor, em pixels. A posição (0,0) é o canto superior esquerdo da tela. Troque para a posição (0,0) em seu código para comprovar a nova posição a qual a janela foi iniciada.

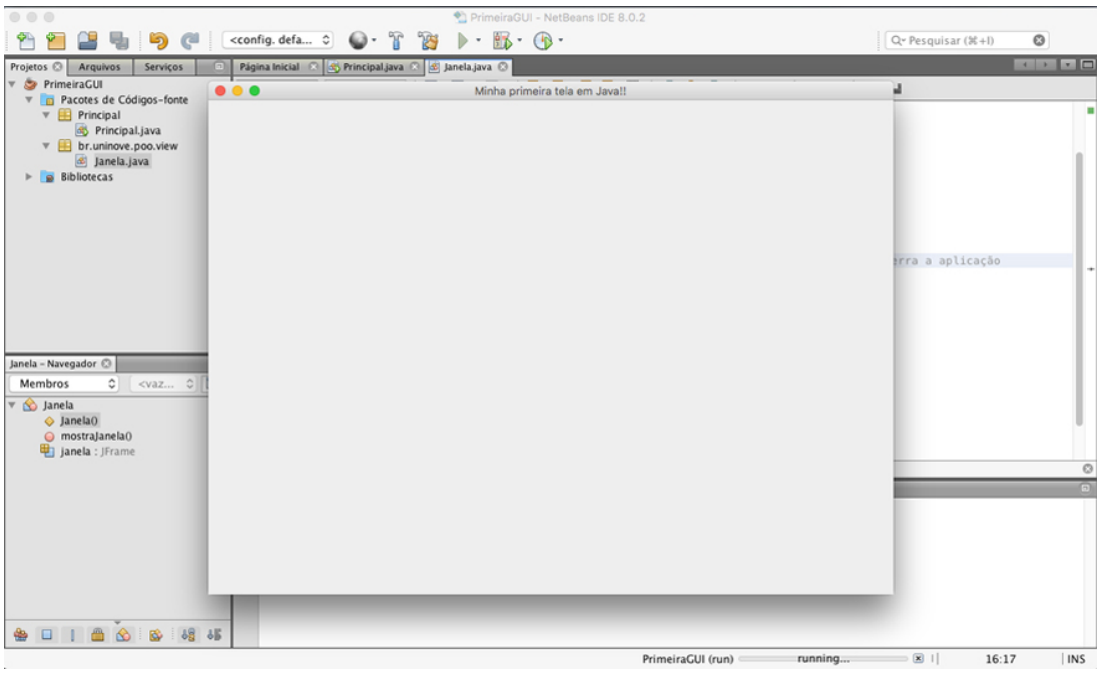
Se você quiser que sua janela fique centralizada na tela, é preciso retirar a seguintes linha de código, na classe “Janela”:

```
1.      janela.setLocation(100, 150); //define a posição da janela coluna e linha
```

E acrescentar:

```
1.      janela.setLocationRelativeTo(null);
```

Veja o novo resultado da execução após essa alteração:



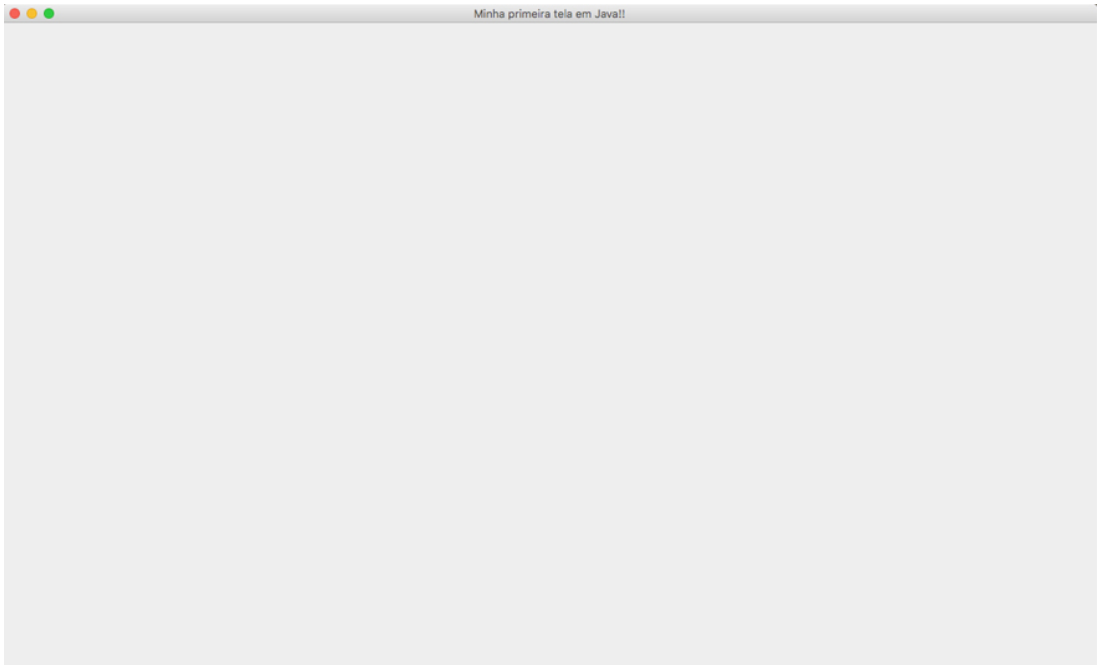
Resultado da execução do projeto de GUI, agora com a janela centralizada

Note, na imagem acima, que a tela foi carregada exatamente no centro da tela completa.

Se você quiser, agora, que a sua janela ocupe 100% do monitor, independentemente da resolução que o monitor possui (pois o programa pode ser rodado em absolutamente qualquer sistema operacional desktop), você precisa **acrescentar** (não precisa retirar nada), o seguinte trecho de código no construtor da classe ?Janela?:

```
1.      janela.setExtendedState(JFrame.MAXIMIZED_BOTH);
```

E novo resultado será como visto abaixo:



Resultado da execução do projeto de GUI, agora com a janela em tamanho máximo

Veja que, agora, a janela ocupa 100% da tela. Isso independe da resolução da tela do cliente, pois o trecho de código está setando o tamanho para "maximizado".

Entendendo melhor o posicionamento dos componentes e das telas

Conforme mostrado nos exemplos acima, tudo é tratado, dentro do **Swing**, com posicionamento em pixels no monitor e dentro da própria tela. Claro, isso se estivermos trabalhando com posicionamento estático. Veremos, na próxima aula, que há gerenciadores automáticos de layout, que fazem o posicionamento de forma automática para nós, mas antes é preciso aprender sobre o posicionamento estático.

No primeiro exemplo de código mostrado aqui, foi apresentado um posicionamento para a janela com a seguinte instrução:

```
1.        janela.setLocation(100, 150);
```

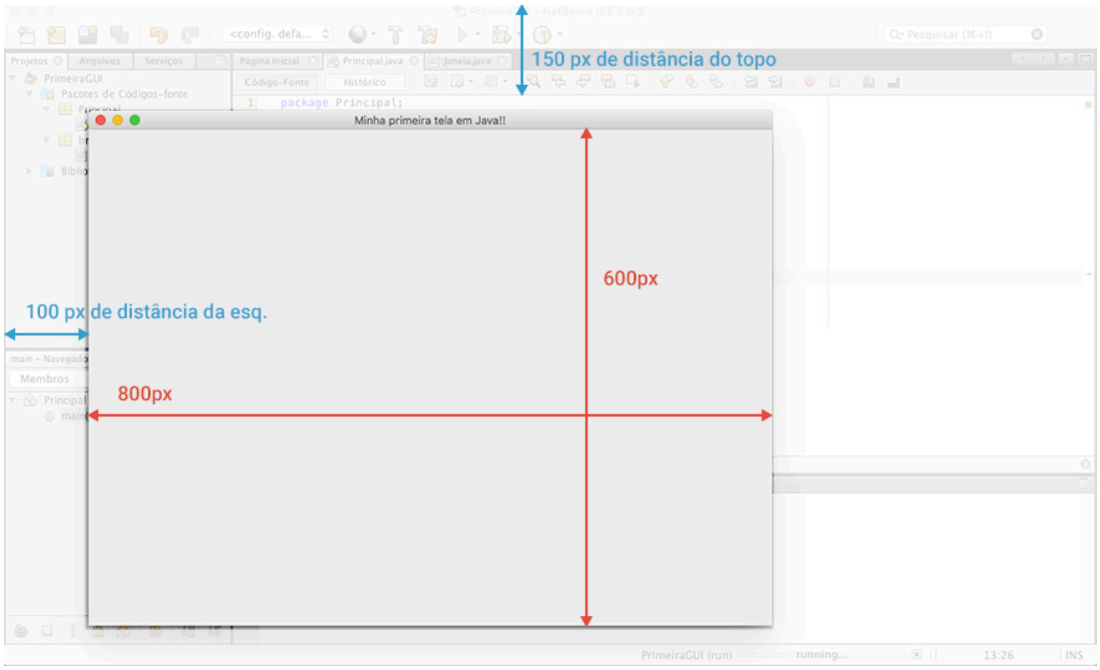
Essa linha de comando indica que a janela irá se posicionar a 100 pixels de distância da parte de cima da tela e 150 pixels de distância da parte esquerda.

E usamos a seguinte instrução para definir o tamanho da janela:

```
1.        janela.setSize(800, 600);
```

Essa instrução define que nossa janela terá 800 pixels de largura e 600 pixels de altura.

A imagem abaixo mostra, graficamente, como isso ocorre, de fato, no monitor:



Demonstração gráfica do tamanho e posicionamento da tela que criamos

Repare na aplicação dos valores setados no código: 100 pixels de distância da borda esquerda e 150 pixels de distância da parte superior da tela. E, em relação ao tamanho, 800 pixels de largura e 600 puxeis de altura.

É muito importante tomar cuidado para não ultrapassar os limites da resolução de seu monitor, para que a tela não fique perdida ou desuniforme.

Componentes

Claro que não vamos desenvolver apenas janelas cinzas, sem nada dentro. Vamos colocar, também, componentes, como botões, caixas de texto, checkboxes, botões radio etc.

Os componentes do pacote Swing mais comuns são:

- **JButton:** Para botões.
- **JTextField:** Para campos de texto.
- **JLabel:** Representa um campo de texto que não pode ser editado pelo usuário, como uma etiqueta.
- **JPasswordField:** São campos de texto cujo texto não é mostrado para o usuário, normalmente usados em campos de senhas.

- **JCheckBox**: São as caixas para múltipla seleção.
- **JComboBox**: São as caixas de seleção suspensas, onde o usuário clica na caixa e uma lista de opções é exibida. O usuário seleciona uma e esta é mostrada no componente.
- **JList:** São listas que permitem múltiplas seleções.
- **JMenu:** São as caixas de menu padrões de aplicações desktop, que aparecem na parte superior da aplicação.
- **JRadioButton:** Representam botões de rádio, onde o usuário possui várias opções e escolhe apenas uma.
- **JSlider:** Representam um controle que pode ser deslizado de um lado para o outro, como um controle de volume, por exemplo.

Para praticar, vamos criar uma nova classe com alguns destes componentes.

É importante ressaltar que quando temos uma classe que representará uma interface gráfica, podemos deriva-la da classe **jFrame** pois já estão implementados dos os metidos e atributos necessários que poderemos reutiliza em nossa classe, ou seja, para facilitar nossa implementação, vamos herdar de **jFrame** todas características e comportamentos e aplicar o que for necessário para a nossa tela (lembre-se do conceito de herança).

Para este exemplo, vamos criar uma tela com dois campos de texto, dois botões e labels para indicar os rótulos dos campos de texto. Veja, abaixo, a codificação da nova tela (criamos uma nova classe para ela, neste caso, chamada “Tela2”):

```
1.  package br.uninove.poo.view;
2.
3.  import javax.swing.JButton;
4.  import javax.swing.JFrame;
5.  import javax.swing.JLabel;
6.  import javax.swing.JTextField;
7.
8.  //Nossa classe Tela2 é um jFrame, por isso o extends
9.  public class Tela2 extends JFrame {
10.
11.     //Atributos locais
12.     private JLabel lblNome, lblRenda;
13.     private JTextField txtNome, txtRenda;
14.     private JButton btnSalvar, btnSair;
15.
16.     //Construtor
17.     public Tela2() {
18.         //Define o leyout como nulo (nós vamos o definir)
19.         setLayout(null);
20.         setDefaultCloseOperation(EXIT_ON_CLOSE); //Ao fechar, encerra o programa
21.
22.         //Propriedades do jFrame:
23.         setSize(600, 200); //Tamanho
24.         setLocation(200, 300); //Posição
25.         setTitle("Controle de Clientes"); //Titulo
26.
27.         //Propriedades dos campos do formulário
28.         lblNome = new JLabel("Nome:");
29.         lblNome.setSize(50, 30);
30.         lblNome.setLocation(30, 10);
31.         add(lblNome);
32.
33.         txtNome = new JTextField();
34.         txtNome.setSize(480, 30);
35.         txtNome.setLocation(80, 10);
36.         add(txtNome);
37.
38.         lblRenda = new JLabel("Renda:");
39.         lblRenda.setSize(50, 30);
40.         lblRenda.setLocation(30, 45);
41.         add(lblRenda);
42.
43.         txtRenda = new JTextField();
44.         txtRenda.setSize(100, 30);
45.         txtRenda.setLocation(80, 45);
46.         add(txtRenda);
47.
48.         btnSalvar = new JButton("Salvar");
49.         btnSalvar.setSize(80, 30);
50.         btnSalvar.setLocation(30, 90);
51.         add(btnSalvar);
52.         btnSair = new JButton("Sair");
53.         btnSair.setSize(80, 30);
54.         btnSair.setLocation(120, 90);
55.         add(btnSair);
56.     }
57. }
```

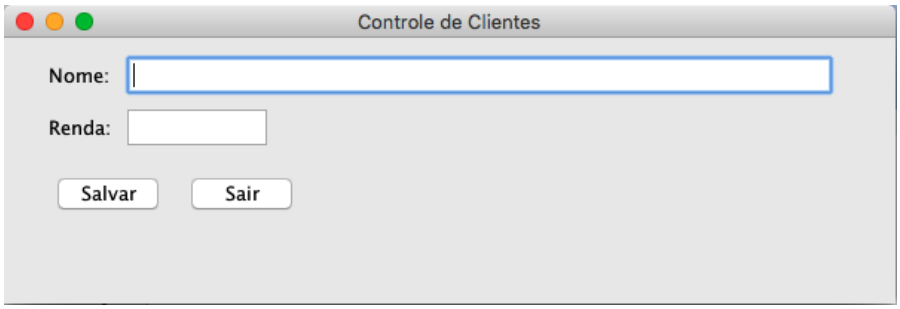
Veja no código acima, que o método “add” é invocado diretamente, pois é um método que pertence a classe mãe que estamos utilizando (jFrame).

Para que essa tela seja exibida, alteramos a classe “Principal” para a seguinte codificação:

```
1.  package Principal;
2.
3.  import br.uninove.poo.view.Tela2;
4.
5.  public class Principal {
6.
7.      public static void main(String args[]) {
8.          //Cria uma instância da classe Tela2
9.          Tela2 t2 = new Tela2();
10.
11.         //Seta a tela2 como visiel
12.         t2.setVisible(true);
13.     }
14. }
```

Cuidado, para manter a organização em pacotes. Ao implementar, se você criou pacotes diferentes, é preciso adaptar na codificação das classes.

O resultado, da execução deste código pode ser viso abaixo:



Resultado da execução do código de GUI com componentes

Para praticar, tente criar outras telas, com outros componentes (conforme listados aqui). Tente, também, alterar deliberadamente o posicionamento das telas e tamanho de suas janelas.

Resumo da aula

Nesta aula você aprendeu a criar suas primeiras interfaces gráficas de usuário. É muito importante lembrar, dessa aula, que:

- Quando você está criando manualmente as interfaces gráficas, você precisa estar cuidadosamente a posição a qual a tela é iniciada e o tamanho de suas janelas.
- Você possui diversos componentes a disposição para usar em suas interfaces gráficas.

É muito importante praticar bastante. Não deixe de programar várias interfaces para praticar. Bons estudos e boa programação.

Quiz

Exercício Final

Conceitos Básicos de GUI em Java

INICIAR

Referências

Deitei P. e Deitel H., 2010, Java : Como programar, 8ª Edição, Pearson Pretice Hall

Teruel, E. C., 2015, Programação Orientada a Objetos com Java - sem mistérios - 1ª Ed., Editora Uninove

Schildt, H., 2015, Schildt, Java para iniciantes : crie, compile e execute programas Java rapidamente, Bookman

Avalie este tópico

Ajuda? (https://ava.un idCurso=)

Mapa do Site