

1 Introdução a Interfaces gráficas

O pacote **javax.swing** e seus sub pacotes possuem classes úteis para a criação de interfaces gráficas para o usuário (GUI: *Graphical User Interface*). Até então, fizemos uso de uma delas a **JOptionPane**. Neste material estudaremos a criação de interfaces gráficas utilizando diferentes recursos, indo além da simples exibição de caixas de diálogo.

2 Desenvolvimento

2.1 (A classe JFrame) A criação de uma interface gráfica (uma tela) envolve o uso da classe **JFrame**, que faz parte do pacote **javax.swing**. A palavra **frame** significa **moldura**. A ideia é construirmos uma moldura e então, adicionarmos conteúdo a ela, como botões, campos textuais, tabelas, menus etc. Um **JFrame** possui um **painel de conteúdo** (do inglês **content pane**). Trata-se de um objeto capaz de armazenar outros objetos, pois não adicionamos conteúdo diretamente a um **JFrame**. Uma vez que tenhamos um **JFrame** em mãos, obtemos uma referência ao seu painel de conteúdo e adicionamos conteúdo a ele.

2.2 (Hello, Swing) Vamos criar um primeiro exemplo para ilustrar o uso de duas classes do pacote **Swing**: **JFrame** e **JLabel**. Criaremos uma aplicação que exibe uma tela com o texto **Hello, Swing**.

- Crie um novo projeto e dê a ele um nome que esteja de acordo com o contexto estudado.
- Crie a classe **HelloSwing** e adicione a ela um método auxiliar, como na Listagem 2.2.1.

Listagem 2.2.1

```
public class HelloSwingTela{  
    public static void criarTela() {  
  
    }  
}
```

- Para instanciar um JFrame, podemos utilizar um construtor que recebe uma String. Ela será utilizada como título da tela. Na Listagem 2.2.2, além de fazermos isso, também utilizamos um método para especificar qual o comportamento esperado quando a tela for fechada pelo usuário. Essa operação é necessária pois, por padrão, caso o usuário feche a aplicação, a tela somente ficará invisível, mas programa continuará em execução.

Listagem 2.2.2

```
public static void criarTela() {  
    JFrame tela = new JFrame ("Hello, Swing!!!");  
    tela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
}
```

- A finalidade da classe **JLabel** é exibir conteúdo textual que o usuário não pode editar. Para fazer isso, é preciso construir um objeto desse tipo, informar o texto a ser exibido no construtor e adicionar o objeto ao painel de conteúdo do JFrame. Veja a Listagem 2.2.3.

Listagem 2.2.3

```
//constroi um JLabel  
JLabel helloSwingLabel = new JLabel ("Hello, Swing!!!!!!");  
//obtem o painel de conteúdo  
Container painelDeConteudo = tela.getContentPane();  
//adiciona o JLabel ao painel de conteúdo  
painelDeConteudo.add(helloSwingLabel);
```

- Por padrão, objetos JFrame são invisíveis. Precisamos tornar o objeto que criamos visível. Isso pode ser feito com o método **setVisible**. Além disso, usamos o método **pack** para fazer com que sua largura e altura se ajustem de acordo com as medidas de seu conteúdo. Veja a Listagem 2.2.4.

Listagem 2.2.4

```
//ajusta largura e altura da tela conforme seu conteúdo  
tela.pack();  
//torna a tela visível  
tela.setVisible(true);
```

- A fim de colocar o programa em execução, precisamos criar o método main. Ocorre que o método main é executado pelo fluxo principal (que chamamos de thread principal). Códigos que envolvem a atualização de componentes visuais devem ser

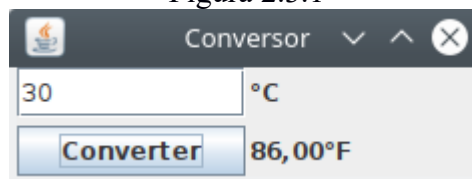
executados em um fluxo de execução à parte, que executa de maneira independente do principal e é reservado somente para isso. A programação concorrente foge do escopo da disciplina, porém é importante ter essa noção básica. A forma correta de se colocar em execução o nosso programa é exibida na Listagem 2.2.5. Não se preocupe com alguns detalhes. Isso se aprende com o tempo. Por hora, lembre-se de iniciar seu programa com interfaces gráficas sempre dessa forma.

Listagem 2.2.5

```
public static void main(String[] args) {  
    SwingUtilities.invokeLater(() -> {  
        criarTela();  
    });  
}
```

2.3 (Conversão de Celsius para Fahrenheit) No Brasil, a temperatura é, em geral, medida em graus Celsius. Em outros países, é comum o uso de graus Fahrenheit. Vamos escrever um programa que permite ao usuário informar um valor em graus Celsius e fazer a conversão para Fahrenheit. Sua tela será parecida com a exibida pela Figura 2.3.1.

Figura 2.3.1



- Comece criando um novo projeto/programa com um nome adequado para o contexto.
- Crie a classe da Listagem 2.3.1. Seu método criarTela, como o nome sugere, será o responsável por criar a tela da aplicação. Ele já instancia um JFrame.

Listagem 2.3.1

```
public class ConversorDeTemperatura {  
  
    public static void criarTela(){  
        JFrame tela = new JFrame ("Conversor");  
    }  
}
```

- Utilizaremos as classes descritas na Tabela 2.3.1 para montar a tela.

Tabela 2.3.1

Classe	Finalidade
JButton	Botões
JLabel	Componentes textuais não editáveis pelo usuário
JTextField	Componentes textuais editáveis pelo usuário

A construção de cada um deles é exibida na Listagem 2.3.2.

Listagem 2.3.2

```
public static void criarTela(){
    JFrame tela = new JFrame ("Conversor");
    JTextField celsiusTextField = new JTextField (10);
    JLabel celsiusLabel = new JLabel ("\u00B0C");
    JButton convertButton = new JButton ("Converter");
    JLabel valorConvertidoLabel = new JLabel ("");
}
```

- Para que apareçam na tela, os componentes precisam ser adicionados ao painel de conteúdo do JFrame. Isso é feito na Listagem 2.3.3.

Listagem 2.3.3

```
public static void criarTela(){
    JFrame tela = new JFrame ("Conversor");
    JTextField celsiusTextField = new JTextField (10);
    JLabel celsiusLabel = new JLabel ("\u00B0C");
    JButton convertButton = new JButton ("Converter");
    JLabel valorConvertidoLabel = new JLabel ("");
    Container painelDeConteudo = tela.getContentPane();
    painelDeConteudo.setLayout (new GridLayout (2, 2, 4, 4));
    painelDeConteudo.add(celsiusTextField);
    painelDeConteudo.add(celsiusLabel);
    painelDeConteudo.add(convertButton);
    painelDeConteudo.add(valorConvertidoLabel);
}
```

- O botão é um componente que pode sofrer um **evento** de interesse. Ele pode ser **clicado**. Desejamos especificar um método que será executado quando esse evento acontecer, pois é nesse momento que a conversão deve acontecer. O tratamento de eventos com Swing é feito por meio da aplicação do padrão de projetos **Observer**. Veja uma possível implementação na Listagem 2.3.4.

Listagem 2.3.4

```
public static void criarTela(){
    JFrame tela = new JFrame ("Conversor");
    JTextField celsiusTextField = new JTextField (10);
    JLabel celsiusLabel = new JLabel ("\u00B0C");
    JButton convertButton = new JButton ("Converter");
    JLabel valorConvertidoLabel = new JLabel ("");
    Container painelDeConteudo = tela.getContentPane();
    painelDeConteudo.setLayout (new GridLayout (2, 4, 2, 4));
    painelDeConteudo.add(celsiusTextField);
    painelDeConteudo.add(celsiusLabel);
    painelDeConteudo.add(convertButton);
    painelDeConteudo.add(valorConvertidoLabel);

    convertButton.addActionListener((e) -> {
        double celsius = Double.parseDouble(
            celsiusTextField.getText()
        );
        double fahrenheit = celsius / 5 * 9 + 32;
        valorConvertidoLabel.setText(
            String.format("%.2f\u00B0F", fahrenheit)
        );
    });
}
```

- Finalmente, fazemos os ajustes para que o JFrame seja exibido adequadamente, como na Listagem 2.3.5.

Listagem 2.3.5

```
public static void criarTela(){
    JFrame tela = new JFrame ("Conversor");
    JTextField celsiusTextField = new JTextField (10);
    JLabel celsiusLabel = new JLabel ("\u00B0C");
    JButton convertButton = new JButton ("Converter");
    JLabel valorConvertidoLabel = new JLabel ("");
    Container painelDeConteudo = tela.getContentPane();
    painelDeConteudo.setLayout (new GridLayout (2, 4, 2, 4));
    painelDeConteudo.add(celsiusTextField);
    painelDeConteudo.add(celsiusLabel);
    painelDeConteudo.add(convertButton);
    painelDeConteudo.add(valorConvertidoLabel);

    convertButton.addActionListener((e) -> {
        double celsius = Double.parseDouble(
            celsiusTextField.getText()
        );
        double fahrenheit = celsius / 5 * 9 + 32;
        valorConvertidoLabel.setText(
            String.format("%.2f\u00B0F", fahrenheit)
        );
    });

    //ajusta largura e altura de acordo com o conteúdo
    tela.pack();
    //centraliza a tela
    tela.setLocationRelativeTo(null);
    //altera comportamento padrão do botão fechar
    tela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    //torna a tela visível
    tela.setVisible(true);
}
```

- Para executar a aplicação, criamos um método main e utilizamos o método `invokeLater` de `SwingUtilities` para chamar o método `criarTela` apropriadamente. Veja a Listagem 2.3.6.

Listagem 2.3.6

```
public static void main(String[] args) {
    SwingUtilities.invokeLater () -> {
        criarTela();
    });
}
```

2.4 (Criando interfaces gráficas com o NetBeans: Um sistema acadêmico) Nesta seção passaremos a utilizar o plugin de arrastar e soltar componentes do NetBeans para implementar um sistema acadêmico. O sistema irá permitir operações básicas de acesso a uma base de dados contendo dados de alunos e cursos de uma instituição de ensino.

2.4.1 (Criando o projeto) Comece criando um novo projeto/programa com nome de acordo com o contexto.

2.4.2 (Criando a tela de login) A primeira tela do sistema permitirá que o usuário insira seus dados de acesso. Para criá-la, clique com o direito no pacote principal da aplicação e escolha **New >> JFrame Form**. Seu nome será **LoginTela**.

2.4.3 (Campo para login) O usuário irá digitar seu login em um **TextField**. Note que há uma paleta de componentes à direita. Arraste um componente do tipo **TextField** para a tela. Faça os seguintes ajustes:

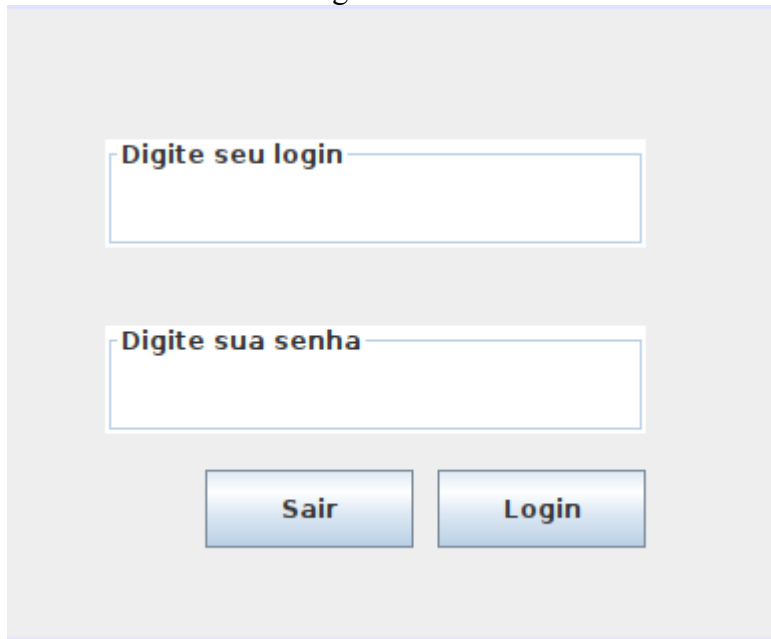
- **Largura:** 270
- **Altura:** 54
- Clique com o direito, escolha **Edit Text** e apague o texto que ele exibe por padrão.
- Clique com o direito, escolha **Change Variable Name** e digite **loginTextField**.
- Mantenha-o selecionado e veja suas propriedades na parte inferior direita da tela. Encontre a propriedade **border**. Escolha **Titled Border** e digite **Digite seu login** no campo **Title**.

2.4.4 (Campo para senha) Arraste e solte um componente do tipo **Password Field** logo **abaixo** do **loginTextField**. Faça os seguintes ajustes:

- Posição, largura e altura iguais aos do **loginTextField**.
- Clique com o direito, escolha **Edit Text** e apague o texto que ele exibe por padrão.
- Clique com o direito, escolha **Change Variable Name** e digite **senhaPasswordField**.
- Mantenha-o selecionado e veja suas propriedades na parte inferior direita da tela. Encontre a propriedade **border**. Escolha **Titled Border** e digite **Digite sua senha** no campo **Title**.

2.4.5 (Botões para sair e para fazer login) Arraste e solte dois componentes **Button** e faça ajustes para que o resultado seja parecido com o que exibe a Figura 2.4.5.1.

Figura 2.4.5.1



The image shows a login form with a light gray background. At the top, there is a text input field with the placeholder text "Digite seu login". Below it is another text input field with the placeholder text "Digite sua senha". At the bottom of the form, there are two buttons: "Sair" on the left and "Login" on the right. Both buttons have a blue gradient and a 3D effect.

- Clique com o direito em cada botão, escolha **Change Variable Name** e altere seus nomes para **sairButton** e **loginButton**.

2.4.6 (Tratando o evento “clique” dos botões) Há uma tarefa específica a ser executada quando cada um dos botões for clicado.

- No caso do botão **sairButton**, apenas desejamos encerrar a aplicação. Para isso, basta clicar duas vezes sobre ele e completar o corpo do método que irá aparecer, como mostra a Listagem 2.4.6.1. Note que o registro do observador já foi feito automaticamente.

Listagem 2.4.6.1

```
private void sairButtonActionPerformed(java.awt.event.ActionEvent evt) {  
    this.dispose();  
}
```

- Para o botão **loginButton**, iremos especificar uma lógica simples, que ainda não acessa a base:

- Primeiro, pegamos o login que o usuário digitou.
- Depois, pegamos a senha. Note que a senha é entregue como um vetor de char. É preciso converter para String.
- Depois, verificamos se ambos são iguais a **admin** (isso vai ser alterado quando passarmos a usar uma base de dados). Em caso positivo, o sistema mostra uma mensagem de boas-vindas. Caso contrário, mostra uma mensagem de usuário inválido. Veja a Listagem 2.4.6.2.

Listagem 2.4.6.2

```
private void loginButtonActionPerformed(java.awt.event.ActionEvent evt) {  
    //pega o login do usuário  
    String login = loginTextField.getText();  
    //pega a senha do usuário como char[] e converte para String  
    String senha = new String (senhaPasswordField.getPassword());  
    //verifica se o usuário é válido  
    if (login.equals("admin") && senha.equals("admin"))  
        JOptionPane.showMessageDialog (null, "Bem-vindo!");  
    else  
        JOptionPane.showMessageDialog(null, "Usuário inválido");  
}
```

2.4.7 (Ajustes finos no código gerado pelo NetBeans) Para demonstrar a possibilidade de personalização no código gerado pelo NetBeans, vamos **centralizar** a tela e adicionar um **título** à moldura. Não podemos editar o código gerado por ele. Podemos, porém, adicionar código ao construtor.

- Encontre o construtor e adicione as linhas destacadas na Listagem 2.4.7.1.

Listagem 2.4.7.1

```
public LoginTela() {  
    super ("Sistema Acadêmico");  
    initComponents();  
    this.setLocationRelativeTo(null);  
}
```

Exercícios

1. Implemente um programa com interface gráfica como o conversor de temperatura. Ele deve permitir que o usuário digite um único valor numérico. Esse valor, que será digitado em centímetros, pode ser convertido para **metros**, **milímetros** e **quilômetros**. Deve haver três botões na aplicação, um para cada possível conversão. Deve haver somente um campo de resultado.
2. Ajuste a aplicação de login e senha para que o usuário seja validado em uma base de dados que tenha uma tabela de usuários.

Referências

DEITEL, P. e DEITEL, H. **Java Como Programar**. 8ª Edição. São Paulo, SP: Pearson, 2010.

LOPES, A. e GARCIA, G. **Introdução à Programação – 500 Algoritmos Resolvidos**. 1ª Edição. São Paulo, SP: Elsevier, 2002.