Anotações da Aula

Tamer Cavalcante

Conteúdo

Capítulo 1

Introdução ao Swing

Até o momento utilizamos apenas o terminal para interagir com os usuário. Assim, para facilitar a utilização dos sistemas é necessário que a aplicação utilize uma GUI (Graphical User Interface). As GUIs apresentam mecanismos amigáveis, consistentes e intuitivos ao usuário para interagir com o sistema. Elas são construídas a partir de componentes que são objetos com o qual o usuário interage, por exemplo, o mouse e teclado.

Uma das bibliotecas gráficas suportadas pelo Java é chamada de Swing. Swing define uma coleção de classes e interfaces que dá suporte a um rico conjunto de componentes visuais. No Swing, uma GUI é composta por dois itens principais: componentes e contêineres. Um componente é um controle visual independente, como um botão ou campo de texto. Um contêiner é composto por um grupo de componentes.

Então, para que um componente seja exibido para o usuário, ele deve estar dentro de um contêiner. Logo, todas as aplicações que utilizam Swing deverão possuir pelo menos um contêiner, na qual o mais usado para é o JFrame. Além disso, todos os componentes são representados por classes definidas dentro do pacote javax.swing e começam com a letra J. Alguns exemplos de componentes Swing: JCheckBox, JRadioButton, JComboBox, JList, JTextArea, etc.

Como forma de apresentar alguns conceitos da biblioteca, vamos analisar dois exemplos. O primeiro deles apresenta uma janela na qual o usuário digita um texto e este texto é transferido para outra região da janela. O seu código está disponível em

`https://www.dropbox.com/s/jtwn6xvj7halawy/Tela1.java`

No primeiro exemplo, Figura 1.1, você pode observar dois componentes: um JTextField e um JButton. O JTextField é um componente utilizado para a criação de formulários onde é necessário a inserção de informações pelo teclado. Já o JButton é usado na criação de botões para executar ações na aplicação.

Agora vamos verificar o código dessa primeira janela apresentado na Figura 1.2. Observamos a criação do JFrame que é responsável por apresentar os componentes na janela. Para inserir componentes basta utilizar a função add().

Figura 1.1: Primeiro exemplo.

```
public class Tela1 implements ActionListener {
    JLabel label;
    JTextField  textField;
    public Tela1() {
        JFrame frame = new JFrame("Titulo");
        frame.setVisible(true);
        frame.setSize(150, 150);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLayout(new FlowLayout());

        textField = new JTextField(10);
        frame.add(textField);

        JButton button = new JButton("Executar");
        button.addActionListener(this);
        frame.add(button);

        label = new JLabel("");
        frame.add(label);
    }
    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                    new Tela1();
            }
        });
    }
    @Override
    public void actionPerformed(ActionEvent e) {
        label.setText(textField.getText());
        textField.setText("");
    }
}
```

Figura 1.2: Código fonte do primeiro exemplo.

Mas antes de adicioná-los, devemos fazer algumas configurações no JFrame, tais como: setVisible(), setSize(), setDefaultCloseOperation(), setLayout(). Recomendo você a comentar essas funções e observar o que acontece com a janela.

Em seguida, instanciamos um objeto JTextField com o tamanho de 10 colunas e adicionamos no JFrame. Logo após, criamos um objeto JButton "Executar", adicionamos uma ação nele através do addActionListener() e, por fim, inserimos o componente na janela. Para finalizar a criação da janela, criamos um JLabel que inicialmente está com o valor vazio, por esta razão que não observamos na janela no primeiro momento. Este JLabel será responsável por receber a informação inserida no JTextField.

Nosso segundo exemplo visa apresentar como diferenciar as ações dentro de uma janela. O seu código pode ser baixado atráves do link:

`https://www.dropbox.com/s/ojivv5ttxmfz708/Tela2.java`

Este exemplo possui dois JButton e um JTextField conforme apresentado na Figura 1.3. Já na Figura 1.4 é apresentado o seu código fonte.

Figura 1.3: Segundo exemplo.

```
public class Tela2 implements ActionListener {
    JLabel label;
    JTextField  textField;
    public Tela2() {
        JFrame frame = new JFrame("Titulo");
        frame.setVisible(true);
        frame.setSize(150, 150);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLayout(new FlowLayout());
        textField = new JTextField(10);
        textField.addActionListener(this);
        frame.add(textField);
        JButton button = new JButton("Botão 1");
        button.addActionListener(this);
        frame.add(button);
        button = new JButton("Botão 2");
        button.addActionListener(this);
        frame.add(button);
        label = new JLabel("");
        frame.add(label);
    }
    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                new Tela2();
            }
        });
    }
    @Override
    public void actionPerformed(ActionEvent e) {
        if(e.getActionCommand().equalsIgnoreCase("Botão 1"))
            label.setText("Botão 1");
        else if(e.getActionCommand().equalsIgnoreCase("Botão 2"))
            label.setText("Botão 2");
        else
            label.setText("Tecla Enter");
    }
}
```

Figura 1.4: Código fonte do segundo exemplo.

Observe que ao executar o exemplo e clicar em algum JButton, o valor apresentado no componente será visualizado no terminal. Além disso, caso o

cursor estiver no JTextField e você aperta a tecla Enter, isso também vai gerar uma ação. Isso ocorre porque tanto o JButton quanto o JTextField chamam o método addActionListener(). Assim, toda vez que você clicar em um botão ou pressionar o Enter, a função actionPerformed() será executada. Caso tenha mais do que uma ação para acontecer, você poderá diferenciá-las através do método getActionCommand().

Capítulo 2

Trabalhando com Imagens

Nessa seção vamos aprender como adicionar imagens nas nossas janela. Para isso, precisamos configurar o nosso projeto para salvar as imagens e depois utilizá-las no sistema. A Figura 2.1 apresenta o caminho para a criação de uma Source Folder. Ao selecionar Source Folder, você deverá colocar o nome da pasta e ela será responsável por guardar as imagens que serão utilizadas em seguida.

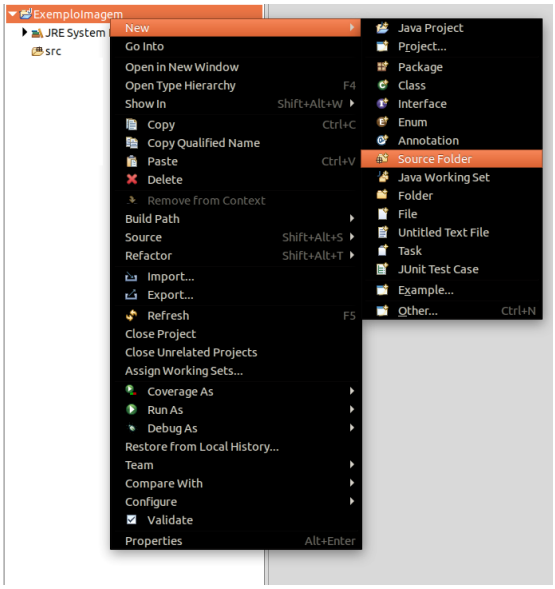


Figura 2.1: Configurando o projeto.

Após criar a pasta e salvar suas imagens nela, a estrutura do projeto ficará de acordo com a Figura 2.2, que pode ser baixado através do link :

`https://www.dropbox.com/s/jno7pn4kser7r05/Projeto.zip`

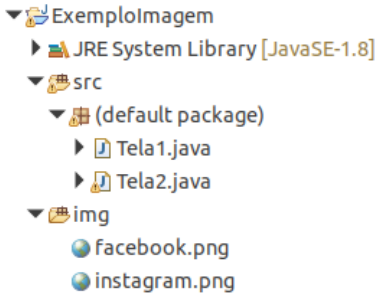
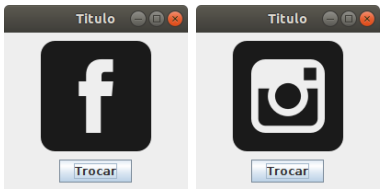


Figura 2.2: Projeto com imagens

Como vimos na seção anterior, uma interface utilizando Swing é composta por componentes e contêineres. Logo, para apresentar uma imagem, ela precisa está dentro de um componente, por exemplo, um JLabel ou um JButton. O objeto responsável por carregar uma imagem é o ImageIcon.



(a) Tela inicial. (b) Ação gerada.

Figura 2.3: Primeiro exemplo manipulando imagens.

A Figura 2.3 apresenta nosso primeiro exemplo, a Tela1.java no projeto. Nele podemos observar uma imagem e um JButton. Ao clicar no botão, a janela troca de uma imagem para a outra. Na primeira imagem temos a logo do Facebook (Figura 2.3a) e na outra a logo do Instagram (Figura 2.3b). Agora vamos analisar o código fonte da Tela1.java disponível na Figura 2.4.

Primeiramente, fizemos as configurações no JFrame: setVisible(), setSize(), setDefaultCloseOperation() e setLayout(). Em seguida, instanciamos dois objetos ImageIcon: fig1 e fig2. No construtor do ImageIcon passamos o caminho para as imagens, por exemplo, "img/facebook.png". Note que o "img/"é o nome da pasta que criamos no projeto, ver Figura 2.2. Caso você coloque outro nome na Source Folder criado, então você deverá colocar este nome no lugar do "img/".

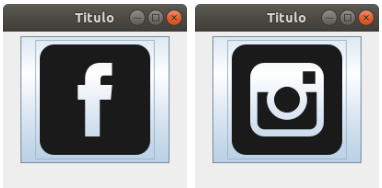
O próximo passo foi instanciar um JLabel e no seu construtor passar como parâmetro um ImageIcon, fig1. Feito isso, adicionamos o JLabel no JFrame através da função add(). Por fim, criamos um JButton com o nome "Trocar", adicionamos ação nele e inserimos no frame.

Para realizar a troca das imagens, foi necessário a criação de uma variável boolean flag que inicialmente foi atribuído o valor true. Assim todas as vezes

```java
public class Tela1 implements ActionListener {
    boolean flag = true;
    ImageIcon fig1,fig2;
    JLabel label;
    public Tela1() {
        JFrame frame = new JFrame("Titulo");
        frame.setVisible(true);
        frame.setSize(200, 200);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLayout(new FlowLayout());

        fig1 = new ImageIcon("img/facebook.png");
        fig2 = new ImageIcon("img/instagram.png");

        label = new JLabel(fig1);
        frame.add(label);

        JButton button = new JButton("Trocar");
        button.addActionListener(this);
        frame.add(button);
    }
    @Override
    public void actionPerformed(ActionEvent e) {
        if(flag)
            label.setIcon(fig2);
        else
            label.setIcon(fig1);
        flag = !flag;
    }
    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                new Tela1();
            }
        });
    }
}
```

Figura 2.4: Código fonte da Tela1.java.

que clicamos no botão "Trocar", o método actionPerformed() é executado. Portanto, verificamos o valor de flag na condicional e caso ele seja true, alteramos o label através da função setIcon() e passamos o objeto ImageIcon que instaciamos no construtor. Após a realização da troca de imagens, nós invertemos o valor da variável flag na atribuição flag = !flag;



(a) Tela inicial. (b) Ação gerada.

Figura 2.5: Segundo exemplo.

O nosso próximo exemplo é apresentado na Figura 2.5. Ele é bem parecido com o anterior mas ao invés de criar um JLabel para adicionar as imagens, nós inserimos direto no construtor do JButton. O código fonte da classe Tela2.java está disponível na Figura 2.6.

```
public class Tela2 implements ActionListener {
    ImageIcon fig1,fig2;
    JButton button;

    public Tela2() {
        JFrame frame = new JFrame("Titulo");
        frame.setVisible(true);
        frame.setSize(200, 200);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLayout(new FlowLayout());

        fig1 = new ImageIcon("img/facebook.png");
        fig2 = new ImageIcon("img/instagram.png");

        button = new JButton(fig1);
        button.addActionListener(this);
        frame.add(button);

    }
    @Override
    public void actionPerformed(ActionEvent e) {
        if(button.getIcon().toString().equalsIgnoreCase("img/facebook.png"))
            button.setIcon(fig2);
        else
            button.setIcon(fig1);

    }
    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                new Tela2();
            }
        });
    }
}
```

Figura 2.6: Código fonte da Tela2.java.

Observe na Figura 2.6 que o construtor de Tela2.java é bem parecido com o construtor da Tela1.java (Figura 2.4). A principal mudança está no mecanismo de troca entre as imagens: fig1 e fig2. Note que a função actionPerformed() usa o objeto button instanciado no construtor e chama o método getIcon(), em seguida, convertemos para uma String usando o método toString() e comparamos com o valor "img/facebook.png". Isso permite saber qual a imagem está sendo apresentada pelo botão, assim, para trocar a imagem do button, basta chamar o método setIcon() passando como parâmetro o objeto ImageIcon com a nova imagem, que no caso seria a fig2.