
una
O MELHOR
CENTRO UNIVERSITÁRIO
PRIVADO DE BH
Fonte: MEC

una | O MELHOR CENTRO UNIVERSITÁRIO PRIVADO DE BH
Fonte: MEC

Interfaces Gráficas com Swing

Professor Leonardo Larback

SWING

Swing é um widget toolkit para uso em Java. Surgiu como sucessor do Abstract Window Toolkit (AWT) à partir do java 1.2. Como uma das principais novidades, a API do Swing favoreceu ao máximo o lema de portabilidade da plataforma Java. O look-and-feel do Swing é único em todas as plataformas, assim, uma aplicação Swing terá uma interface (cores, tamanhos etc) semelhante em qualquer sistema operacional.

SWING - Componentes

Swing traz inúmeros componentes para utilizarmos na criação das interfaces gráficas dos usuários (GUI) – botões, caixas de texto, botões de seleção, caixas de opções, tabelas, janelas, abas, barras de rolagem etc. A biblioteca swing está no pacote *javax.swing*

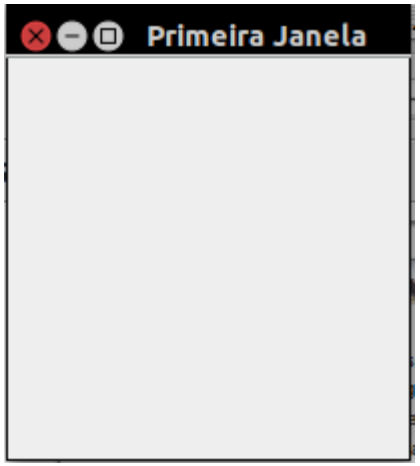
SWING – Construindo uma janela

Aplicações gráficas são construídas sobre Main Windows – uma janela top-level contendo título e borda. Esta janela, em java, é um objeto da classe JFrame. Assim, para criarmos uma janela, basta instanciarmos um objeto de JFrame e configurarmos suas propriedades utilizando seus setters:

SWING – Construindo uma janela

```
import javax.swing.JFrame;

public class Tela1 {
    public static void criaTela(){
        JFrame jfTela = new JFrame("Primeira Janela");
        jfTela.setSize(200, 200);
        jfTela.setLocationRelativeTo(null);
        jfTela.setLocation(0, 0);
        jfTela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        jfTela.setVisible(true);
    }
    public static void main(String [] args){
        criaTela();
    }
}
```



SWING – Entendendo o código

JFrame jfTela = new JFrame(“Primeira Janela”);

Nesta linha, criamos um objeto de JFrame. A string passada como parâmetro ao construtor será o título da janela.

jfTela.setSize(200, 200);

Configuramos o tamanho da janela. As medidas aqui são largura e altura.

jfTela.setLocationRelativeTo(null);

Define em relação a quem a janela será posicionada. Quando passamos null como parâmetro, a janela será posicionada em relação ao monitor do usuário.

jfTela.setLocation(0, 0);

Define a localização da janela. Os valores informados registram a distância da janela em relação à margem. O primeiro valor é a distância da margem esquerda e o segundo a distância em relação ao topo.

jfTela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

Define qual será o comportamento da janela quando a o usuário fechá-la. A constante EXIT_ON_CLOSE encerra a aplicação.

jfTela.setVisible(true);

Torna a janela visível.

SWING – Segundo exemplo

```
public class Tela2 {
    public static void criaTela(){
        JFrame jfTela = new JFrame("Primeira Janela");
        JTextField jtNome = new JTextField();
        JLabel jlNome = new JLabel("Digite seu nome:");
        JButton jbOk = new JButton("Ok");
        jfTela.setSize(400, 200);
        jfTela.setLocationRelativeTo(null);
        jfTela.setLocation (0, 0);
        jfTela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        jfTela.setLayout(null);
        jlNome.setBounds(0, 10, 120, 25);
        jfTela.add(jlNome);
        jtNome.setBounds(125,10,275,25);
        jfTela.add(jtNome);
        jbOk.setBounds(125,40, 100, 25);
        jfTela.add(jbOk);
        jbOk.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                JOptionPane.showMessageDialog(null, "Bem vindo a programação java com Swing");
                System.exit(0);
            }
        });
        jfTela.setVisible(true);
    }
    public static void main(String [] args){
        criaTela();
    }
}
```

SWING – Entendendo o código

Boa parte do código dessa segunda classe já foi explicado no primeiro, assim, nos atentemos as diferenças.

Este segundo exemplo apresenta 3 novos componentes: JButton, JLabel e JTextField – Um botão de ação, um rótulo de dados e uma caixa de texto. Esses componentes são instanciados nas linhas:

JTextField jtNome = new JTextField();

JLabel jlNome = new JLabel("Digite seu nome:");

JButton jbOk = new JButton("Ok");

Os textos informados nos construtores de JButton e JLabel serão apresentados como conteúdos dos seus respectivos objetos.

SWING – Entendendo o código

Após instanciarmos os objetos, configuramos o JFrame – Se tiver alguma dúvida sobre algum dos métodos, volte no primeiro exemplo.

Como teremos outros componentes dentro desta janela, precisamos configurar o layout que será utilizado. Neste exemplo, posicionaremos os componente manualmente, assim, definimos o layout como null.

jfTela.setLayout(null);

Para posicionarmos os componentes utilizamos o método setBounds(). Os parâmetros informados a esse métodos são distância da margem esquerda (x), distância do topo (y), largura(width) e altura(height).

jlNome.setBounds(0, 10, 120, 25);

Embora o setBounds() defina a posição do componente, ele não adiciona o mesmo na janela. Para isso, devemos utilizar o método add().

jfTela.add(jlNome);

SWING – Entendendo o código

Utilizando a mesma lógica (e os mesmos métodos), posicionamos e adicionamos todos os componentes que compõem essa tela.

```
jtNome.setBounds(125,10,275,25);

jfTela.add(jtNome);

jbOk.setBounds(125,40, 100, 25);

jfTela.add(jbOk);
```

Para definirmos uma ação para o botão, utilizamos uma classe interna anônima. Esta classe implementa a interface ActionListener e a ação é colocada dentro do método void actionPerformed(ActionEvent).

SWING – Entendendo o código

```
jbOk.addActionListener(new ActionListener() {
@Override
public void actionPerformed(ActionEvent e) {
JOptionPane.showMessageDialog(null, "Bem vindo a programação java com Swing");
System.exit(0);
}
});
```

A interface ActionListener é utilizada como um ouvinte de eventos comuns. Quase sempre que utilizamos um JButton, queremos capturar o click neste JButton. Assim, a interface ActionListener fornece um caminho mais curto para implementarmos este recurso. A interface possui apenas um método e este será executado quando o componente receber um click. A implementação feita exibe uma caixa de mensagem e encerra a aplicação.

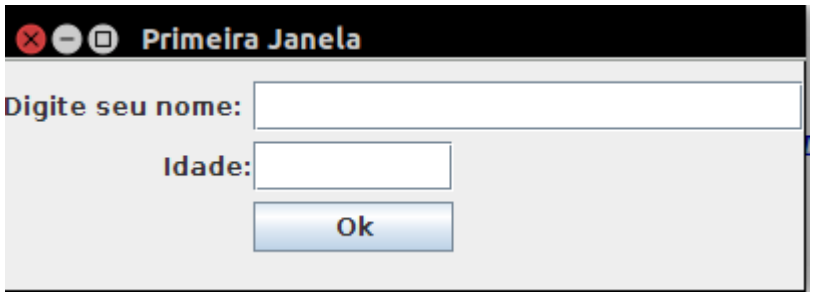
SWING – Entendendo o código

```
jbOk.addActionListener(new ActionListener() {
@Override
public void actionPerformed(ActionEvent e) {
JOptionPane.showMessageDialog(null, "Bem vindo a programação java com Swing");
System.exit(0);
}
});
```

A interface ActionListener é utilizada como um ouvinte de eventos comuns. Quase sempre que utilizamos um JButton, queremos capturar o click neste JButton. Assim, a interface ActionListener fornece um caminho mais curto para implementarmos este recurso. A interface possui apenas um método e este será executado quando o componente receber um click. A implementação feita exibe uma caixa de mensagem e encerra a aplicação.

SWING – Exercício

Para praticar, desenvolva a GUI abaixo. Sabemos que a tentação será grande, mas não copie o código dos exemplos, digite novamente cada linha – Não é desperdício de tempo, é parte do processo de aprendizagem.



Bons estudos