

II Workshop LabTEVE

Desenvolvimento de Aplicações Desktop com Swing

Aluno: **Herbet Ferreira Rodrigues**

Orientadora: **Liliane dos Santos Machado**



LabTEVE

Laboratório de Tecnologias para o
Ensino Virtual e Estatístico

Conteúdo

- **Introdução**
 - Interface Gráfica com o Usuário (GUI)
- **Swing**
 - Visão geral do Swing
 - Swing vs. AWT
 - Arquitetura MVC no Swing
 - Componentes do Swing
 - Eventos
 - Gerenciadores de Layout
- **Bibliografia**

Introdução

- A Interface Gráfica com o Usuário
 - GUI – *graphical user interface*
- Aparência e Comportamento
 - Janelas, ícones, menus
- Usabilidade
 - Facilidade de aprendizado, eficiência, facilidade de lembrar, satisfação subjetiva
- Navegabilidade
 - Elementos que facilitam interação com o usuário

Swing: Visão Geral

- Subconjunto do Java Foundation Classes (JFC)
 - AWT, Swing, Java 2D
- Tornou-se padrão a partir da versão 1.2 da plataforma Java 2
- Escrito totalmente em Java
 - Resolve problemas de portabilidade
- Conhecidos como “*lightweights*”
 - Componentes pesos-leve

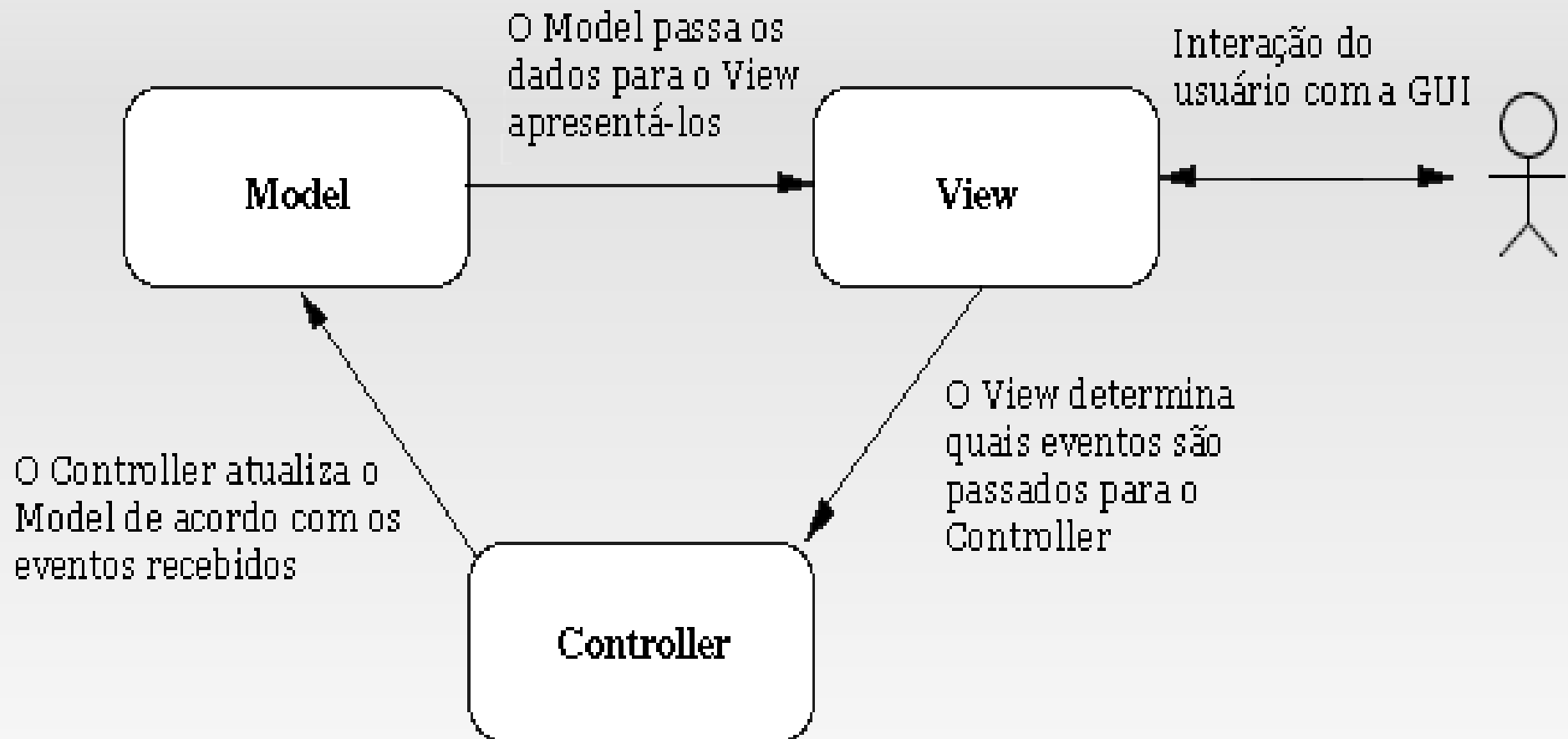
Swing vs. AWT

1. Swing possui conjunto maior de elementos
 - Maior funcionalidade
2. Swing tem controle total dos componentes
 - Controla aparência, comportamentos (*Pluggable Look-and-Feel*)
3. Swing faz a distinção entre dados dos componentes (model) e a atual visão (view)
 - Componentes flexíveis

Arquitetura MVC

- Criado para separar a lógica de negócio (model), a visão do usuário (view) e o controle da aplicação (controller)
 - **Model** – dados/conteúdo
 - Exemplo: estado de um botão, texto
 - **View** – aparência
 - Exemplo: cor, tamanho
 - **Controller** – comportamento
 - Exemplo: reação a eventos

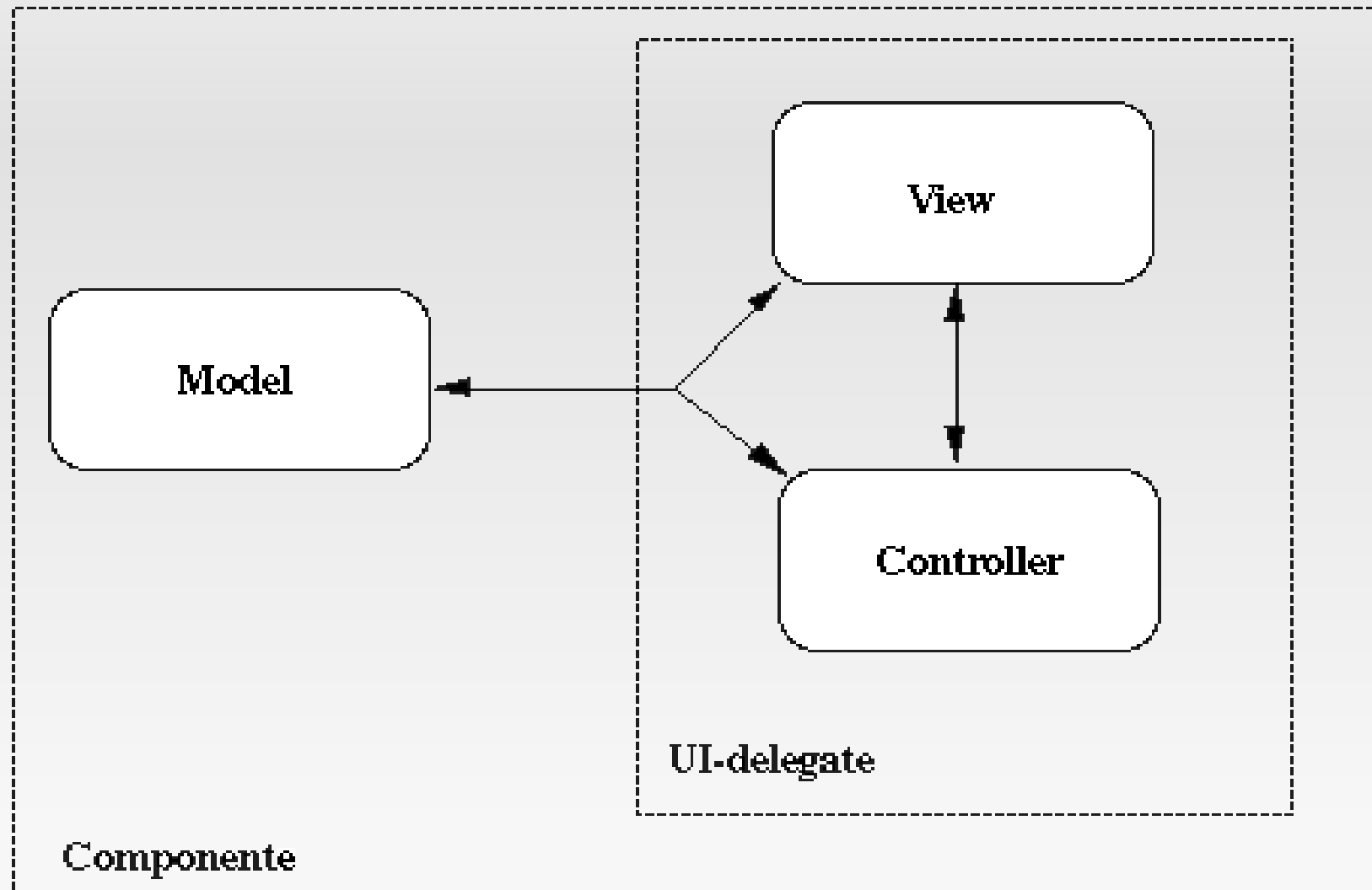
Arquitetura MVC



MVC no Swing

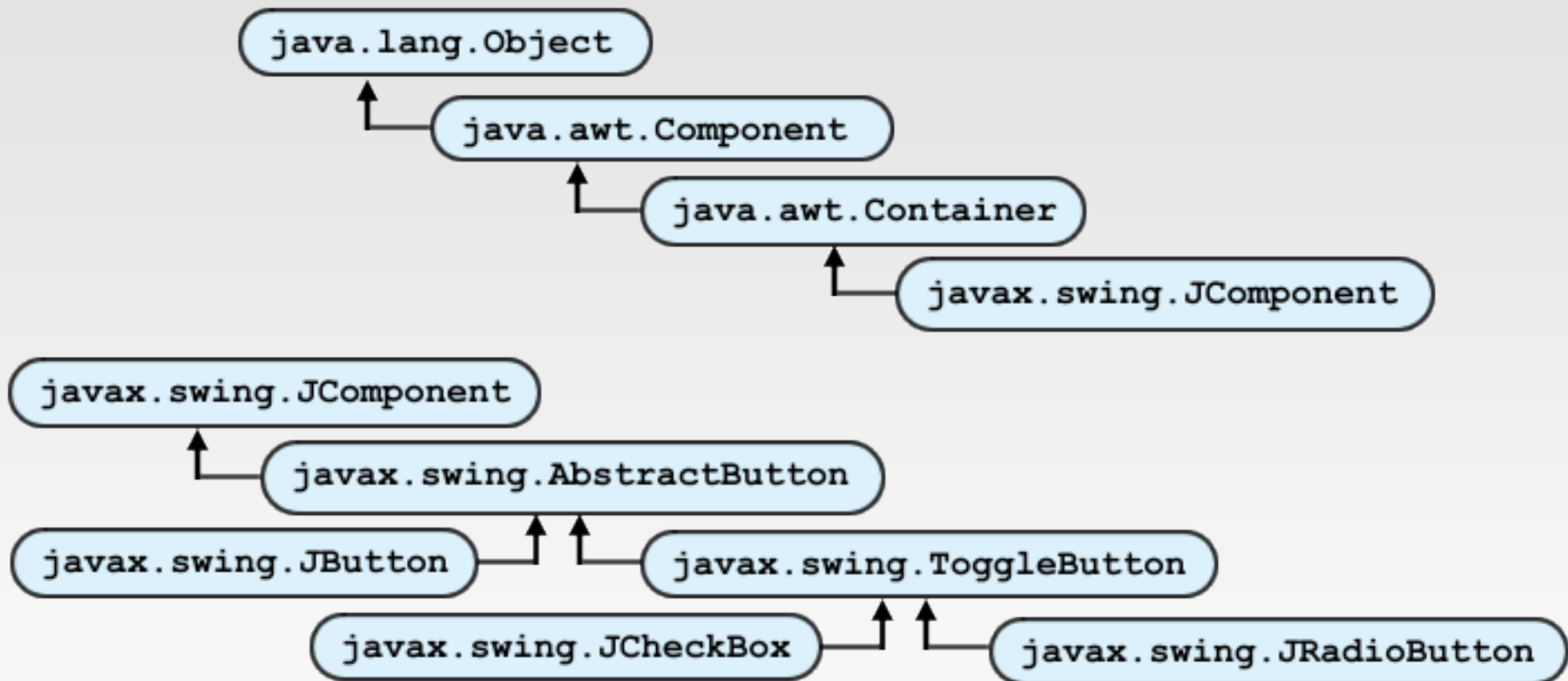
- Swing usa uma variante simplificada do MVC chamada ***model-delegate***
 - Combinados o *View* e o *Controller* em um simples elemento: ***UI-delegate***
 - Controlando a aparência e o comportamento (*look-and-feel*) de um componente como uma simples unidade
 - *Model* responsável por manter informações sobre o estado do componentes e a interface do componente na tela
 - Interagindo em duas direções

MVC no Swing



Hierarquia de Classes

- `javax.swing.*`



Interface Gráfica

- Uma interface gráfica em Java é baseada em dois elementos:
 - **Containers** – servem para agrupar e exibir outros componentes
 - Exemplos: JFrame, JDialog, JApplet
 - **Components** – botões, labels, scrollbars, *etc*
 - JLabel, JButton, JCheckBox, JList

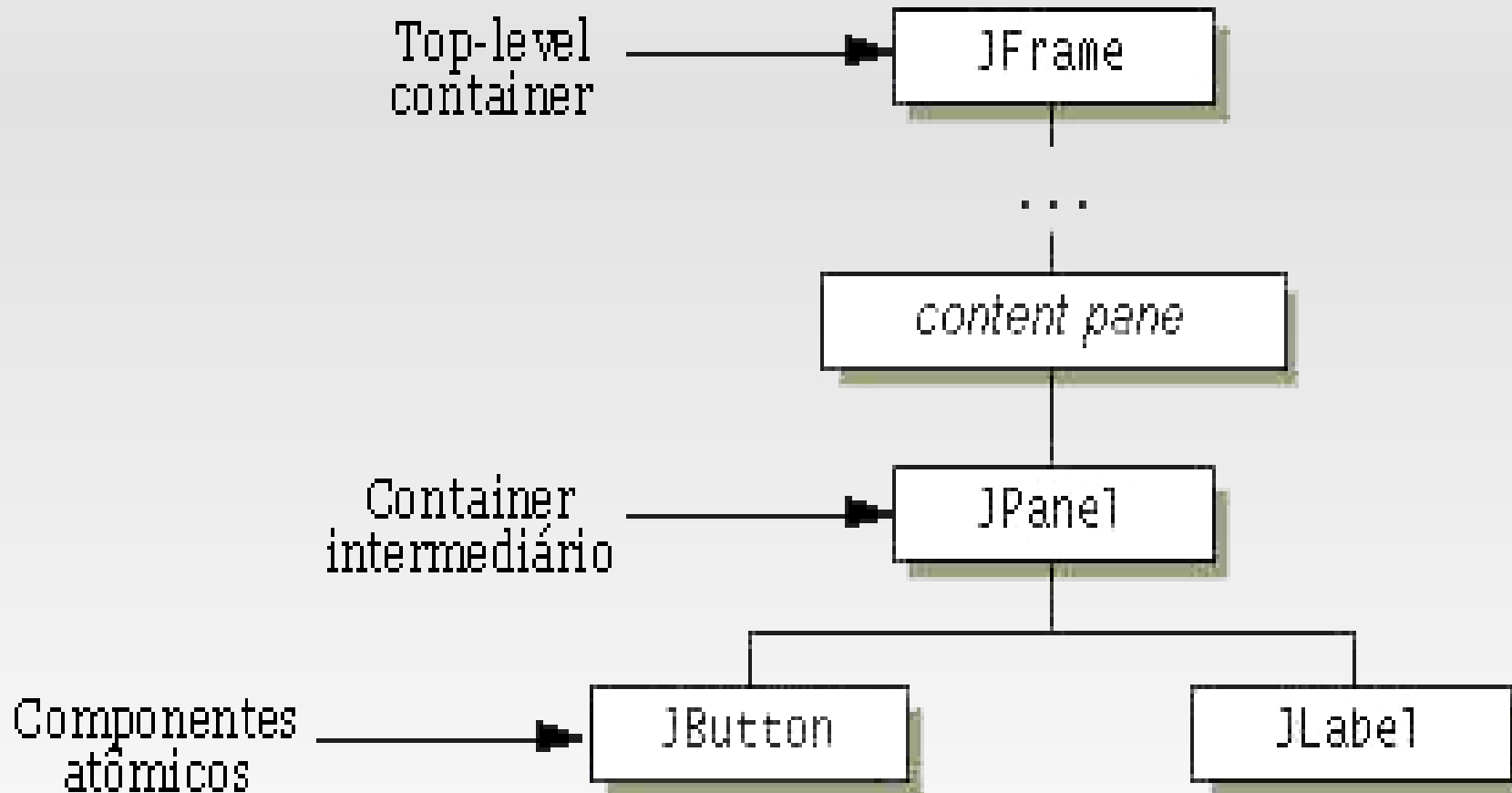
Interface Gráfica (2)

- Implementação envolve três etapas:
 - 1) Instanciar componentes da interface
 - Botões, Textos, Tabelas, etc
 - 2) Adicionar os componentes ao container
 - 3) Estabelecer o tratamento de eventos da interface
 - Exemplo: o que deve acontecer quando o usuário clicar em um botão, ou se um usuário seleciona um ítem de menu.

Top-Level Containers

- Uma aplicação desktop simples em Swing é constituída tipicamente pelos elementos:
 - Janela (***top-level container***): lugar onde os outros componentes serão desenhados
 - Painel (***container intermediário***): facilita o agrupamento de outros componentes
 - Botões e Campos de texto (***componentes atômicos***): elementos da interface que não agrupam outros componentes. Servem para responder ao usuário.

Swing: Hierarquia Básica



JFrame

- Um objeto JFrame representa uma janela do sistema gráfico que usamos para construir nossas aplicações desktop
- Pode ser formado por diversos componentes:
 - Botões (JButton)
 - Labels (JLabel)
 - CheckBoxes (JCheckBox)
 - Campo de Texto (JTextField)
 - Tabelas (JTable)
 - *etc*

JFrame: Exemplo

```
import java.awt.Container;
import javax.swing.*;

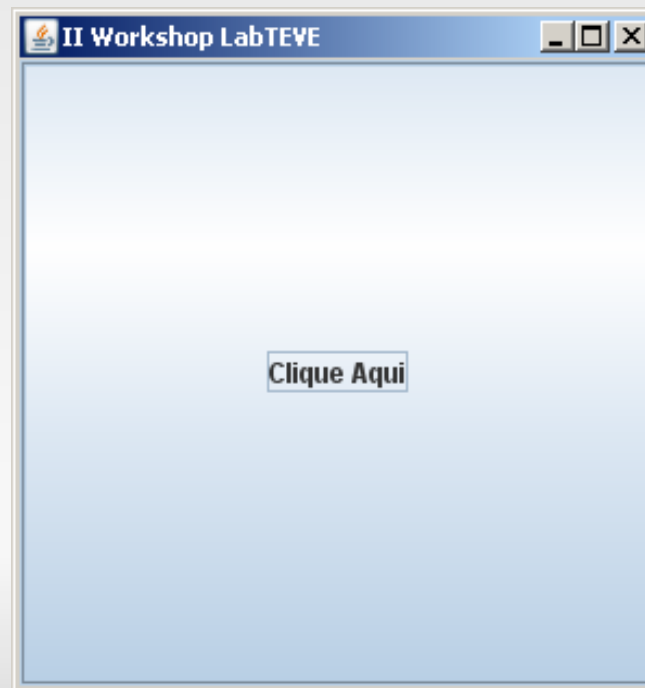
public class JanelaPrincipal extends JFrame {

    public JanelaPrincipal() {
        // Título da janela
        super("II Workshop LabTEVE");
        // Finaliza o programa se a janela for fechada
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        // Define container principal da janela
        Container content = getContentPane();
        // Cria um novo botão
        JButton botao = new JButton("Clique Aqui");
        // Adiciona o botão ao container
        content.add(botao);
        // Define largura e altura da janela
        setSize(300, 300);
    }

    public static void main(String[] args) {
        // Instancia objeto do tipo JanelaPrincipal
        JanelaPrincipal janela = new JanelaPrincipal();
        // Permite a visualização da janela
        janela.setVisible(true);
    }
}
```

JFrame: Exemplo (2)

- Exibe uma janela com um botão ocupando todo o espaço
- Quando clica no botão, nada acontece. Apenas mostra que ele está sendo clicado.



JFrame: Exemplo (3)

- O que fazer para algo executar quando o botão for clicado?
- Necessitamos de duas coisas:
 1. Um método que execute a ação
 2. Algo que identifique que o botão foi clicado:
 - **EVENTO**

Evento

- Interação do mundo externo com a aplicação, indicando que alguma coisa aconteceu
 - Programa passa a ser chamado quando algum evento é gerado na interface da aplicação
- Exemplos de Eventos:
 - Clique no botão
 - Botão está sendo pressionado
 - Clique no mouse
 - Movimento do mouse
 - *etc*

Evento (2)

- Para processar um evento:
 - Registrar um ouvinte de eventos (*listener*)
 - Objetos de um classe que implementa uma interface *event-listener* (`java.awt.event` ou `javax.swing.event`)
 - Implementar um tratador de eventos (*handler*)
 - Método chamado em resposta ao evento

Evento: Exemplo

```
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.*;

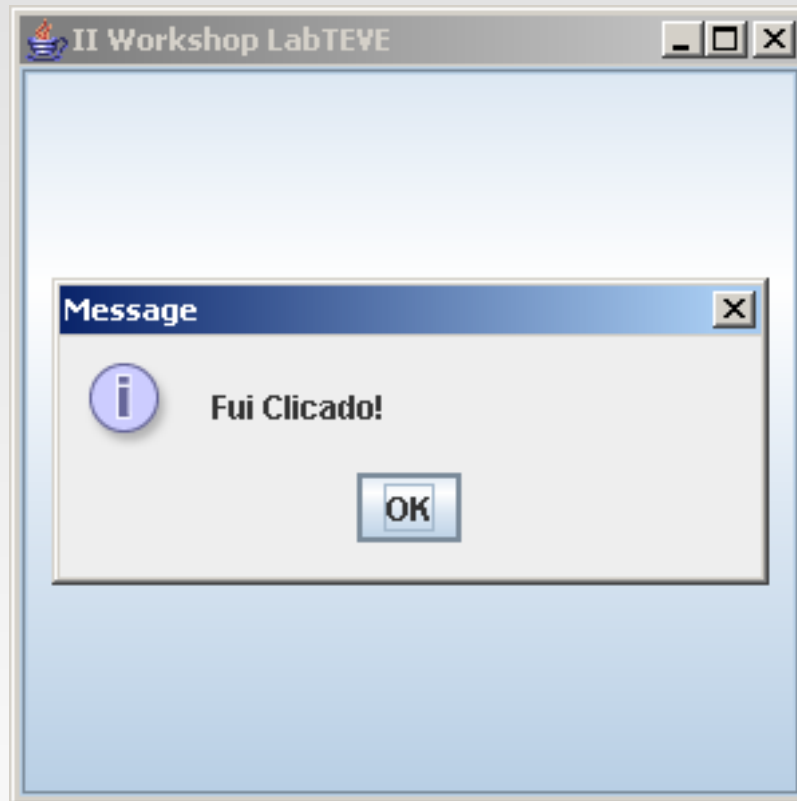
public class JanelaPrincipal extends JFrame {

    public JanelaPrincipal(){
        /* ... */
        JButton botao = new JButton("Clique Aqui");
        // Registra o ouvinte que chamará o evento
        botao.addActionListener( new BotaoClicado(this));
    }
    // Classe que implementa a interface ActionListener
    private class BotaoClicado implements ActionListener {
        private JFrame janela;

        BotaoClicado(JFrame janela){
            this.janela = janela;
        }
        // Método que realiza o tratamento do evento
        public void actionPerformed(ActionEvent e) {
            JOptionPane.showMessageDialog(janela, "Fui Clicado!");
        }
    }
}
```

Evento: Exemplo (2)

- Execução do exemplo:



Evento (4)

- **ActionListener** – tratamento de eventos provenientes de ações do usuário em objetos das classes JButton, JMenuBar, entre outros.
- **ItemListener** – tratamento de eventos sobre objetos que significam itens de menu, como JCheckBox, JRadioButton, JComboBox
- **MouseListener** – tratamento de eventos do mouse, tais como clicks, coordenadas e outros
- **KeyListener** – tratamento de evento do teclado

Gerenciadores de Layout

- Determinam o tamanho e a posição dos componentes em um container ou painel
- Organização dos componentes em diferentes plataformas
- Posicionam os componentes seguindo uma regra
- Utiliza o método *add()* para adicionar o componente ao container ou painel

Gerenciadores de Layout (2)

- **FlowLayout** - Coloca os componentes seqüencialmente da esquerda para a direita na ordem em que foram adicionados



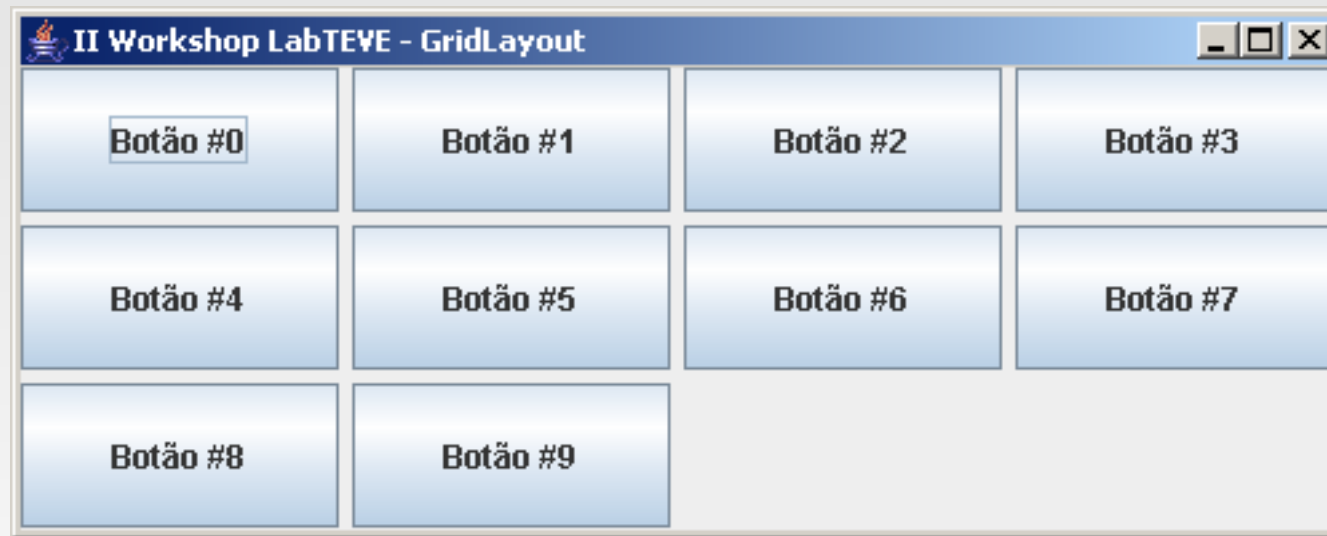
Gerenciadores de Layout (3)

- **BorderLayout** - Organiza os componentes em cinco áreas: norte, sul, leste, oeste e centro



Gerenciadores de Layout (4)

- **GridLayout** - Organiza os componentes nas linhas e colunas



Fim

Perguntas?

Referências

- [1] Java, como programar / H. M. Deitel e P. J. Deitel; trad. Carlos Arthur Lang Lisboa. - 4.ed. - Porto Alegre : Bookman, 2003.
- [2] M., Márcia T. Baptistella. “A importância de interfaces acessíveis em sistemas computacionais”. Araçatuba – UniToledo, São Paulo.
- [3] SwingWiki: Java Developer Wiki.
[http://www.swingwiki.org/table_of_contents.](http://www.swingwiki.org/table_of_contents)