Universidade Estadual de Maringá - UEM
Departamento de Informática
Bacharel em Engenharia de Produção

Programação de Sistemas: GUI com Swing e AWT

Prof. Ms. Ricardo Theis Geraldi

Introdução

- Java fornece um forte suporte para a implementação de interfaces gráficas por meio dos pacotes **java.awt** e **javax.swing**;

- Substituída pela **JFC** (Java Foundation Classes) (**AWT e Swing**), compartilham partes fundamentais (eventos).

Elementos Gráficos

- **JFC** (Java Foundation Classes). É um conjunto de aspectos (componentes) implementados e disponibilizados em uma biblioteca que serve para criar interfaces gráficas de usuário.

Elementos Gráficos

- Os elementos chaves de uma interface gráfica em Java são compostos por:
- **Componentes Gráficos** (JFrame; JDialog; JOptionPane e muitos outros);
- **Gerenciadores de Layout** (GridLayout, FlowLayout, BorderLayout etc.);
- **Processamento de Eventos** (Listeners).

Elementos Gráficos

- **Componentes Gráficos**, tais como, janelas (JFrame e JDialog), mensagens (JOptionPane), campo texto (JTextField) e os botões (JButton), são elementos que o usuário manipula com o mouse ou com o teclado;

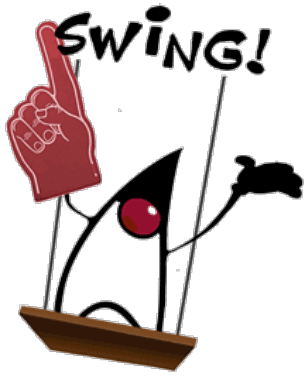
- **Gerenciadores de Layout** governam a maneira pela qual os componentes aparecem na tela;

- **Eventos** assinalam ações do usuário consideradas importantes, como o clique de um mouse em cima de um botão de fechar de um janela ou botão.

GUI - Graphical User Interfaces

- GUI - Interface Gráfica do Usuário.

- Em Java as interfaces gráficas podem ser projetadas por meio da utilização dos pacotes: **java.awt** (AWT) e **javax.swing** (Swing).



Criando Aplicações Gráficas AWT

- A criação de interfaces gráficas AWT consiste na criação (instância) de objetos do tipo *Component* (botões, textos etc.) e na criação de recipientes ou objetos da classe *Container* (janelas, painéis etc.)
- Os Gerenciadores de Layouts se encarregam de definir a localização e aspecto visual dos componentes (*Component*) inseridos nos recipientes (*Container*).

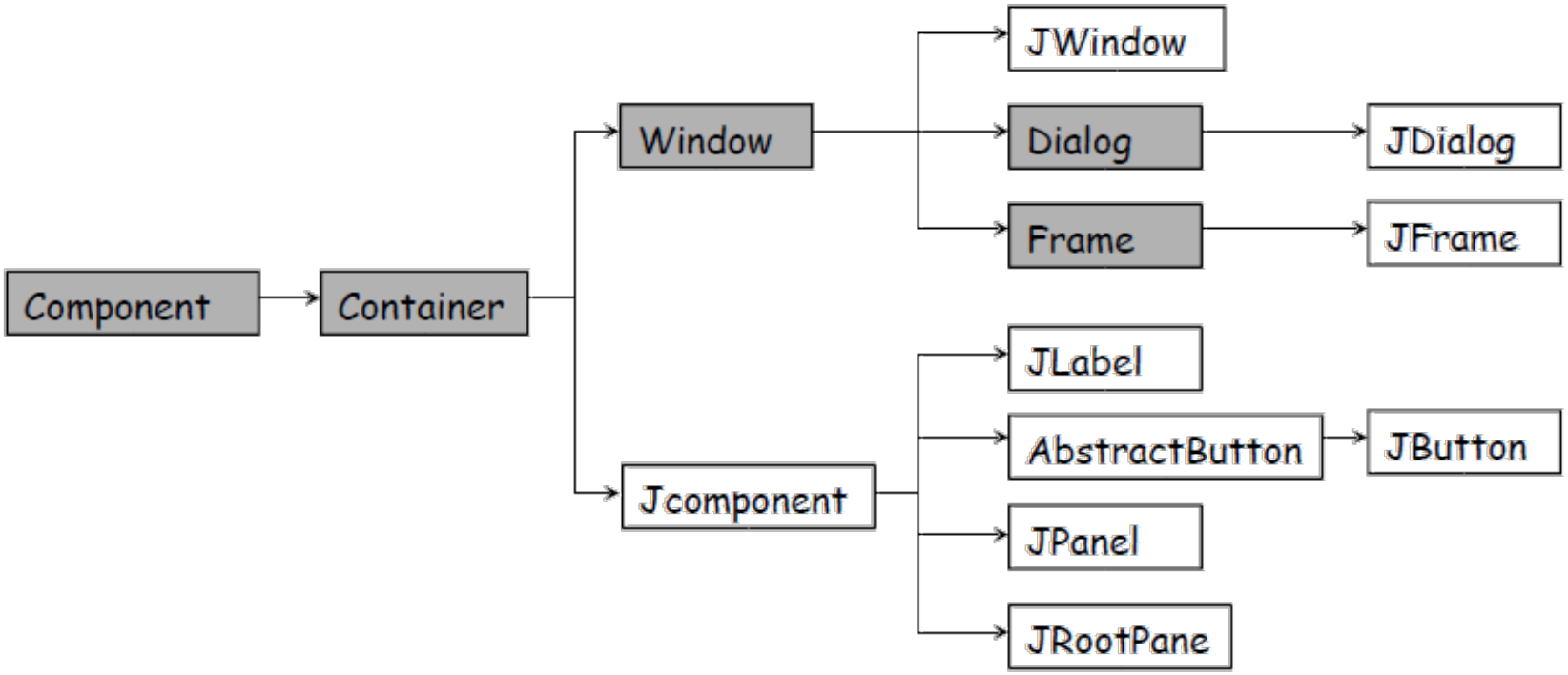
Container

- Um *Container* é uma **categoria especial de componente gráfico** que **pode conter outros componentes ou mesmo outros *Containers***.
- **Todos os *Containers* são componentes, mas nem todos os componentes são *Containers***.
- Cada *Container* possui **associado um Gerenciador de Layout** para controlar a maneira pela qual seus componentes serão mostrados (tamanho e posição).

Window

- Uma *Window* é uma **janela sem barra de título e borda** e que necessariamente tem que estar **associada com outro objeto da classe Frame** (janela) para que possa existir.
- **A classe *Window* especializa a classe *Container***, logo poderá conter outros componentes internamente.
- A classe *Window* normalmente é utilizada para implementar janelas ***pop-up***.
- Quando um objeto da classe *Window* é criado, automaticamente é associado a ele o gerenciador de layout *BorderLayout*.

Estrutura Hierárquica AWT e Swing



- Os retângulos cinzas representam os controles AWT

Estrutura Hierárquica AWT e Swing

- Alguns componentes Swing, atualmente, são derivados de componentes AWT (*Abstract Window Toolkit)*.
- **Exemplos:**
  - Frame (AWT) para JFrame (Swing);
  - Dialog (AWT) para JDialog (Swing).

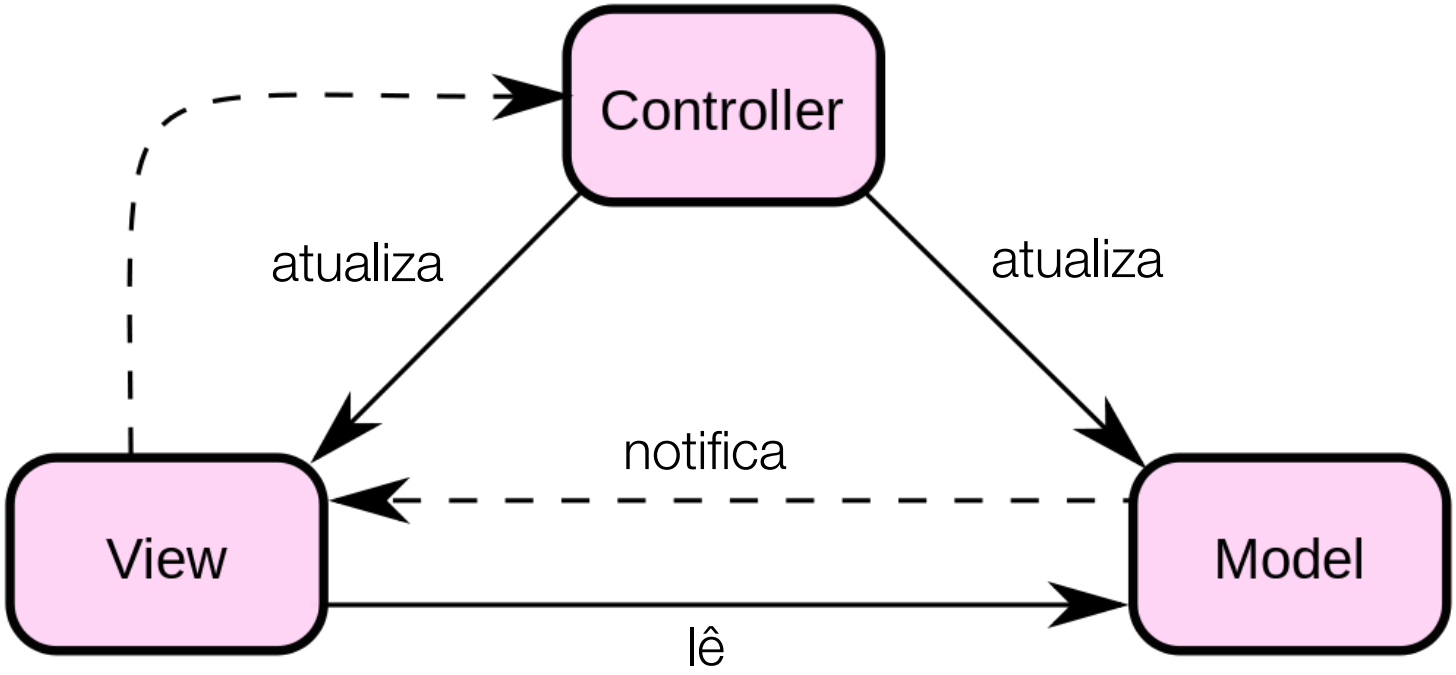
Swing

- Componentes leves;
- Não substitui integralmente o AWT:
  - Estende o AWT;
  - Diagramadores e *Listeners.*
- *Look & Feel* configurável (skins de aparência):
  - Windows, Motif, Metal, Nimbus.
- Double-buffering automático;
- Arquitetura MVC.

MVC Swing

- O Swing adota uma arquitetura conhecida como *Model-View-Controller* (MVC)
  - *Model* (Modelo) = dados/conteúdo (estado de um botão, texto);
  - *View* (Visão) = aparência (cor, tamanho, etc.);
  - *Controller* (Controlador) = comportamento (eventos).

MVC Swing



MVC Swing

- O objeto *view* registra-se como ouvinte (*listener*) no *model*. Qualquer alteração nos dados do *model* provoca uma notificação a todos os ouvintes registrados. Assim, uma notificação sempre será recebida pelo *view*;

- O *controller* fica atrelado ao *view*. Ou seja, qualquer ação do usuário que for feita no *view* chamará um método *listener* na classe *controller*;

- O *controller* recebe uma referência do *model* subjacente. Desta forma, o *controller* poderá chamar métodos do *model*.

Migração AWT para Swing

- Os desenvolvedores gastavam muito tempo e esforço com os defeitos do AWT.
- A meta do projeto Swing foi acrescentar novas funcionalidades, por meio de uma biblioteca de classes, a fim de suprir as demandas dos usuários.

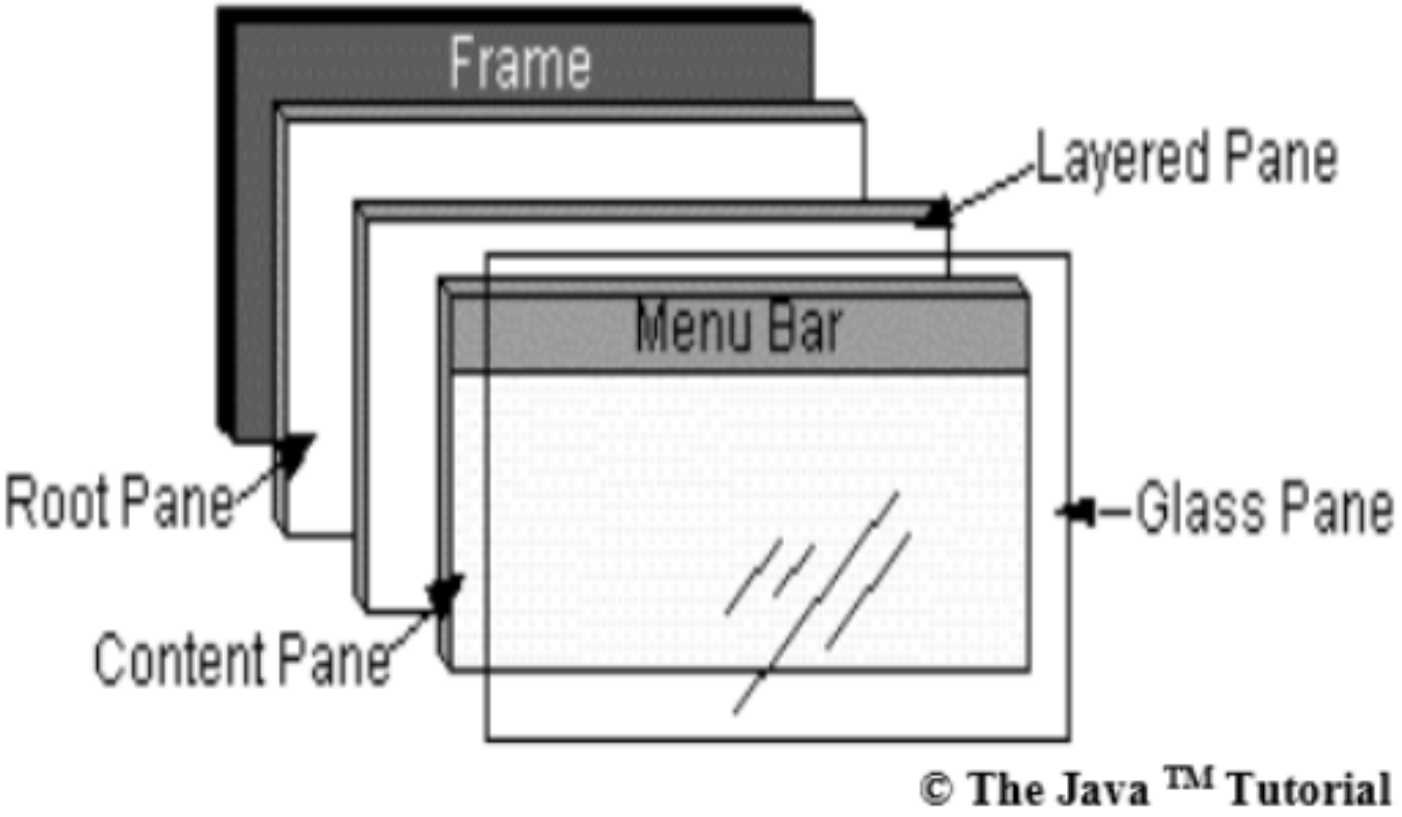
Popularidade Swing

- É possível utilizar alguns componentes básicos do Swing de forma direta;
- Com o Swing não há mudanças fundamentais no modo como as aplicações são construídas;
- Pode-se criar a janela principal da aplicação, usar *frames, components* e gerenciadores de layout e é possível estabelecer conexão quase que da mesma forma.

Classe JFrame - Swing

- No Swing, a interface principal da aplicação é implementada pelo componente **javax.swing.JFrame**.
- JFrame é derivado do Frame da AWT

Estrutura JFrame - Swing



Camadas JFrame - Swing

- RootPane
  - Gerencia as demais camadas;
  - Botão "*default*".
- LayeredPane
  - Contém a *menu bar* e o ContentPane;
  - Pode conter subcamadas.
- ContentPane
  - Contém os componentes visíveis.
- GlassPane
  - Invisível por *default;*
  - Interceptação de eventos/pintura sobre uma região.

JFrame - TelaSimples1

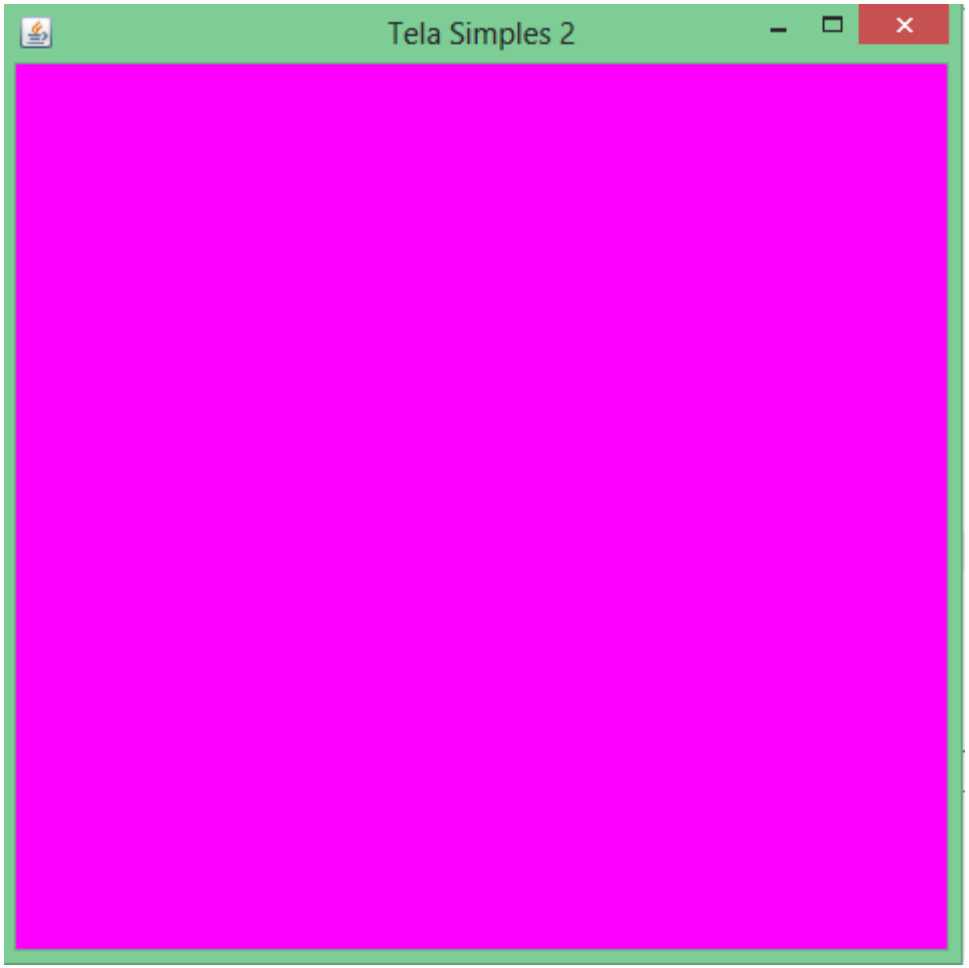
```
package br.com.aulagui.view.jframe;

import java.awt.Color;
import javax.swing.JFrame;

public class TelaSimples1 extends JFrame {

    public static void main(String[] args) { // Define a classe como principal
        TelaSimples1 tsum = new TelaSimples1("Tela Simples 1"); // Instância a TelaSimples1
    }

    TelaSimples1 (String title) {
        super(title); // Define o título da tela
        // setTitle("Tela Simples 1"); // Outra forma para definir o título da tela
        setSize(640, 480); // Estipula a largura e a altura da tela
        setVisible(true); // A tela está visível true = verdadeiro
        getContentPane().setBackground(Color.cyan); // Define a cor de fundo da tela
        setDefaultCloseOperation(EXIT_ON_CLOSE); // Fecha a tela e termina o aplicativo
    }
}
```

JFrame - TelaSimples1

JFrame - TelaSimples2

```
package br.com.aulagui.view.jframe;

import java.awt.Color;
import javax.swing.JFrame;
import static javax.swing.JFrame.EXIT_ON_CLOSE;

public class TelaSimples2 {

    public static void main(String[] args) { // Define a classe como principal
        JFrame tsdois = new JFrame(); // Instância a TelaSimples2

        tsdois.setTitle("Tela Simples 2"); // Define o título da tela
        tsdois.setSize(480, 480); // Estipula a largura e a altura da tela
        tsdois.getContentPane().setBackground(Color.magenta); // Define a cor de fundo da tela
        tsdois.setVisible(true); // A tela está visível true = verdadeiro
        tsdois.setLocationRelativeTo(null); // Centraliza o tela na tela
        tsdois.setDefaultCloseOperation(EXIT_ON_CLOSE); // Fecha a tela e termina o aplicativo
        // setExtendedState(MAXIMIZED_BOTH); // Inicializa a tela Maximizada
    }
}
```

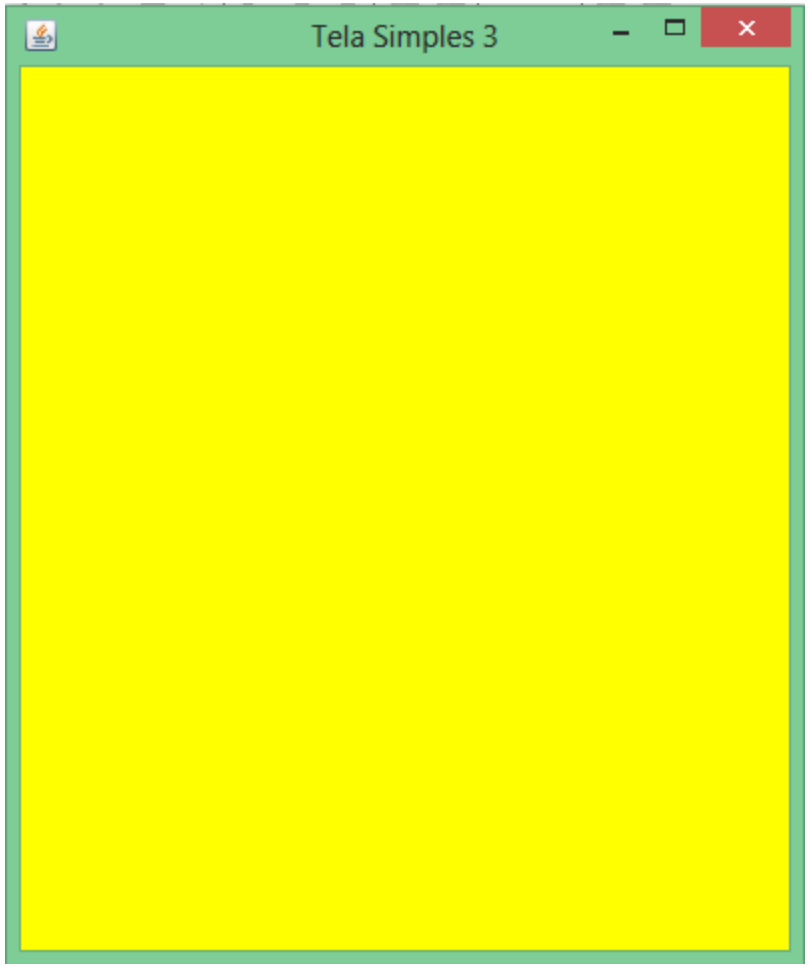
JFrame - TelaSimples2



JFrame - TelaSimples3

```
package br.com.aulagui.view.jframe;

import java.awt.Color;
import javax.swing.JFrame;

public class TelaSimples3 extends JFrame {

    public static void main(String[] args) { // Define a classe como principal
        TelaSimples3 tstres = new TelaSimples3(); // Instância a TelaSimples3

        tstres.setTitle("Tela Simples 3"); // Define o título da tela
        tstres.setSize(400, 480); // Estipula a largura e a altura da tela
        tstres.setVisible(true); // A tela está visível true = verdadeiro
        tstres.getContentPane().setBackground(Color.yellow); // Define a cor de fundo da tela
        tstres.setLocationRelativeTo(null); // Centraliza o tela na tela
        tstres.setDefaultCloseOperation(EXIT_ON_CLOSE); // Fecha a tela e termina o aplicativo
    }
}
```

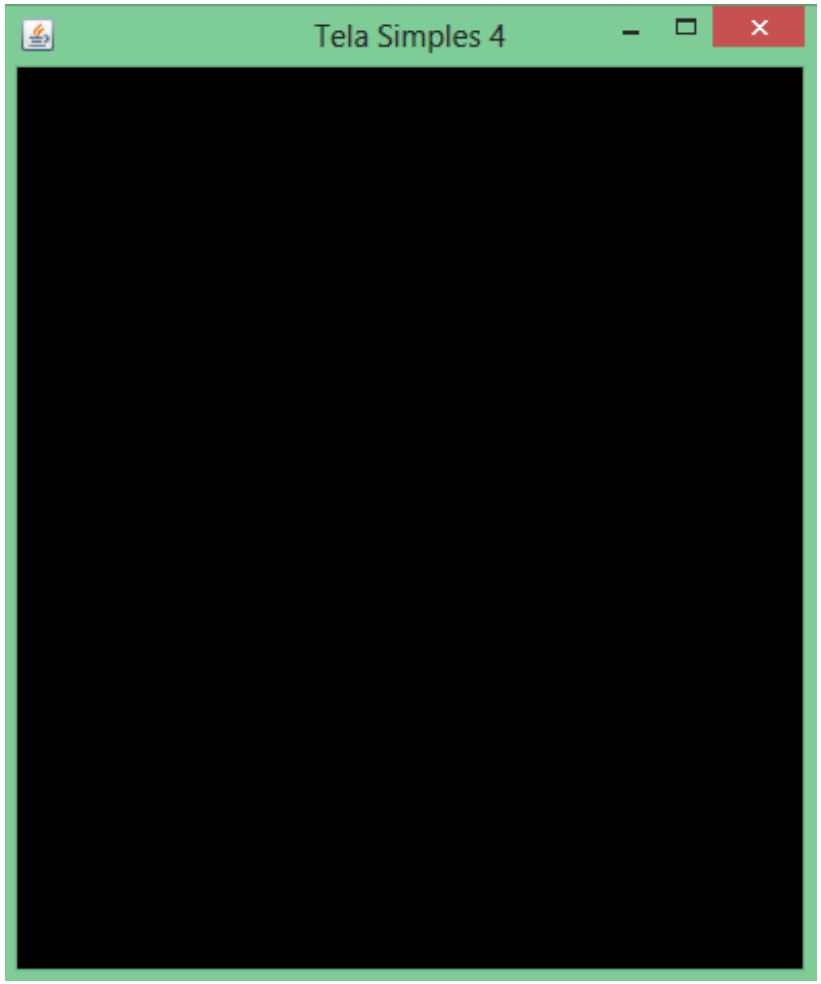
JFrame - TelaSimples3



JFrame - TelaSimples4

```
package br.com.aulagui.view.jframe;

import java.awt.Color;
import javax.swing.JFrame;

public class TelaSimples4 extends JFrame {

    public static void main(String[] args) { // Define a classe como principal
        TelaSimples4 tsquatro = new TelaSimples4("Tela Simples 4"); // Instância a TelaSimples4

        tsquatro.setSize(400, 480); // Estipula a largura e a altura da tela
        tsquatro.getContentPane().setBackground(Color.black); // Define a cor de fundo da tela
        tsquatro.setVisible(true); // A tela está visível true = verdadeiro
        tsquatro.setDefaultCloseOperation(EXIT_ON_CLOSE); // Fecha a tela e termina o aplicativo
    }

    TelaSimples4 (String titulo) {
        super(titulo);
    }
}
```

JFrame - TelaSimples4



JFrame - ChamarMinhaTela

```
package br.com.aulagui.view.jframe;

public class ChamarMinhaTela {

    public static void main(String[] args) { // Define a classe como principal
        MinhaTelaExemplo mte = new MinhaTelaExemplo("Minha Tela"); // Define a classe como principal
    }
}
```

- Classe "MinhaTelaExemplo“ instânciada acima na classe "ChamarMinhaTela"

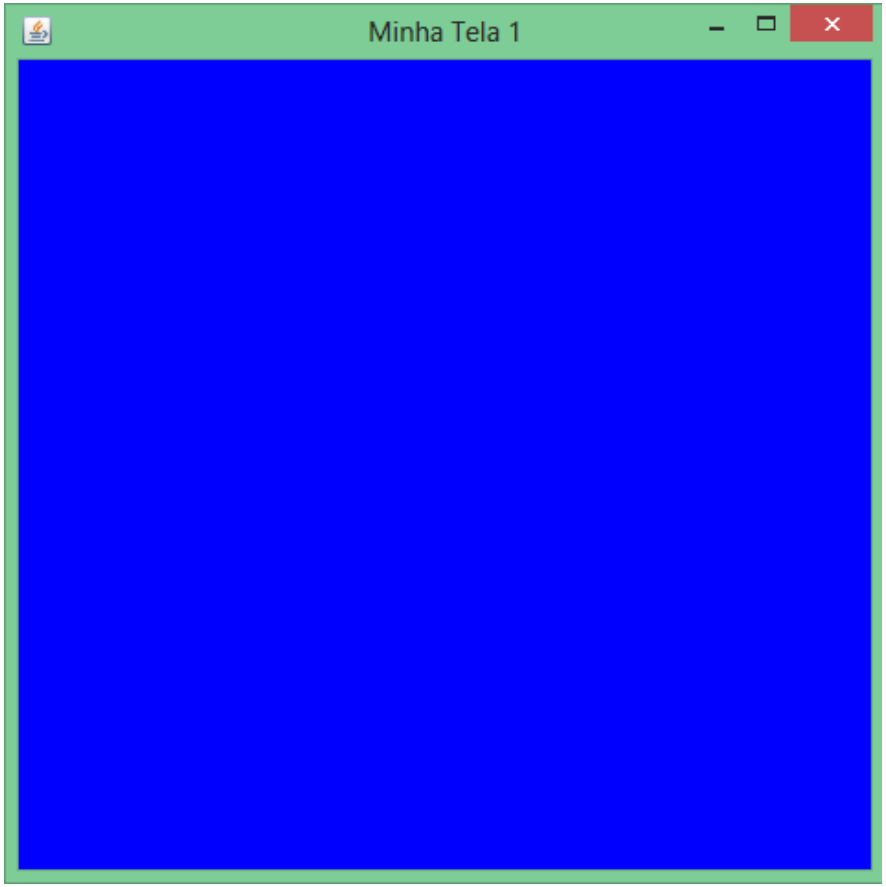
```
public class MinhaTelaExemplo extends JFrame {
    MinhaTelaExemplo (String titulo) {
        super(titulo);
        setSize(640, 480); // Estipula a largura e a altura da tela
        getContentPane().setBackground(Color.orange); // Define a cor de fundo da tela
        setVisible(true); // A tela está visível true = verdadeiro
        setDefaultCloseOperation(EXIT_ON_CLOSE); // Fecha a tela e termina o aplicativo
    }
}
```

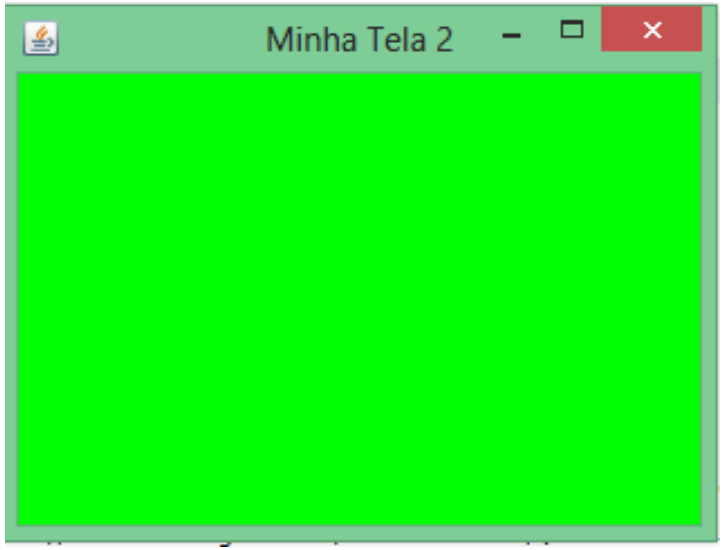
JFrame - ChamarMinhaTela

JFrame - MinhaTela

```
package br.com.aulagui.view.jframe;

import java.awt.Color;
import javax.swing.JFrame;

public class MinhaTela extends JFrame {

    public static void main(String[] args) { // Define a classe como principal
        MinhaTela mt1 = new MinhaTela();
        mt1.setTitle("Minha Tela 1"); // Define o título da tela
        mt1.setSize(480, 480); // Estipula a largura e a altura da tela
        mt1.getContentPane().setBackground(Color.blue); // Define a cor de fundo da tela
        mt1.setVisible(true); // A tela está visível true = verdadeiro

        MinhaTela mt2 = new MinhaTela();
        mt2.setTitle("Minha Tela 2"); // Define o título da tela
        mt2.setSize(320, 240); // Estipula a largura e a altura da tela
        mt2.getContentPane().setBackground(Color.green); // Define a cor de fundo da tela
        mt2.setLocationRelativeTo(null); // Centraliza o tela na tela
        mt2.setVisible(true); // A tela está visível true = verdadeiro
        // mt2.toBack(); // Posiciona a tela atrás da outra
```

JFrame - MinhaTela





JFrame - Exercícios

- Conforme os exemplos apresentados, implemente JFrame's no NetBeans, considerando pelo menos os métodos referentes ao:
  - título da janela;
  - tamanho da janela;
  - cor de fundo da janela;
  - se a janela está visível ou não para o usuário;
  - se janela inicia ou não centralizada na tela; e
  - se quando a janela é fechada finaliza o aplicativo.

JDialog - Swing

- Um JDialog é uma janela com uma barra de título;
- Caixas de diálogos são em geral usadas para obter/mostrar informações do/para o usuário;
- O Swing fornece um rico conjunto de diálogos que permite interações básicas com o usuário sem a necessidade de escrever muito código.

JDialog - Swing

- Caixas de diálogos podem ser modais ou não-modais:
  - Caixas modais não permitem que outras janelas sejam acessadas até que a caixa de diálogo seja fechada.
    - Ex.: Janela de confirmação de exclusão de arquivos no Windows
  - Caixas não-modais permitem que outras janelas sejam manipuladas concomitantemente a janela de diálogo.

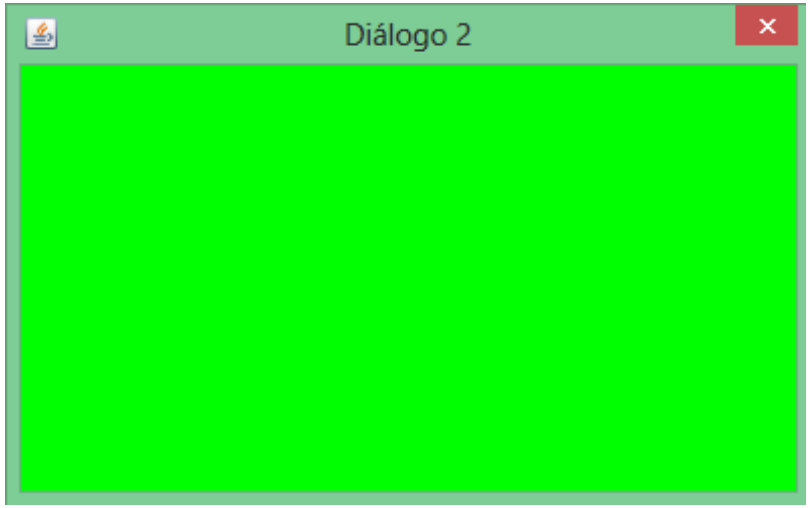
JDialog - Swing

- BorderLayout é o gerenciador de layout usado implicitamente por caixas de diálogos;
- JDialog utiliza uma estrutura de camadas como a classe JFrame.

JDialog - Exemplo1

```
package br.com.aulagui.view.jdialog;

import java.awt.Color;
import javax.swing.JDialog;

public class Exemplo1 extends JDialog {

    public static void main(String[] arts) { // Define a classe como principal
        Exemplo1 ex1 = new Exemplo1(); // Instância a classe referente ao diálogo Exemplo1
        ex1.setTitle("Diálogo 1"); // Define o título da diálogo
        ex1.setSize(600,200); // Define a largura e a altura do diálogo
        ex1.setModal(true); // Se o diálogo é modal (true) ou não (false)
        ex1.getContentPane().setBackground(Color.red); // Define a cor de fundo do diálogo
        ex1.setVisible(true); // O diálogo está visível true = verdadeiro

        Exemplo2 ex2 = new Exemplo2(); // Instância a classe referente ao diálogo Exemplo2
        ex2.setTitle("Diálogo 2"); // Define o título da diálogo
        ex2.setSize(400,250); // Define a largura e a altura do diálogo
        ex2.isModal(); // Método para confirmar se o diálogo é ou não modal
        ex2.getContentPane().setBackground(Color.green); // Define a cor de fundo da tela
        ex2.setVisible(true); // O diálogo está visível true = verdadeiro
    }
}
```

JDialog - Exemplo1

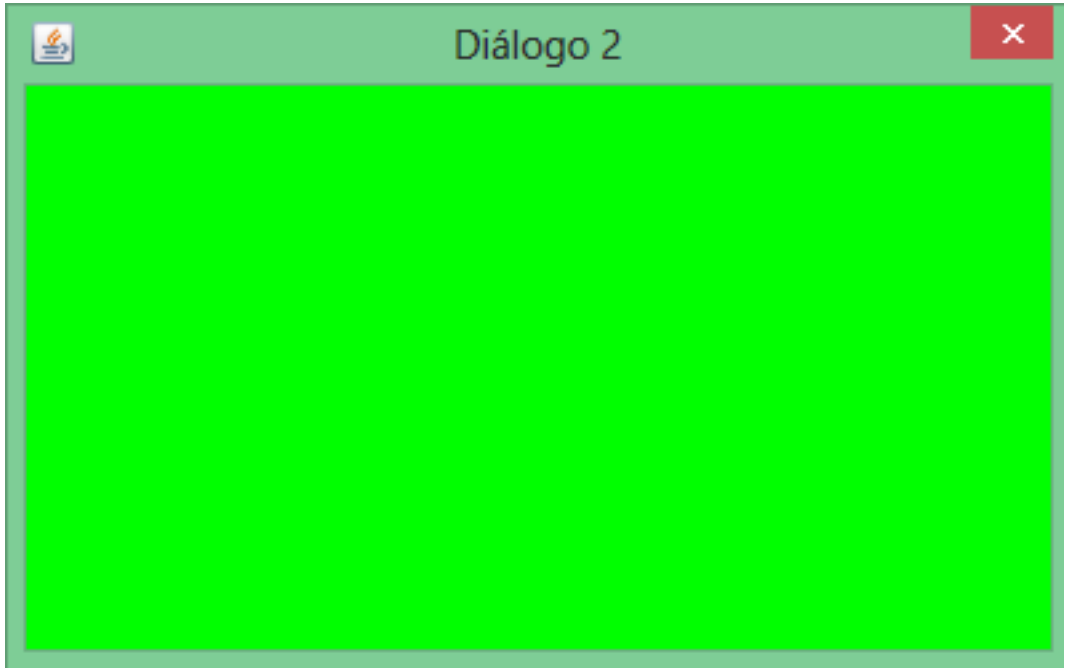


Temos dois diálogos modais:

O **Diálogo 1** deve ser fechado para que o **Diálogo 2** apareça.

JDialog - Exemplo1

JDialog - Exemplo1



JDialog - Exemplo2

- A classe "Exemplo2“ abaixo é instânciada e chamada na classe "Exemplo1"

```
package br.com.aulagui.view.jdialog;

import javax.swing.JDialog;

public class Exemplo2 extends JDialog {
    public static void main(String[] arts) { // Define a classe como principal
        Exemplo2 ex2 = new Exemplo2(); // Instância a classe referente ao diálogo Exemplo2
    }
}
```

JDialog - AbrirJDialogModal

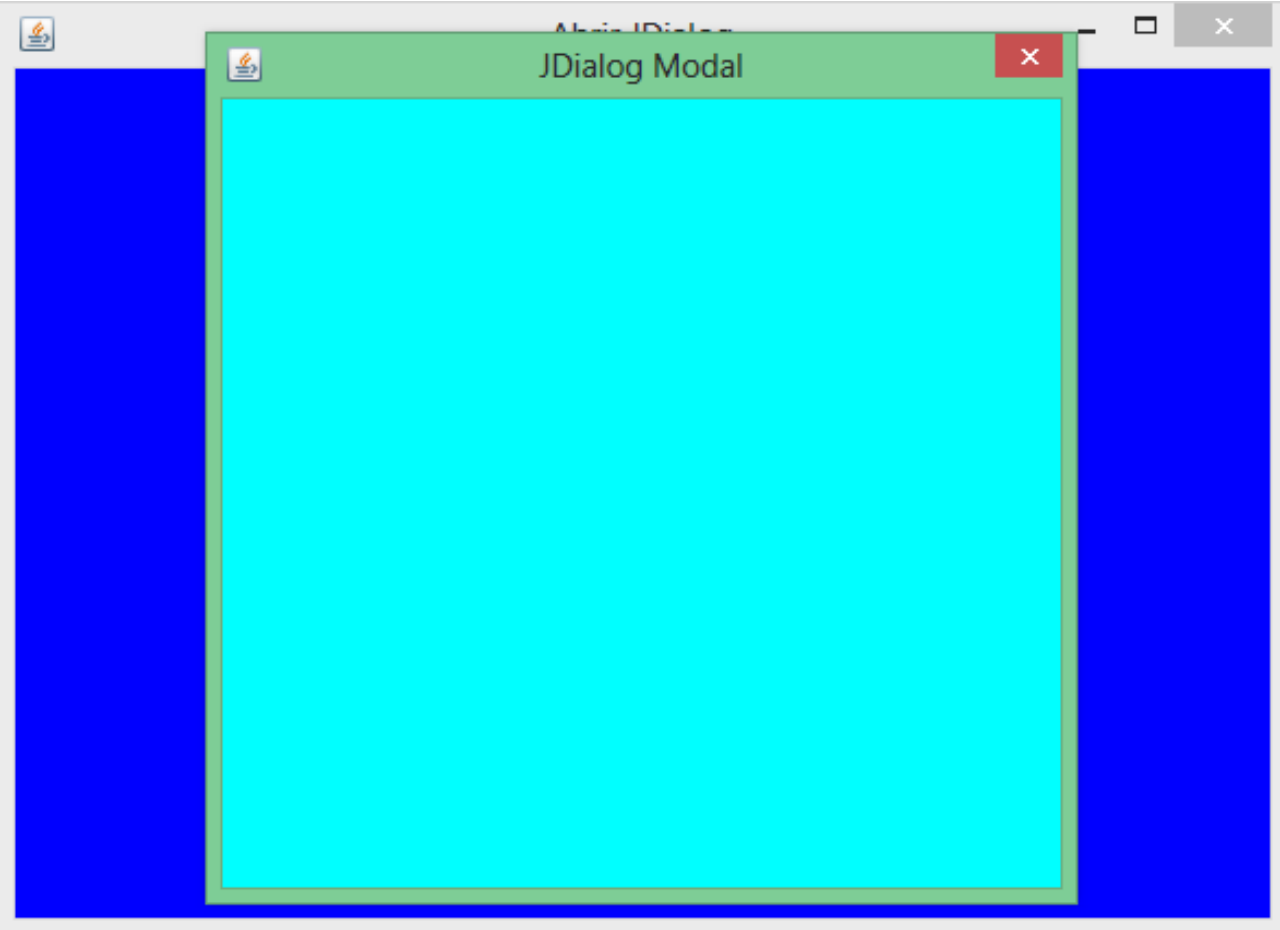
- A classe “Exemplo1“ é instânciada dentro da ação disparada pelo evento do botão (JButton), criado no Netbeans.

- Quando clicado o botão abre um diálogo modal…

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
      Exemplo1 ex1 = new Exemplo1();
      ex1.setSize(400,400);
      ex1.setTitle("JDialog Modal");
      ex1.setModal(true);
      ex1.getContentPane().setBackground(Color.cyan);
      ex1.setLocationRelativeTo(null);
      ex1.setVisible(true);
}
```

JDialog - AbrirJDialogModal

JDialog - AbrirJDialogModal



O diálogo em azul escuro só pode ser acessado quando o diálogo em azul claro é fechado, pois ambos são modais.

JDialog - Exercícios

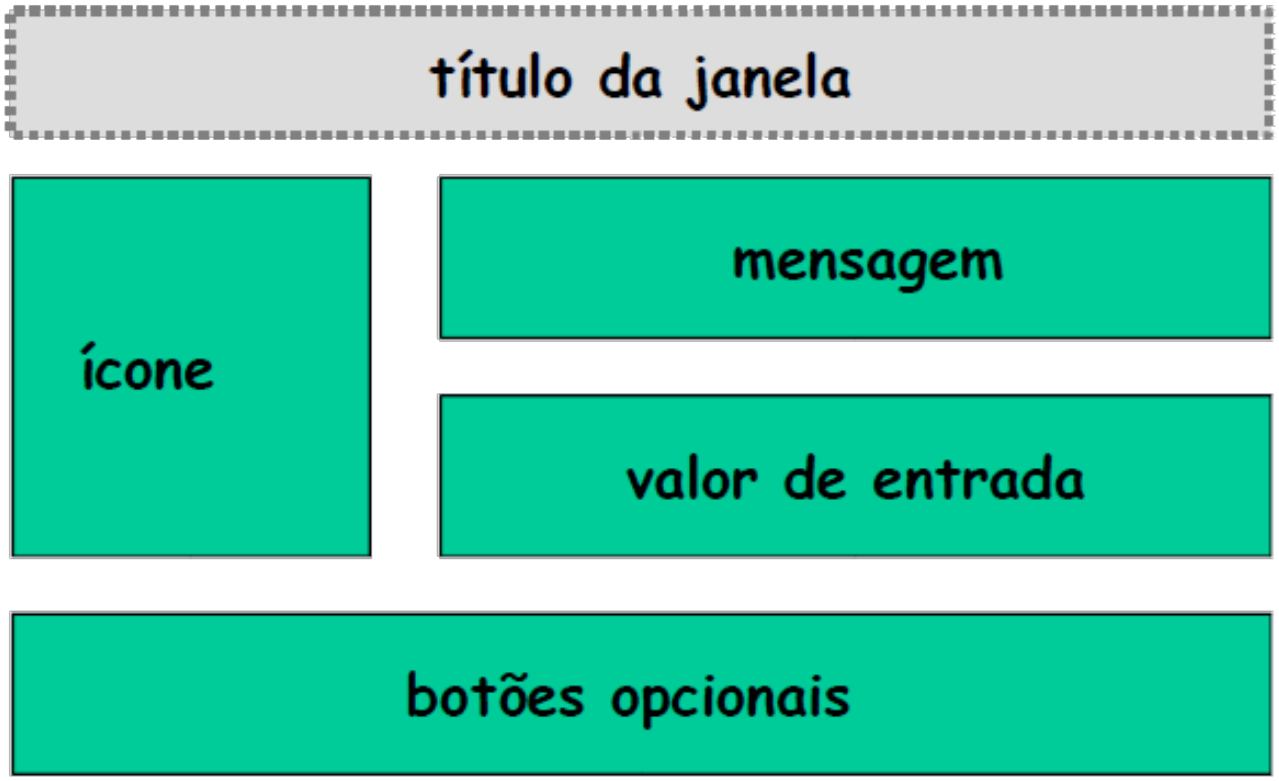
- Conforme os exemplos apresentados, implemente JDialog's no NetBeans, considerando pelo menos os métodos referentes ao:
  - título do diálogo;
  - tamanho do diálogo;
  - cor de fundo do diálogo;
  - se o diálogo é modal ou não-modal; e
  - se o diálogo está visível ou não para o usuário.

JOptionPane - Swing

- O Swing oferece um conjunto de diálogos (mensagens) simples pré-definidos para uso em interações breves com o usuário:
  - Mensagens de erro e de alerta;
  - Obtenção de uma confirmação; e
  - Entrada de um único campo de texto.
- Esses diálogos são modais.

Classe JOptionPane - Swing

- A aparência das caixas de diálogo é similar à figura:

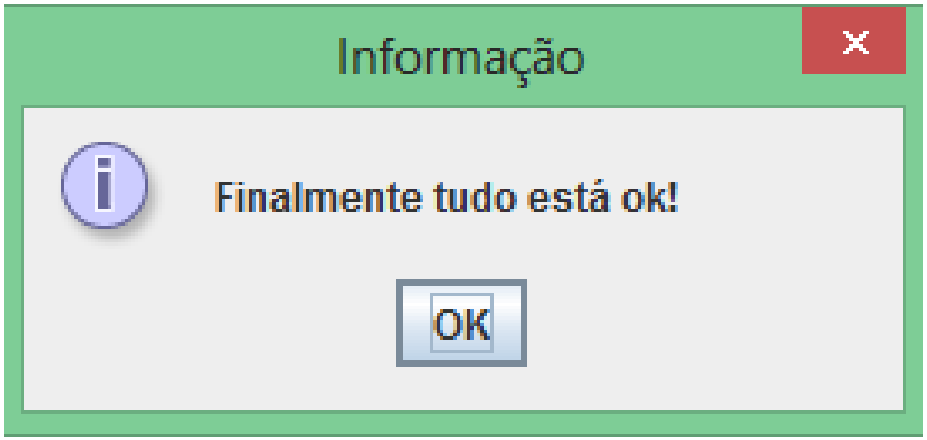


JOptionPane - MessageDialog

```
package br.com.aulagui.view.joptionpane;

import java.awt.Component;
import javax.swing.JOptionPane;

public class MessageDialogGUI {
    private static Component janelaMensagem;

    public static void main(String[] args) { // Define a classe como principal

        // Mensagem de informação
        JOptionPane.showMessageDialog(janelaMensagem,
                                      "Finalmente tudo está ok!",
                                      "Informação",
                                      JOptionPane.INFORMATION_MESSAGE);
    }
}
```

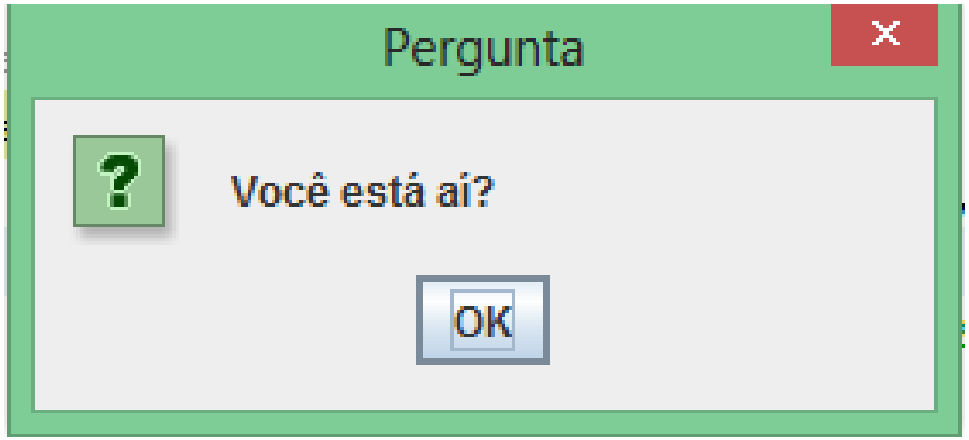
JOptionPane - MessageDialog

```
// Mensagem de informação
JOptionPane.showMessageDialog(janelaMensagem,
                              "Finalmente tudo está ok!",
                              "Informação",
                              JOptionPane.INFORMATION_MESSAGE);
```



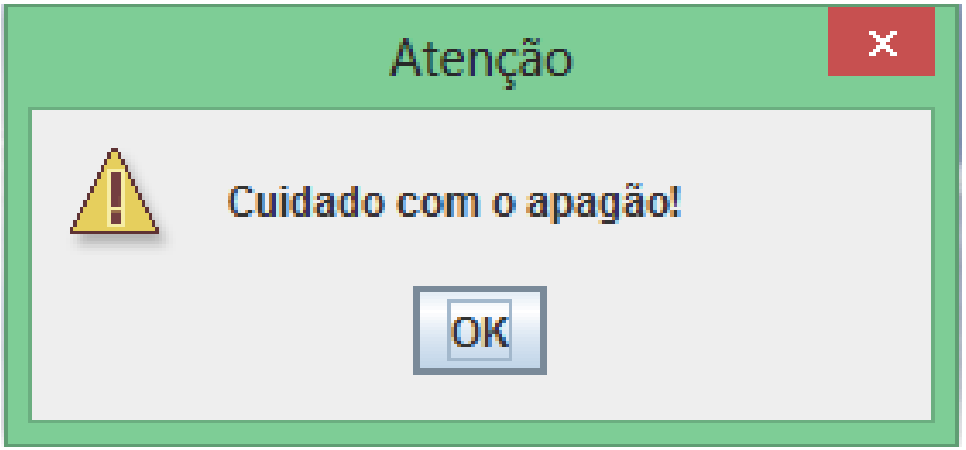
JOptionPane - MessageDialog

```
// Mensagem de pergunta
JOptionPane.showMessageDialog(janelaMensagem,
                              "Você está aí?",
                              "Pergunta",
                              JOptionPane.QUESTION_MESSAGE);
```



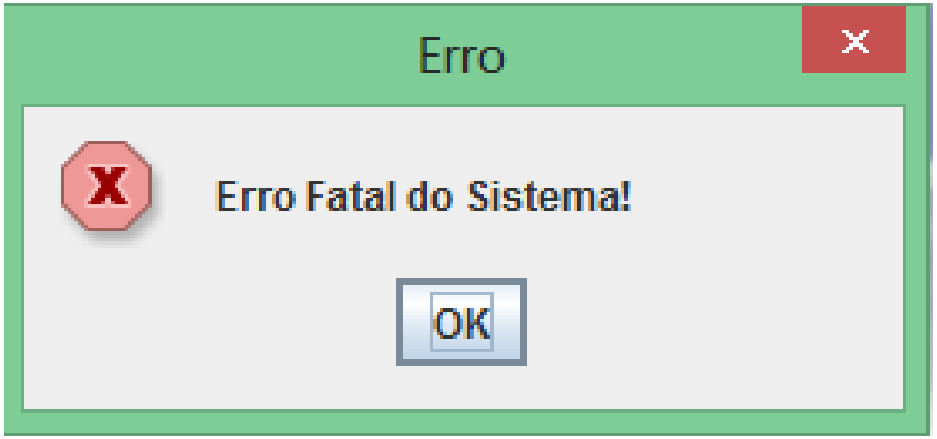
JOptionPane - MessageDialog

```
// Mensagem de aviso
JOptionPane.showMessageDialog(janelaMensagem,
                              "Cuidado com o apagão!",
                              "Atenção",
                              JOptionPane.WARNING_MESSAGE);
```



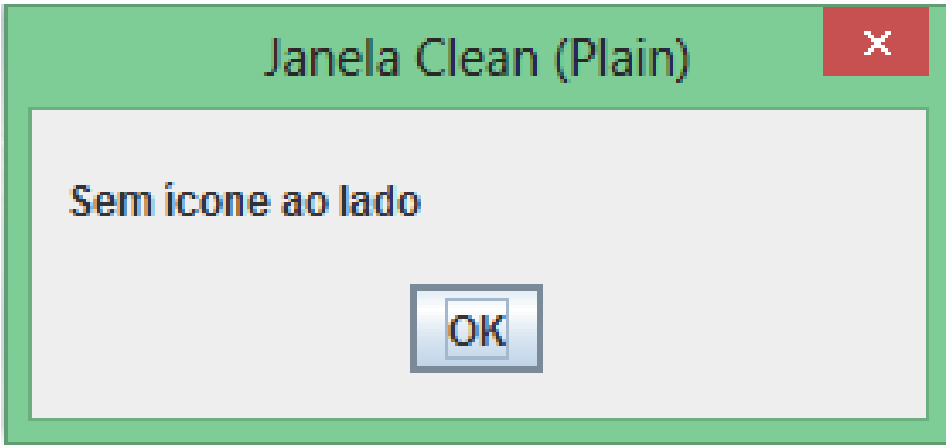
JOptionPane - MessageDialog

```
// Mensagem de erro
JOptionPane.showMessageDialog(janelaMensagem,
                              "Erro Fatal do Sistema!",
                              "Erro",
                              JOptionPane.ERROR_MESSAGE);
```



JOptionPane - MessageDialog

```
// Mensagem limpa sem ícone
JOptionPane.showMessageDialog(janelaMensagem,
                              "Sem ícone ao lado",
                              "Janela Clean (Plain)",
                              JOptionPane.PLAIN_MESSAGE);
```

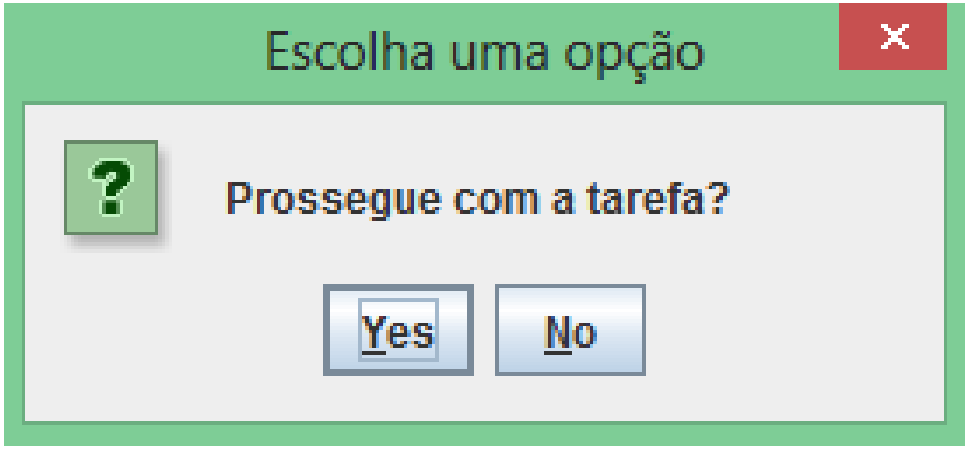


JOptionPane - ConfirmDialog

```
package br.com.aulagui.view.joptionpane;

import java.awt.Component;
import javax.swing.JOptionPane;

public class ConfirmDialogGUI {
    private static Component janelaConfirmacao;

    public static void main(String[] args) { // Define a classe como principal
        // Mensagens de confirmação (optionType)
        JOptionPane.showConfirmDialog(janelaConfirmacao,
                                      "Prossegue com a tarefa?",
                                      "Escolha uma opção",
                                      JOptionPane.YES_NO_OPTION);

        JOptionPane.showConfirmDialog(janelaConfirmacao,
                                      "Deseja realizar esta ação?",
                                      "Escolha uma opção",
                                      JOptionPane.YES_NO_CANCEL_OPTION);

        JOptionPane.showConfirmDialog(janelaConfirmacao,
                                      "Tem Certeza?",
                                      "Escolha uma opção",
                                      JOptionPane.OK_CANCEL_OPTION);
    }
}
```

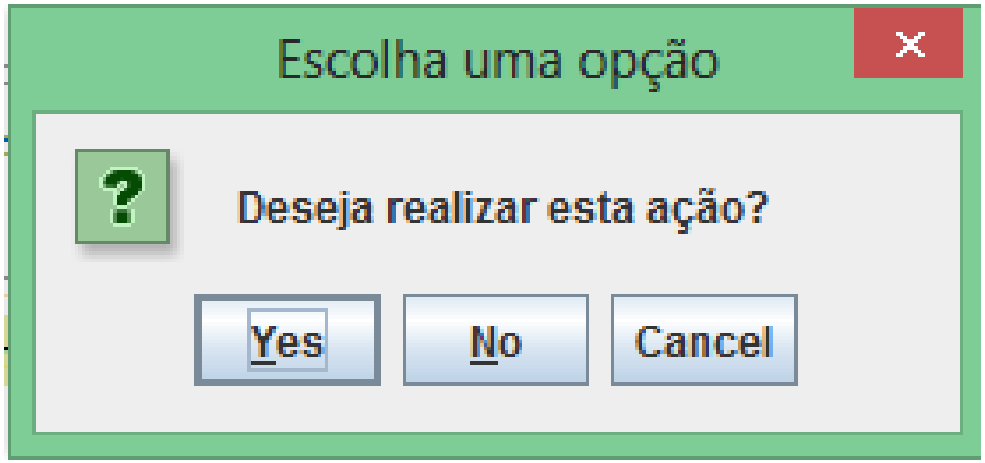
JOptionPane - ConfirmDialog

```
// Mensagens de confirmação (optionType)
JOptionPane.showConfirmDialog(janelaConfirmacao,
                              "Prossegue com a tarefa?",
                              "Escolha uma opção",
                              JOptionPane.YES_NO_OPTION);
```



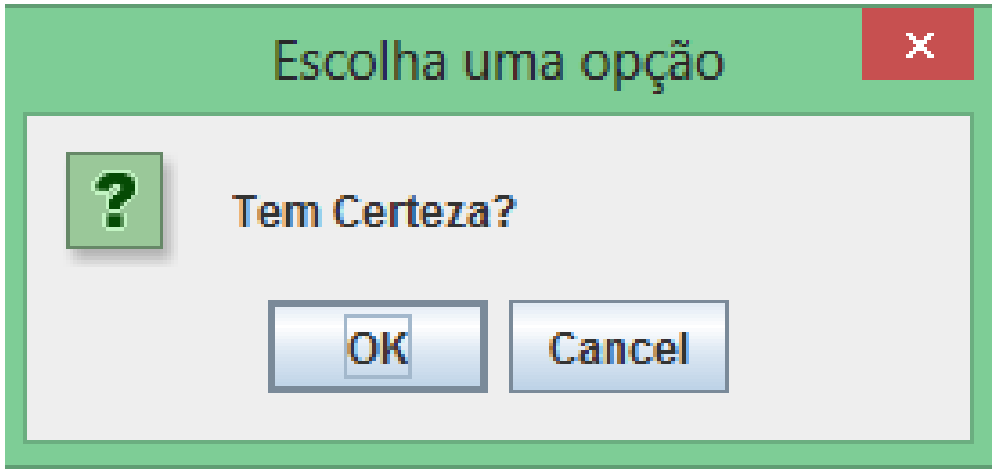
JOptionPane - ConfirmDialog

```
JOptionPane.showConfirmDialog(janelaConfirmacao,
                              "Deseja realizar esta ação?",
                              "Escolha uma opção",
                              JOptionPane.YES_NO_CANCEL_OPTION);
```



JOptionPane - ConfirmDialog

```
JOptionPane.showConfirmDialog(janelaConfirmacao,
                              "Tem Certeza?",
                              "Escolha uma opção",
                              JOptionPane.OK_CANCEL_OPTION);
```



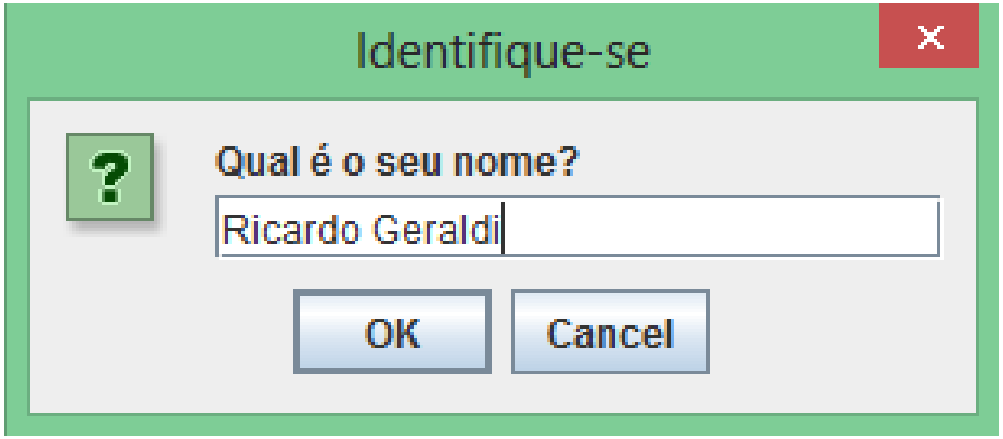
JOptionPane - InputDialog

```
package br.com.aulagui.view.joptionpane;

import java.awt.Component;
import javax.swing.JOptionPane;

public class InputDialogGUI {
    private static Component janelaEntrada;
    private static Component janelaResposta;

    public static void main(String[] args) { // Define a classe como principal
        // Janela de entrada
        String nome = JOptionPane.showInputDialog(janelaEntrada,
                                                  "Qual é o seu nome?",
                                                  "Identifique-se",
                                                  JOptionPane.QUESTION_MESSAGE);
        // Janela de resposta
        JOptionPane.showMessageDialog(janelaResposta,
                                      nome,
                                      "Resposta",
                                      JOptionPane.INFORMATION_MESSAGE);
    }
}
```

JOptionPane - InputDialog

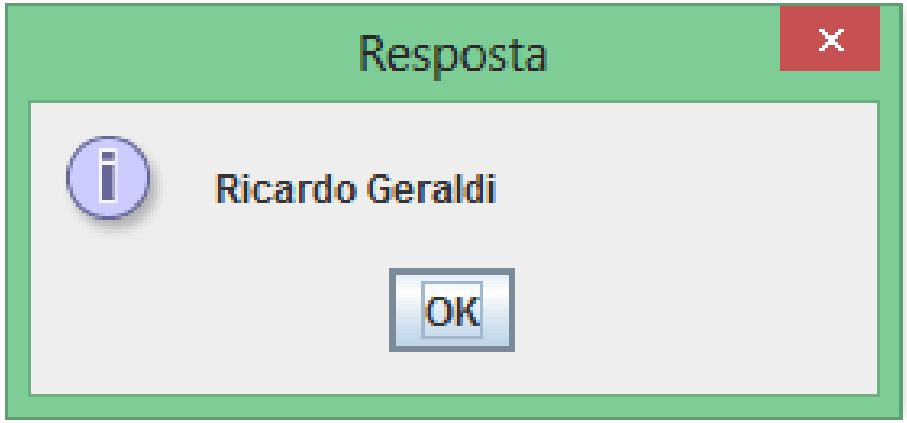
```
// Janela de entrada
String nome = JOptionPane.showInputDialog(janelaEntrada,
                                          "Qual é o seu nome?",
                                          "Identifique-se",
                                          JOptionPane.QUESTION_MESSAGE);
// Janela de resposta
JOptionPane.showMessageDialog(janelaResposta,
                              nome,
                              "Resposta",
                              JOptionPane.INFORMATION_MESSAGE);
```

JOptionPane - InputDialog



InputDialog

MessageDialog



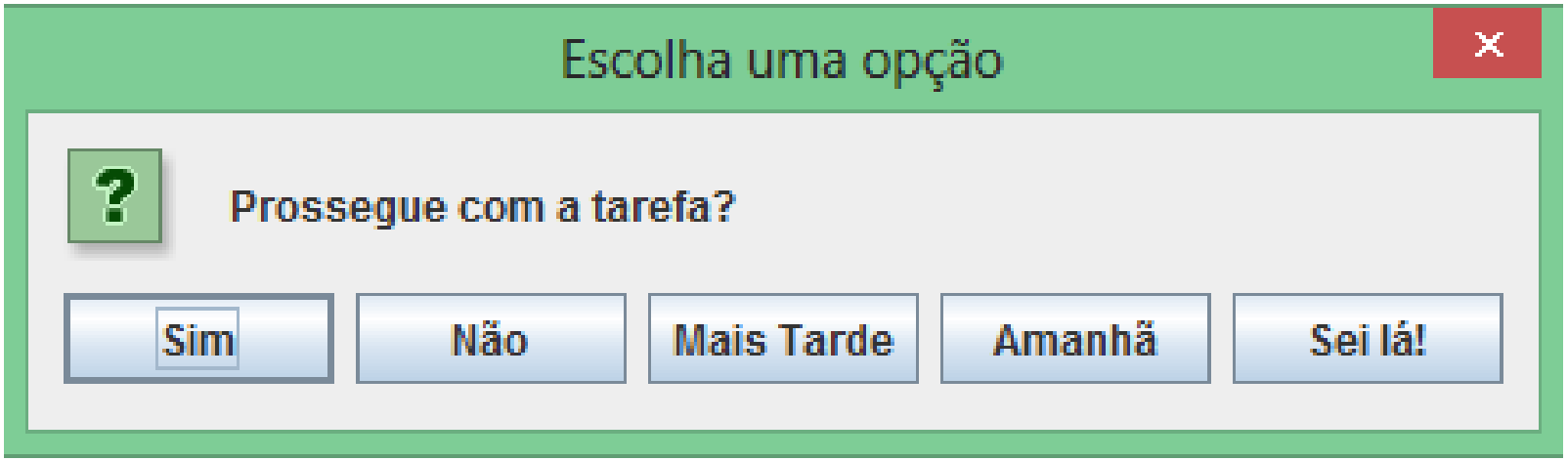
JOptionPane - OptionDialog

```
package br.com.aulagui.view.joptionpane;

import java.awt.Component;
import javax.swing.JOptionPane;

public class OptionDialogGUI {
    private static Component janelaOpcao;

    public static void main(String[] args) { // Define a classe como principal
        // Mensagem de informação
        Object[] opcoes =  {"Sim", "Não", "Mais Tarde", "Amanhã", "Sei lá!"};

        // Diálogo de confirmação com opções
        int resposta = JOptionPane.showOptionDialog(janelaOpcao,
                                    "Prossegue com a tarefa?",
                                    "Escolha uma opção",
                                    JOptionPane.DEFAULT_OPTION,
                                    JOptionPane.QUESTION_MESSAGE,
                                    null,
                                    opcoes,
                                    opcoes[0]);

    }
}
```

JOptionPane - OptionDialog

```
// Mensagem de informação
Object[] opcoes =  {"Sim", "Não", "Mais Tarde", "Amanhã", "Sei lá!"};

// Diálogo de confirmação com opções
int resposta = JOptionPane.showOptionDialog(janelaOpcao,
                                            "Prossegue com a tarefa?",
                                            "Escolha uma opção",
                                            JOptionPane.DEFAULT_OPTION,
                                            JOptionPane.QUESTION_MESSAGE,
                                            null,
                                            opcoes,
                                            opcoes[0]);
```

JOptionPane - OptionDialog



JOptionPane - Exercícios

- Conforme os exemplos apresentados, implemente JOptionPane's no NetBeans, considerando os tipos de mensagens dos diálogos:
  - diálogo de Mensagem;
  - diálogo de Mensagem de Confirmação;
  - diálogo de Mensagem de Entrada; e
  - diálogo de Mensagem com Opções.

MVC (*Model-View-Controller*)

- O padrão MVC (*Model-View-Controller*) – em português Modelo-Visão-Controlador – soluciona dificuldades de organização no desenvolvimento projetos na medida em que propõe desacoplar o acesso aos dados e as regras de negócio da maneira como o usuário pode visualizá-los.

- Padrão muito comum, conhecido e utilizado por desenvolvedores em todo o mundo.

MVC (*Model-View-Controller*)

- **Modelo (*Model*)** – responsável por manter o estado da aplicação. Ele é mais que uma classe para armazenar os dados, nele devem estar contidas todas as regras de negócio que se aplicam a esses dados. Também é responsável por comunicar-se com o banco de dados, se for necessário;

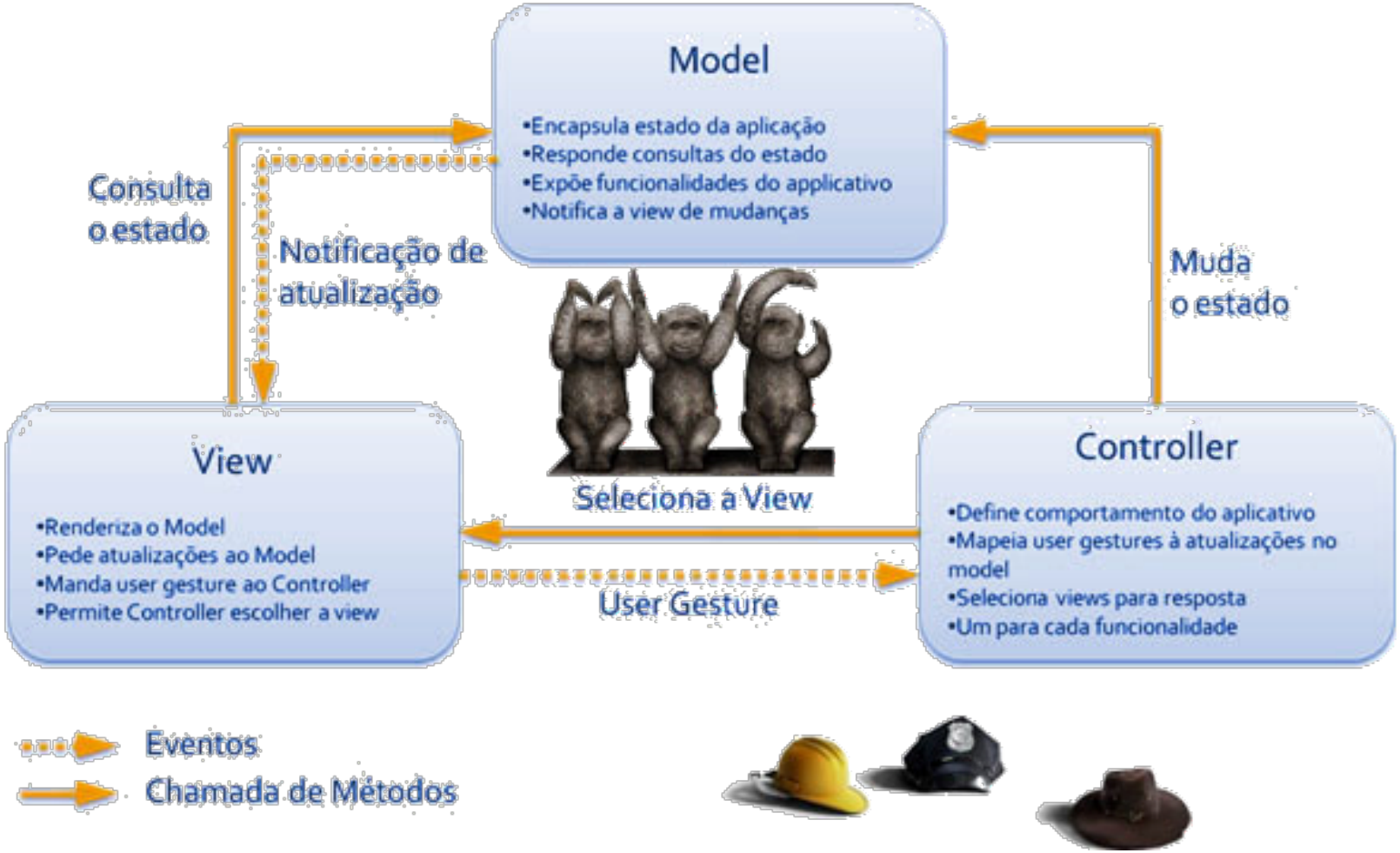
MVC (*Model-View-Controller*)

- **Visão (*View*)** – especifica exatamente como o modelo deve ser apresentado. É a interface do usuário. A visão é dinâmica, se adequando a qualquer modificação no modelo;

MVC (*Model-View-Controller*)

- Controlador (*Controller*) – o controlador traduz as interações do usuário com a visão, mapeando-as para tarefas que o modelo irá realizar. Em uma aplicação *stand-alone*, a interação do usuário poderia ser um clique de botão, por exemplo.

MVC (*Model-View-Controller*)



Referências

Araújo, Carlos. Criando Aplicações com MVC. Java Magazine 83.
http://www.devmedia.com.br/criando-aplicacoes-com-mvc-java-magazine-83/18021

Campione, Mary. The Java Tutorial: object-oriented programming for the internet. Addison Wesley Longman, California- USA, 1998.

Classes Essenciais Java:
http://java.sun.com/docs/books/tutorial/essential/TOC.html.

Cornell, Gary. Core Java; tradução Daniel Vieira; revisão técnica Rodrigo Rodrigues. São Paulo, Makron Books, 1997.

Deitel, H. M.;Deitel, P. J. Java, Como Programar. 3a. edição – Porto Alegre: Bookman, 2001.

Referências

Eckel, Bruce. Thinking in Java. Prentice Hall PTR, USA, 1998.

Java Tutorial:
http://java.sun.com/docs/books/tutorial/java/TOC.html#nutsandbolts.

JAVA Swing (JFC):
http://java.sun.com/docs/books/tutorial/uiswing/TOC.html#start.

JAVADOC: http://java.sun.com/j2se/1.4/docs/api/index.html.

JAVA WORLD e CORE JAVA:
http://www.javaworld.com/channel_content/jw-core-index.shtml.

Referências

PUC-Rio/Departamento de Informática - Grupo de Linguagens de Programação. Programação Java – Construção de Interfaces com Swing. http://www.inf.puc-rio.br/~java/progjava/protected/apostilas/Swing1.pdf

PUC-Rio/Departamento de Informática - Grupo de Linguagens de Programação. Programação Java – Construção de Interfaces com Swing II. http://www.inf.puc-rio.br/~java/progjava/protected/apostilas/Swing2.pdf

Rose India. Java Model View Controller (MVC) Design Pattern. 2010. http://www.roseindia.net/tutorial/java/jdbc/javamvcdesignpattern.html.

UCB. Interfaces Gráfica em Java. Módulo I. Introdução a Linguagem Java

WIKIPÉDIA. MVC. http://pt.wikipedia.org/wiki/MVC