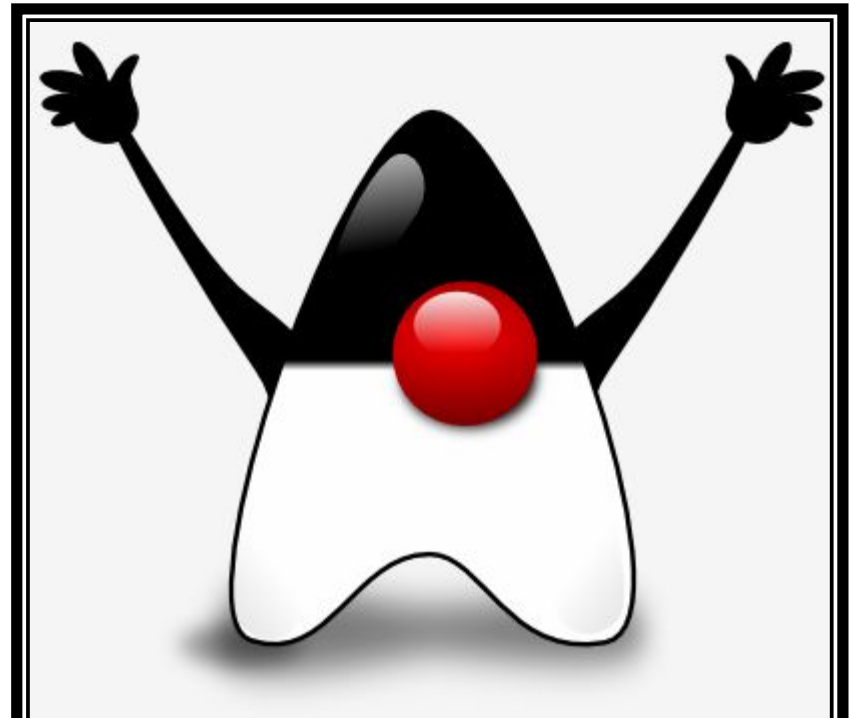


# PROGRAMAÇÃO ORIENTADA A OBJETO

Programando Interfaces Gráficas

# Roteiro

- Introdução
- Containers e Componentes
- Componentes Swing
- Exercícios



# Introdução

- Java permite o desenvolvimento para diversos tipos de interface.
  - Para isto precisamos trabalhar com APIs (bibliotecas) específicas.
- O objetivo da aula de hoje é discutir e apresentar as ferramentas e possibilidades de desenvolvimento de interfaces gráficas.

# Introdução

- Há dois tipos de aplicações gráficas em Java
  - Componentes iniciados via browser (*applets*)
  - Aplicações *standalone* iniciadas via sistema operacional
- *Ambas capturam eventos do sistema e desenhavam-se sobre um contexto gráfico fornecido pelo sistema*

# Introdução

- IDEs:
  - Netbeans
    - Plugin para desenvolvimento gráfico já vem instalado



- Eclipse
  - É necessário baixar plugin, por exemplo :
    - Visual Editor Project



# Introdução

- IDEs:
  - Netbeans
    - Plugin para desenvolvimento gráfico já vem instalado



- Eclipse
  - É necessário baixar plugin, por exemplo :
    - Visual Editor Project



# Introdução

- Graphical User Interface (GUI)
  - Existe uma infinidade de funcionalidades disponíveis nas bibliotecas de classes Java, destinadas a prover a comunicação homem-máquina gráfica.
  - Na próxima aula teremos outro tipo de Interface, mas são coisas completamente distintas entre si.

# Introdução

- Graphical User Interface (GUI)
  - Os elementos básicos necessários para criar um GUI residem em dois pacotes:
  - `java.awt`: Abstract Windowing Toolkit (classes básicas);
  - `javax.swing`: Swing Components - fornece melhores alternativas aos componentes definidos na classe `java.awt`.
- Exemplo: estudaremos a classe `JButton` do Swing no lugar da classe `Button`, do `java.awt`.



# Containers e componentes

- Uma interface gráfica em Java é baseada em dois elementos:
  - containers: servem para agrupar e exibir outros componentes
  - componentes: botões, labels, scrollbars, etc.
- Dessa forma, todo programa que ofereça uma interface vai possuir pelo menos um container, que pode ser:
  - JFrame: janela principal do programa
  - JDialog: janela para diálogos
  - JApplet: janela para Applets

# Containers e componentes

- Para construirmos uma interface gráfica em JAVA, adicionamos componentes (Botões, Menus, Textos, Tabelas, Listas, etc.) sobre a área da janela.
- Por essa razão a área da janela é um container, ou seja, um elemento capaz de armazenar uma lista de componentes.

# Containers essenciais

- Frame (AWT) e JFrame (Swing)
  - Servem de base para qualquer aplicação gráfica
- Panel e JPanel
  - Container de propósito geral
  - Serve para agrupar outros componentes e permitir layout em camadas
- Applet e JApplet
  - Tipo de Panel (JPanel) que serve de base para aplicações que rodam dentro de browsers
  - Pode ser inserido dentro de uma página HTML e ocupar o contexto gráfico do browser

# Introdução

- Graphical User Interface (GUI)
  - As classes Swing são parte de um conjunto mais genérico de capacidades gráficas, chamado de Java Foundation Classes (JFC). Suporta:
    - definição de botões, menus, etc.
    - desenho 2D (`java.awt.geom`)
    - funcionalidades drag-and-drop (`java.awt.dnd`)
    - API com acessibilidade a usuários (`javax.accessibility`)
  - Swing é mais flexível que `java.awt` porque é implementada toda em Java, enquanto que `java.awt` é implementada em código nativo.

# Introdução

- Package Swing
  - Criado em 1997
  - Extensão da AWT - Abstract Window Toolkit
  - Classes implementadas inteiramente em Java
  - Mesma estrutura que os componentes AWT
  - Componentes que fornecem melhores alternativas para a implementação de interfaces gráficas
    - JButton no lugar de Button, JFrame no lugar de Frame, etc.
  - As classes de Swing fazem parte de um conjunto mais genérico de classes com capacidades gráficas: JFC (Java Foundation Classes)

# Introdução



**Por onde eu devo começar então?**

# Praticando

- Agora vamos começar a construir nossa primeira aplicação com GUI.
  - Modifique as dimensões da Janela Principal
  - Imprima algum texto na janela

# Prática I



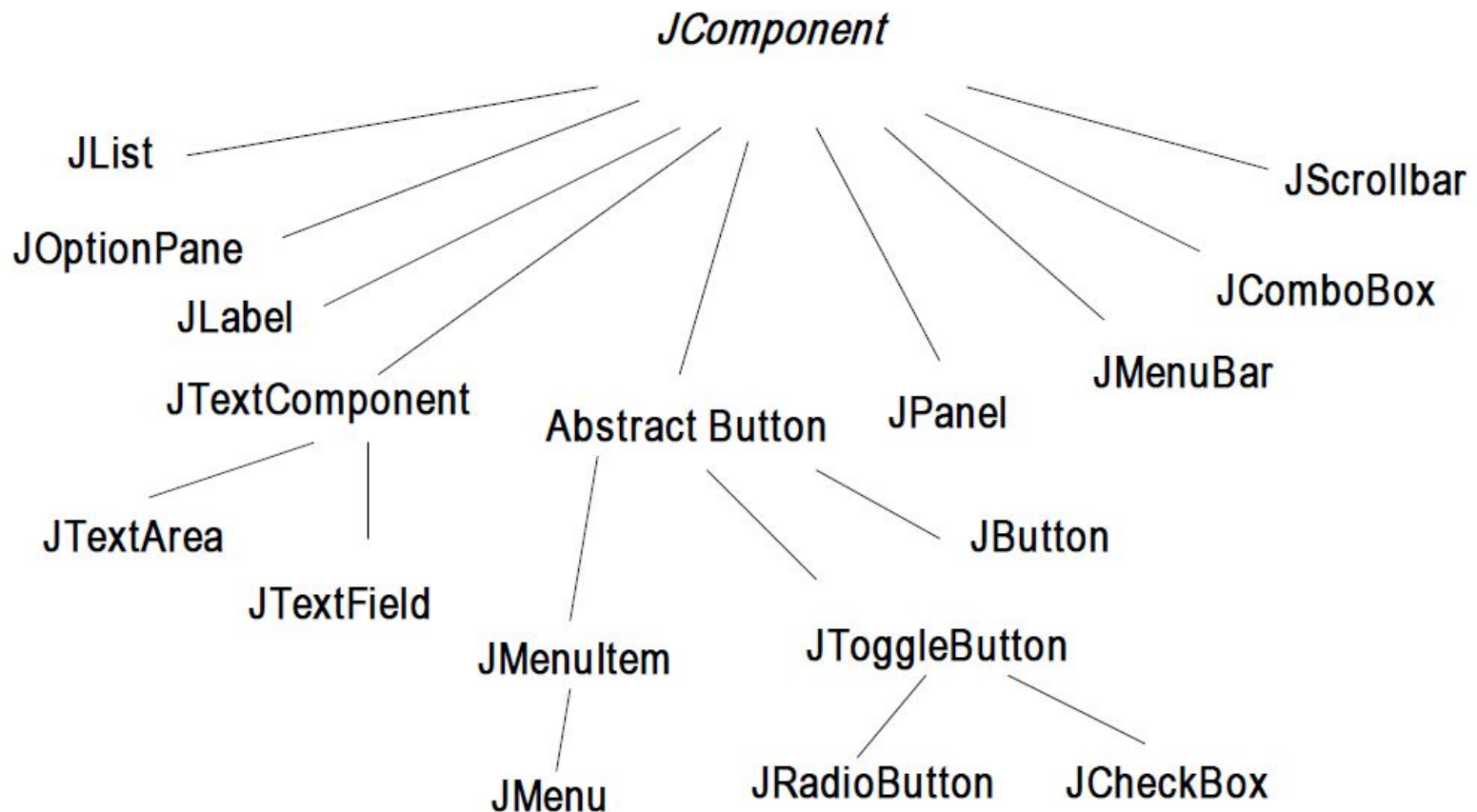
- Faça um Hello World com o componente área de texto.
- Três minutos



# Componentes Swing

- Componentes swing são divididos em:
  - Visuais:
    - botões, menus, barras de ferramentas, etc.
  - Não-visuais, de auxílio aos outros:
    - Painel, painel tabulado, etc.

# Componentes Swing

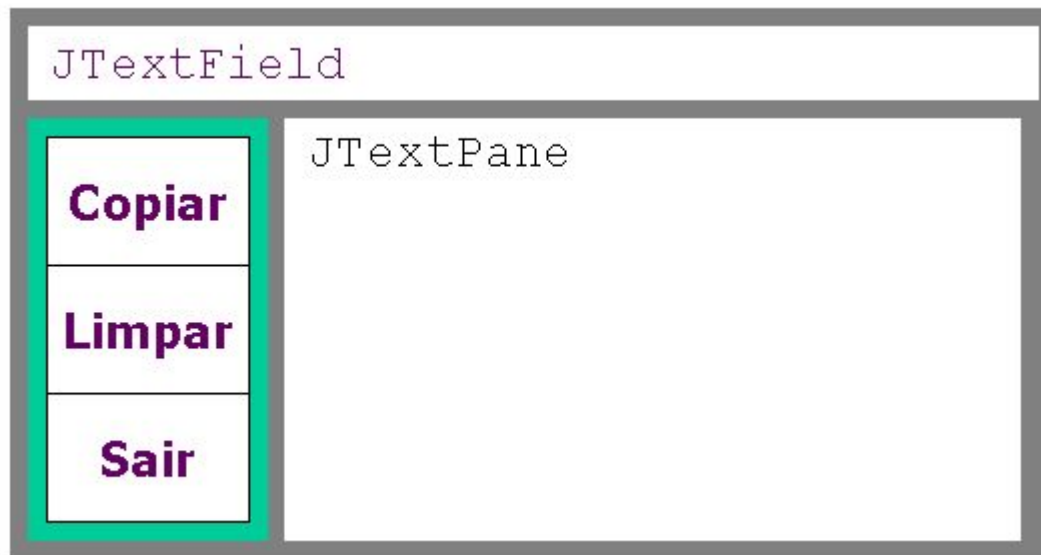


# Componentes Swing

- Alguns atributos de componentes:
  - posição (x,y): posição do objeto em relação ao seu container;
  - nome do componente (`myWindow.setName("Teste");`);
  - tamanho: altura e largura;
  - cor do objeto e cor de fundo;
  - fonte
  - aparência do cursor;
  - objeto habilitado ou não (`isEnabled()`, `myWindow.setEnabled()`);
  - objeto visível ou não (`isVisible()`, `myWindow.setVisible()`);
  - objeto válido ou não.
- Todos atributos são private.

# Prática II

1. Construa uma aplicação gráfica que contenha três botões (JButton), um JTextField e um JTextPane distribuídos da seguinte forma:



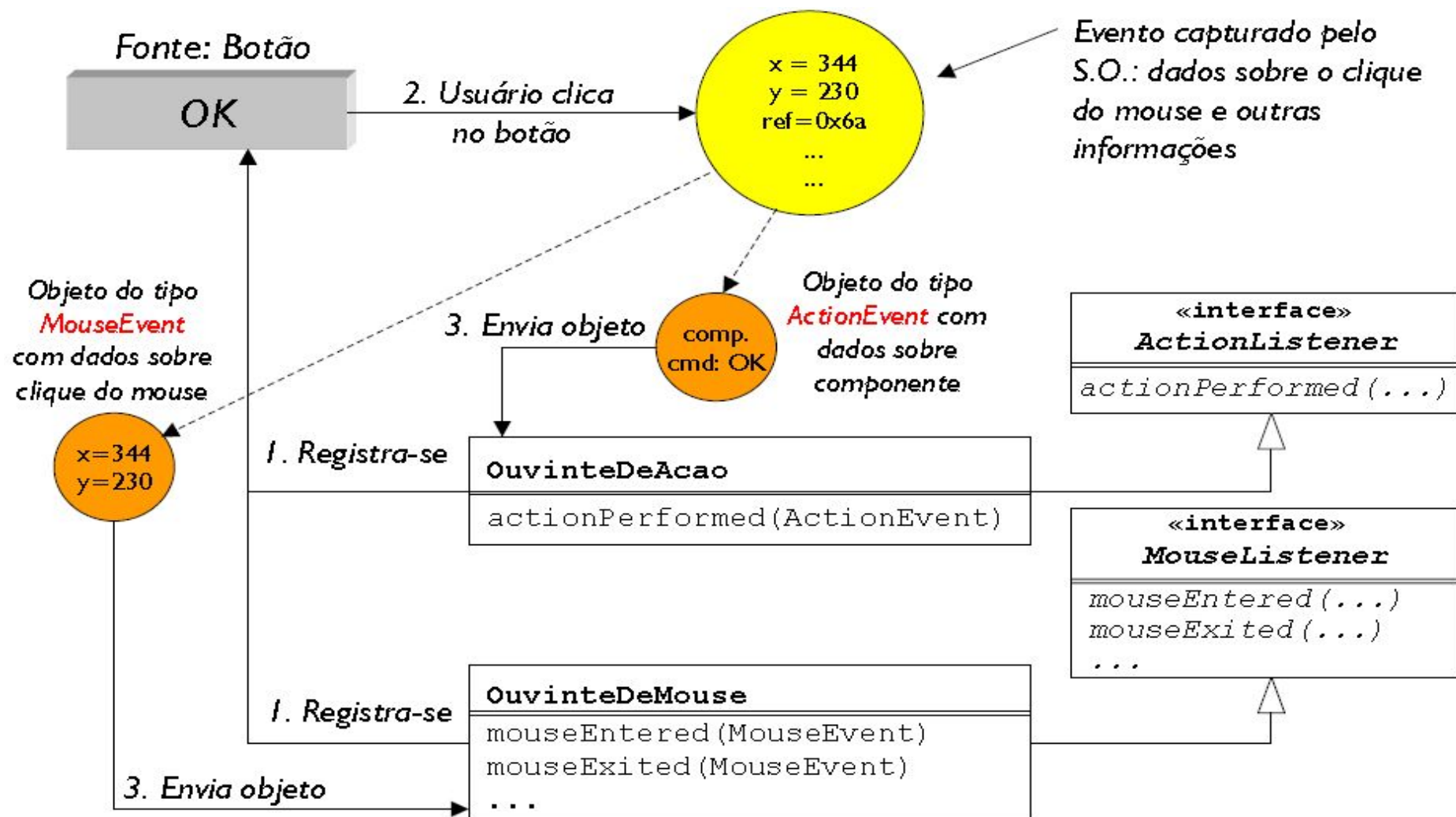
# Eventos

- Eventos em Java são objetos - `java.util.EventObject`
- Todo evento tem um objeto que é sua fonte
  - ***Object fonte = evento.getSource();***
- Métodos de ouvintes (*listeners*) que desejam tratar eventos, recebem eventos como argumento

```
public void eventoOcorreu(EventObject evento) {  
    Object fonte = evento.getSource();  
    System.out.println(""" + evento + " em " + fonte);  
}
```

- Ouvintes precisam ser registrados nas fontes
  - Quando ocorre um evento, um método de todos os ouvintes registrados é chamado e evento é passado como argumento

# Eventos



# Eventos

- Descendentes de `java.awt.event.AWTEvent`
  - Divididos em categorias (`java.awt.event`)
    - `ActionEvent` (fonte: componentes de ação)
    - `MouseEvent` (fonte: componentes afetados pelo mouse)
    - `ItemEvent` (fonte: checkboxes e similares)
    - `AdjustmentEvent` (fonte: scrollbars)
    - `TextEvent` (fonte: componentes de texto)
    - `WindowEvent` (fonte: janelas)
    - `FocusEvent` (fonte: componentes em geral)
    - `KeyEvent` (fonte: componentes afetados pelo teclado) ...

# Prática III

- Agora crie um formulário para cadastrar Pessoas em uma lista de Pessoas (lista do tipo ArrayList).
  - A pessoa deve ter os seguintes atributos:
    - nome, endereço e telefone.



# Prática IV (Sua vez!)

- Faça um formulário de cadastro de Alunos em um Sistema Acadêmico.
  - Os alunos devem ter os seguintes atributos:
    - Nome
    - Turma
    - Endereço
  - Por fim, salve em uma lista no momento que clicar em salvar.
  - Utilizem o projeto anterior para interface

# Uma janela simples...

- A aplicação do exercício anterior possui os seguintes elementos:
  - JFrame: armazena os demais componentes
  - JPanel: painel, serve para facilitar o posicionamento do botão e do label
  - JButton
  - JLabel

# Uma janela simples...

- JFrames são top-level containers: sempre estão presentes
- JPanels são intermediate containers: podem estar ou não presentes (mas geralmente estão)
- JButton e JLabel são componentes atômicos: não podem ser usados para conter e normalmente respondem ao usuário

# Pratica IV

- Faça uma interface gráfica igual a esta:

The image shows a screenshot of a software window titled "Form Preview [ContactEditor]". The window has a blue title bar with standard Windows window controls (minimize, maximize, close). The main content area is divided into two sections: "Name" and "E-mail".

**Name Section:**

- First Name: [Text Input Field]
- Last Name: [Text Input Field]
- Title: [Text Input Field]
- Nickname: [Text Input Field]
- Display Format: [Dropdown Menu] (Currently showing "Item 1")

**E-mail Section:**

- E-mail Address: [Text Input Field]
- [List Box containing: Item 1, Item 2, Item 3, Item 4, Item 5]
- [Buttons: Add, Edit, Remove, Advanced]
- Mail Format: [Radio Buttons] (HTML, Plain Text, Custom)

At the bottom right of the window are "OK" and "Cancel" buttons.



# DÚVIDAS?

alanamm.prof@gmail.com