ELE2 – LPG3

Eletiva 2
Linguagem de Programação 3 (JAVA)

Interfaces gráficas

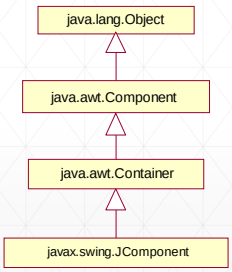
GUI (*Graphical User Interface*)

- A interface gráfica com o usuário (*GUI - Graphical User Interface*) possibilita, de forma intuitiva, que o usuário tenha um nível básico de familiaridade com um programa, sem que jamais o tenha utilizado. Dessa forma, é reduzido o tempo de aprendizado do programa pelo usuário.
- As GUIs são construídas a partir de *componentes GUI*. O componente GUI é um objeto com o qual o usuário interage através de interfaces (mouse, teclado, monitor etc).

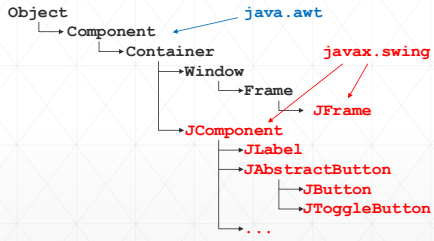
Bibliotecas: AWT x Swing

- Os componentes AWT (*Abstract Windowing Toolkit,* do pacote **java.awt**) foram a primeira versão de componentes GUI em Java e estão diretamente associados com os recursos gráficos da plataforma local. Assim, os componentes são exibidos com uma aparência diferente em cada plataforma (Windows, Apple, Solaris, etc);
- A versão 1.2 da linguagem Java (Java 2) trouxe os componentes Swing (pacote **javax.swing**), desenvolvidos totalmente em Java, que possibilitam a especificação de uma aparência uniforme independentemente da plataforma;
- Alguns componentes Swing utilizam o pacote AWT como superclasses de suas classes.

Hierarquia comum entre AWT e Swing



Classes do pacote javax.swing



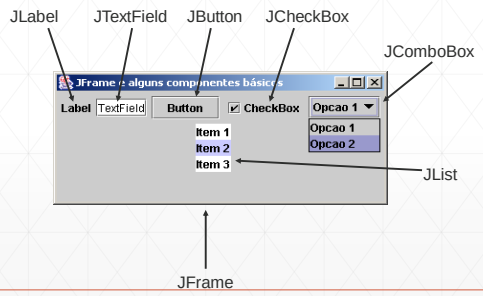
A classe JComponent

- A classe **JComponent** é a superclasse de todos os componentes Swing;
- Os objetos **JButton**, **JCheckbox**, e **JTextField** são todos exemplos de objetos das subclasses de **JComponent;**
- A classe **JComponent** é uma subclasse direta da classe **java.awt.Container** que, por sua vez, é uma subclasse direta de **java.awt.Component**.

Alguns Componentes GUI Básicos

- **JFrame:** é um *conteiner* (formulário) para outros componentes;
- **JLabel**: elemento para exibição de texto não-editável ou imagens;
- **JTextField**: elemento para receber dados do usuário via teclado;
- **JButton**: componente que aciona um evento ao clique do usuário;
- **JComboBox**: lista de itens com valores pré-definidos para seleção;
- **JCheckBox**: possui dois estados: selecionado ou não selecionado;
- **JRadioButton**: botão de escolha que pode ser agrupado com outros, permitindo que apenas um elemento seja escolhido;
- **JList**: área em que uma lista é exibida, possibilitando a seleção;
- **JPanel**: Contêiner em que os componentes podem ser colocados.

Alguns Componentes GUI Básicos

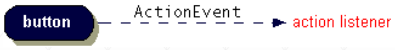


Tratamento de Eventos

- As GUIs são baseados em eventos gerados pela interação do usuário. Por exemplo, mover o mouse, clicar no mouse, digitar um campo de texto, fechar uma janela etc;
- Tanto os componentes AWT como os componentes Swing utilizam os tipos de eventos do pacote **java.awt.event** (embora o Swing também tenha seus próprios eventos no pacote **javax.swing.event**).

Tratamento de Eventos

- O mecanismo de tratamento de eventos possui três partes: a origem do evento, o objeto evento e o "ouvinte" (*listener*) do evento;



- O programador precisa:
  - Registrar um ouvinte de evento no componente;
  - implementar um método de tratamento do evento.

Tratamento de Eventos

- Algumas interfaces mais utilizadas são:
  - ActionListener;
  - KeyListener;
  - MouseListener;
  - FocusListener;
  - WindowListener.
- Cada interface *listener* de eventos especifica um ou mais métodos de tratamento de evento, que devem ser definidos na classe que implementa a interface relacionada.

Manipulando eventos – exemplo

- Para o ActionEvent (evento de <Enter> num botão), deve-se:
  - criar um objeto que implementa a interface **ActionListener**;
  - registrar esse objeto como um "ouvinte" (*listener*) desse botão usando o método **addActionListener**;
  - quando o botão for pressionado, ele dispara um evento, o que resulta na execução do método **actionPerformed** do objeto ouvinte;
  - o argumento desse método é um objeto **ActionEvent**, que informa acerca do evento e de sua fonte.

Manipulando eventos – exemplo

- De uma forma geral:
  - uma classe X que estiver interessada em processar eventos de ação deve implementar a interface **ActionListener**;
  - um objeto de X é registrado com um ***listener***, usando o método **addActionListener** desse componente;
  - quando a ação ocorre, o método **actionPerformed** é executado.

13

Manipulando eventos – exemplo

- Exemplo de código-fonte:

```
button.addActionListener(new ActionListener(){
  public void actionPerformed(ActionEvent e) {
    numeroDeCliques++;
    label.setText(numeroDeCliques);
  }
});
```

14