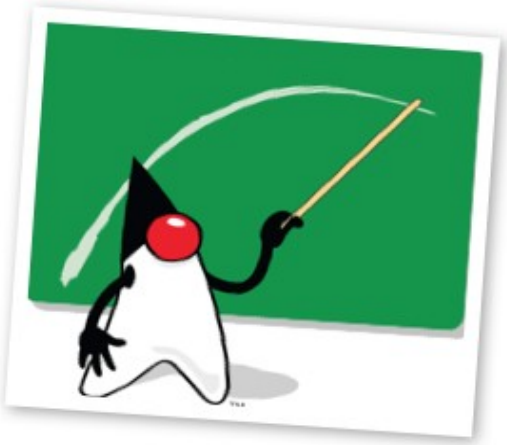
**Hortolândia, 16 de outubro de 2013.**

# Aulas 45 e 46 – Classes Gráficas em Java – Parte I

## 1. Introdução



A presente aula apresenta uma primeira introdução ao uso de classes gráficas em Java.

Um dos primeiros pacotes que surgiram para dar suporte às aplicações gráficas em Java foi o **javax.swing**.

Este pacote, popularmente conhecido como **Swing**, contém componentes de Interface Gráfica com o Usuário (Graphical User Interface - GUI).

Anteriormente ao swing, havia o pacote **java.awt**, cujos componentes estão diretamente associados com as capacidades de GUI da plataforma local. Um programa em Java que usasse apenas awt, poderia ter a sua aparência modificada conforme a plataforma em que estivesse executando (DEITEL, 2001).

Em nossa disciplina serão apresentadas inicialmente classes de uso geral, do pacote **Swing**. Tais classes podem ser utilizadas, bastando no início da classe que as for utilizar incluir o pacote correspondente (por exemplo, com a instrução `import javax.swing.*`).

Nesta aula e na próxima serão apresentados alguns dos principais componentes gráficos do Swing.

Em aulas futuras, será mostrado como utilizar a **IDE NetBeans** a fim de auxiliar a criação de interfaces gráficas. O NetBeans utiliza o mesmo pacote Swing, porém oferece a comodidade de arrastar e soltar componentes a fim de criar cada interface gráfica.

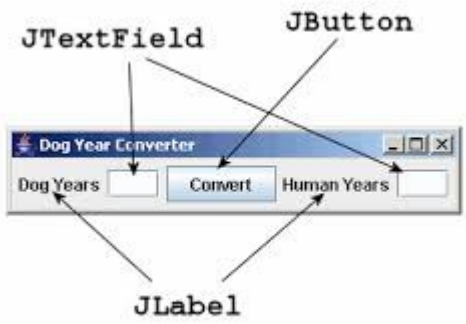
## 2. Componentes Gráficos do Swing

### 2.1. JLabel

O **JLabel** é um rótulo para outros componentes gráficos do pacote Swing. Um rótulo exibe uma única linha de texto, em modo somente de leitura.

O construtor de JLabel pode ser chamado com ou sem um argumento de texto, e também com um argumento de texto mais um ícone e uma constante de alinhamento.

Ele se aproveita das relações de **classe**, nas quais os **objetos** de uma certa classe - tal como uma classe de veículos - têm as mesmas características.




A Figura 2 ilustra os componentes **JLabel**, **JTextField** e **JButton**.

**Exemplos de uso de JLabel**:

```
    JLabel label1 = new JLabel(); // construtor sem argumentos.

    JLabel label2 = new JLabel("Rótulo com Texto"); // construtor com um argumento de texto.

    Icon bug = new ImageIcon("bug.gif");

    // criação de um ícone

    JLabel label3 = new JLabel("Rótulo com Texto e Ícone", bug, SwingConstants.LEFT);
```

O **JLabel**, assim como os demais componentes gráficos do **Swing**, deve ser adicionado a um container para ser exibido na tela.

Um programa para teste do JLabel pode está representado na **Listagem 1** a seguir.

```
import java.awt.Container;

import java.awt.Dimension;

import java.awt.FlowLayout;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

import javax.swing.*;

public class TesteJLabel extends JFrame{

    private JLabel label1, label2, label3;

    public TesteJLabel(){

    super("Teste de Uso do JLabel");

    Container c = getContentPane();

    c.setLayout(new FlowLayout());

    label1 = new JLabel("Rótulo do JLabel");

    ImageIcon icone1 = new javax.swing.ImageIcon(getClass().getResource("./linux.gif"));

    label2 = new JLabel("Linux", icone1, SwingConstants.LEFT);

    // c.add(label1);

    c.add(label2);

    this.setPreferredSize(new Dimension(200,250));

    this.pack();

    this.setVisible(true);

    }

    public static void main(String[] args){
```

```
        TesteJLabel tjl = new TesteJLabel();

        tjl.addWindowListener(new WindowAdapter(){

        @Override

        public void windowClosing(WindowEvent e){System.exit(0);}});

        }

}
```

**Listagem 1**: Teste de uso de um componente **JLabel**.

### 2.2. JTextField

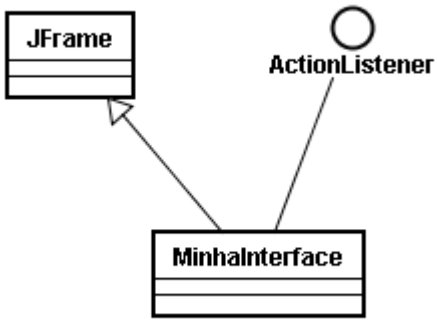
Um objeto **JTextField** é utilizado quando se deseja obter uma linha de informações do usuário via teclado ou exibir informações na tela.

Os **JTextFields** podem ser editáveis ou não editáveis. Um JTextField editável tem um fundo branco por padrão. Já um JTextField não editável tem fundo cinza.

### 2.3. JButton

Um objeto **JButton** corresponde a um botão na tela, que quando pressionado pelo usuário, faz com que alguma ação por parte do programa seja realizada.

Os botões, ao serem acionados, geram eventos. A ação de clicar no botão precisa ser capturada por um "ouvidor de eventos". Os "ouvidores de eventos" são objetos Java que implementam a interface "**ActionListener**".




A classe que implementar a interface "ActionListener", deve possuir o método "**actionPerformed**()", o qual será acionado quando o botão for pressionado.

Além disso, após criado o botão, deve-se adicionar ao mesmo o **ActionListener** criado para manipular os eventos desse botão.

#### 2.3.1. Exemplo de uso de JButton

```
JButton botao = new JButton("Enviar");

botao.addActionListener(this);
```

No exemplo acima, as linhas de código devem estar dentro de uma classe que implementa a interface **ActionListener**.

O método **actionPerformed**( ) é um dos vários métodos que processam informações entre o usuário e os componentes GUI.

A primeira linha do método

```
public void actionPerformed(ActionEvent e)
```

indica que **actionPerformed**( ) é um método público que não retorna nada (void) ao completar a sua tarefa.

Este método recebe um argumento - um ActionEvent - quando é chamado automaticamente em resposta a uma ação realizada em um componente GUI.

O argumento ActionEvent contém informações sobre a ação que ocorreu, tais como o nome da ação e o horário de ocorrência.

A classe **FrameComCampoTexto** ilustra um exemplo de utilização dos componentes JTexField e JButton do pacote Swing. Ela está representada na Listagem 2 a seguir.

```
import java.awt.BorderLayout;
import java.awt.Container;
import java.awt.Dimension;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JTextField;
public class FrameComCampoTexto extends JFrame {
      private int largura = 500;
      private int altura = 300;
      private JLabel titulo, rotuloNome;
      private JTextField campoNome;
      private JButton botao;
      private BotaoHandler handler;
      public FrameComCampoTexto() {
            super("Frame com Campo de Texto");
            setPreferredSize(new Dimension(largura, altura));
            Container c = getContentPane();
            c.setLayout(new BorderLayout());
            titulo = new JLabel("Dados Cadastrais");
            rotuloNome = new JLabel("Nome: ");
            campoNome = new JTextField(20);
            botao = new JButton("Enviar");
            handler = new BotaoHandler();
            botao.addActionListener(handler);
            c.add(titulo,BorderLayout.NORTH);
            c.add(rotuloNome,BorderLayout.WEST);
            c.add(campoNome,BorderLayout.CENTER);
```

```
            c.add(botao,BorderLayout.SOUTH);

      }

      private class BotaoHandler implements ActionListener{

      @Override

      public void actionPerformed(ActionEvent e){

      System.out.println("Evento: " + e.getActionCommand());

System.out.println("Valor do Campo nome: " +
campoNome.getText());

      }

      }

}
```

## 3. Referências Bibliográficas

BORATTI, Isaias Camilo. **Programação Orientada a Objetos usando Delphi.** Quarta Edição. Editora Visual Books. Florianópolis, 2007.

DEITEL, H.M., DEITEL, P.J. **Java – Como programar**. Terceira edição. Porto Alegre: Bookman Editora, 2001.

SANTOS, Rafael. **Introdução à Programação orientada a objetos usando Java**. 8ª Reimpressão. Rio de Janeiro: Campus - Elsevier, 2003.