Programação em Java - Fundamentos
7 - Introdução ao Swing

V2.0

Citeforma

Jose Aser Lorenzo

jose.l.aser@sapo.pt

Abril de 2012

Sumário

Introdução ao Swing

citeforma

Programação em Java
Fundamentos

Capítulo 7 – Introdução ao Swing

José Aser Lorenzo
Pedro Nunes
Paulo Jorge Martins

Objetivos

Objectivos

- Conhecer as propriedades dos objetos que permitem construir interfaces gráficas;
- Distribuir esses objetos numa janela (ou outro "container") e gerir os eventos que o utilizador provoca sobre eles;
- Construir programas que interagem com o utilizador usando uma interface gráfica;
- Construir interfaces gráficas que correm em todas as plataformas onde há um JRE;
- É um resumo do curso de Java Swing

Java Fundamentos Capítulo 7 - Introdução ao Swing 1

O slide descreve os objetivos deste capítulo. As classes desenvolvidas nos exercícios deste capítulo deverão ficar dentro do projeto **JavaFundamentos** e dentro do package **capitulo7**.

A primeira versão de componentes gráficas da linguagem Java era o AWT (Advanced Windowing Toolkit). As suas classes estão armazenadas no package java.awt.

Como o AWT tinha vários problemas a SUN desenvolveu o Swing, também conhecido por JFC (Java Foundation Classes).

Tanto o AWT como o Swing permite construir interfaces gráficas que herdam o aspeto do sistema operativo hospedeiro ou que desenham o seu próprio "look".

Este capítulo pretende fazer uma introdução às potencialidades e funcionamento do AWT/Swing. Este tema é muito vasto, sendo estudado com maior detalhe num curso específico que consta da nossa oferta formativa.

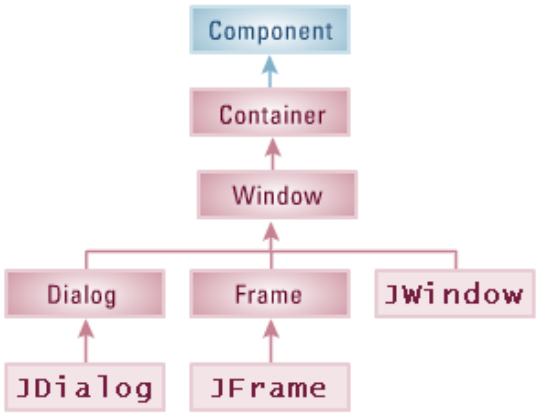
JFrame, JDialog e JWindow

Sumário

- JFrame, JDialog e JWindow;
- JPanel e JScrollPane
- Alguns componentes gráficos do package swing:
  - JLabel;
  - JButton;
  - JTextField e JTextArea;
  - JComboBox;
  - JCheckBox e JRadioButton;
- Distribuição dos componentes;
- Gestão de eventos;

JFrame, JDialog e JWindow

- As classes JFrame, JDialog e JWindow são **top-level containers** que permitem criar janelas;
- Pertencem ao package javax.swing, sendo evoluções das classes Frame, Dialog e Window do package java.awt;



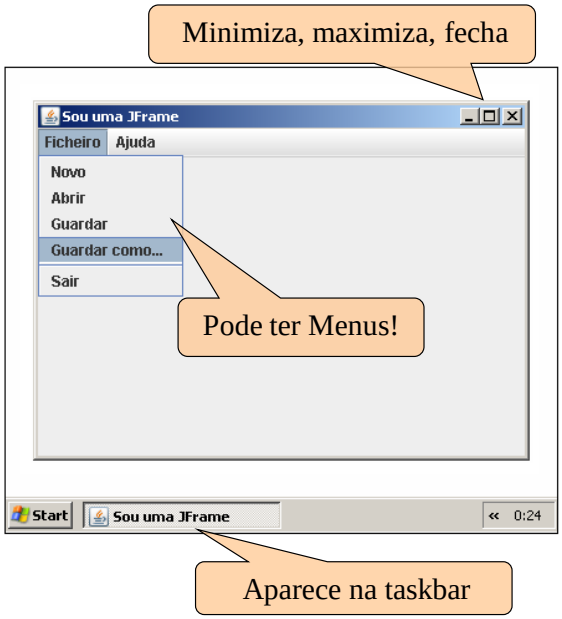
Os nomes das classes do package javax.swing começam por **J**

Os objetos destas classes são Windows que é uma particularização de Container, ou seja, podem receber Componentes. Repare que um Container é ele próprio um Component.

JFrame

JFrame

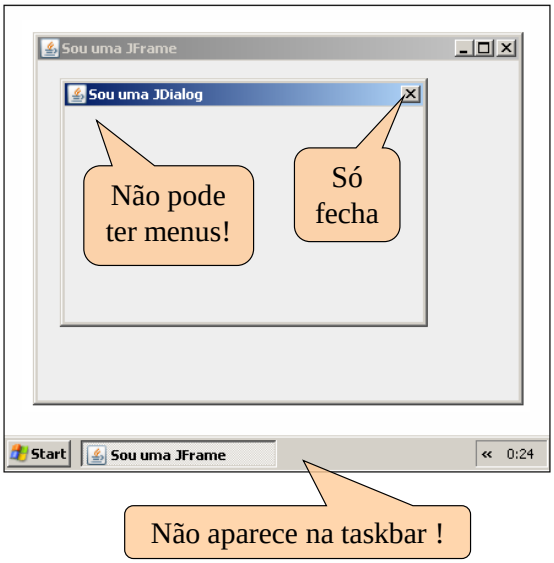
- Tem um título;
- Maximizável e minimizável para a barra de tarefas;
- Pode ter menus "pull-down";
- É usada como janela principal das aplicações;



JDialog

JDialog

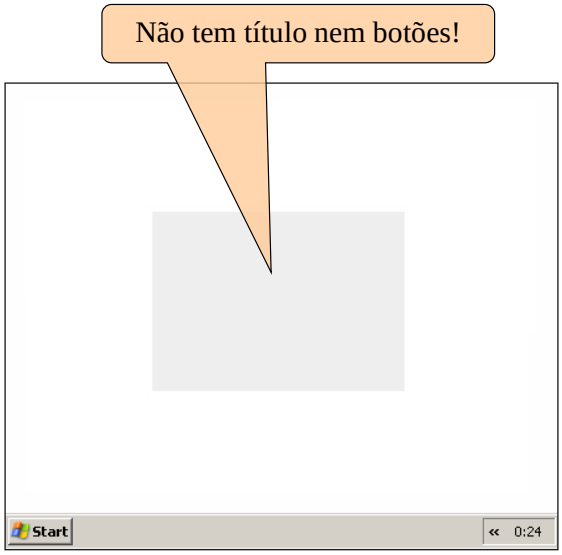
- Tem um título;
- Não maximiza nem minimiza, só fecha;
- Não pode ter menus;
- É usada como Janela secundária (auxiliar);
- Dependente de uma JFrame ou de outra JDialog;
- Pode ser modal ou não modal;



JWindow

JWindow

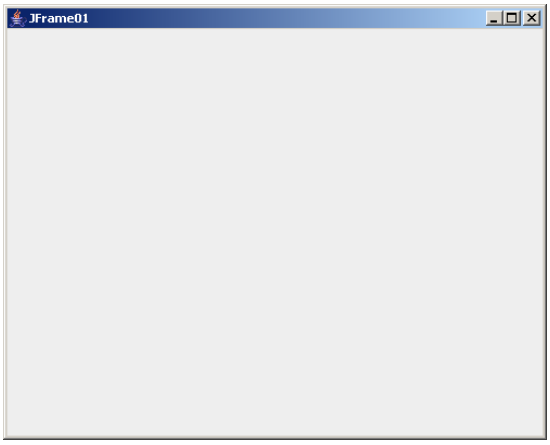
- Não tem título, não é deslocável nem tem botões;
- Não aparece na taskbar nem pode ter menus;
- Usada como "Splash-screen" no arranque das aplicações;



Exercício com JFrame

Exercício com JFrame #1/2

- Criar a classe JFrame01 no package capitulo7;
- Introduzir o código do próximo slide;
- Verificar quando é impresso o texto;



Exercício com JFrame #2/2

```
package capitulo7;
import javax.swing.JFrame;
public class JFrame01 {
    public static void main(String a[ ]) {
        JFrame f = new JFrame();
        f.setTitle("JFrame01");
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setSize(500,400);
        f.setLocationRelativeTo(null);
        f.setVisible(true);
        System.out.println("JFrame01 visível");
    }
}
```

Cria JFrame

Define "Title"

Termina quando se fecha a janela

Define dimensão

Centra

Torna a janela visível

```
 1 package capitulo7;
 2
 3 import javax.swing.JFrame;
 4
 5 public class JFrame01 {
 6
 7    public static void main(String a[]) {
 8       JFrame f = new JFrame();
 9       f.setTitle("JFrame01");
10       f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
11       f.setSize(500, 400);
12       f.setLocationRelativeTo(null);
13       f.setVisible(true);
14       System.out.println("JFrame01 visível");
15    }
16 }
```

Exercício com JDialog

Exercício com JDialog #1/2

- Criar a classe JDialog01 no package capitulo7;
- Introduzir o código do próximo slide;
- Verificar o comportamento da JDialog com o parâmetro modal;

Exercício com JDialog #2/2

```
package capitulo7;
import javax.swing.JDialog;
import javax.swing.JFrame;
public class JDialog01 {
   public static void main(String[] args) {
      JFrame f = new JFrame("Uma JFrame");
      f.setSize(500, 400);
      f.setDefaultCloseOperation(JDialog.EXIT_ON_CLOSE);
      f.setLocationRelativeTo(null);

      JDialog d = new JDialog(f, "Uma JDialog", true);
      d.setSize(300, 200);
      d.setLocationRelativeTo(f);

      f.setVisible(true);
      d.setVisible(true);
      System.out.println("Dialog01 visível");
   }
}
```

Parent JFrame

Centrar Dialog no Parent

Modal=true Janela parent fica inacessível

```
 1 package capitulo7;
 2
 3 import javax.swing.JDialog;
 4 import javax.swing.JFrame;
 5
 6 public class JDialog01 {
 7
 8    public static void main(String[] args) {
 9       JFrame f = new JFrame("Uma JFrame");
10       f.setSize(500, 400);
11       f.setDefaultCloseOperation(JDialog.EXIT_ON_CLOSE);
12       f.setLocationRelativeTo(null);
13
14       JDialog d = new JDialog(f, "Uma JDialog", true);
15       d.setSize(300, 200);
16       d.setLocationRelativeTo(f);
17
18       f.setVisible(true);
19       d.setVisible(true);
20       System.out.println("Dialog01 visível");
21    }
22 }
```

JPanel e JScrollPane

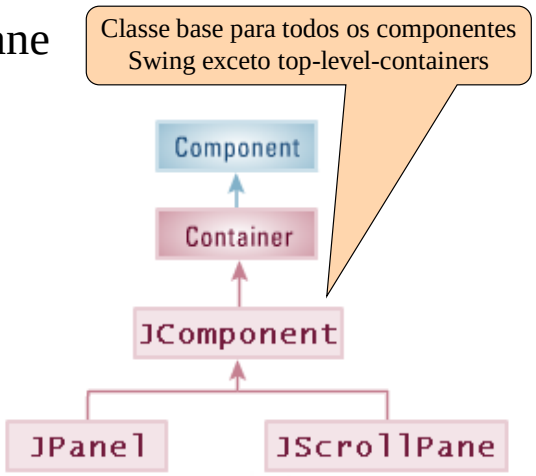
Sumário

- JFrame, JDialog e JWindow;
- JPanel e JScrollPane
- Alguns componentes gráficos do package swing:
  - JLabel;
  - JButton;
  - JTextField;
  - JComboBox;
  - JCheckBox e JRadioButton;
- Distribuição dos componentes;
- Gestão de eventos;

JPanel e JScrollPane

JPanel e JScrollPane

- As classes JPanel e JScrollPane são Containers;
- Para serem apresentadas é preciso adicioná-las a um top-level container (ex: JFrame e JDialog);
- Um JPanel serve para colocar componentes gráficos, incluindo outros JPanel;
- JScrollPane é como JPanel mas com scrollbars;



Todos os componentes Swing são extensões da classe JComponent.

Exercício com JPanel

Exercício com JPanel #1/2

- Criar a classe JFrame02 no package capitulo7;
- Criar uma JFrame e colocar um JPanel amarelo como seu Container (usar o ContentPane da JFrame);
- Introduzir o código do próximo slide;

Exercício com JPanel #2/2

```
package capitulo7;
import java.awt.Color;
import java.awt.Container;
import javax.swing.JFrame;
import javax.swing.JPanel;

public class JFrame02 extends JFrame {

   public JFrame02(String title, Container c, int width, int height) {
      super(title);
      this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      this.setContentPane(c);
      this.setSize(width, height);
      this.setLocationRelativeTo(null);
   }
   public static void main(String[] args) {
      JPanel panel = new JPanel();
      panel.setBackground(Color.YELLOW);
      JFrame02 f = new JFrame02("JPanel dentro de JFrame", panel, 300, 200);
      f.setVisible(true);
   }
}
```

Por herança JFrame02 é também uma JFrame

O Container é passado como argumento para o construtor

Coloca o Container como ContentPane da JFrame

Cria um JPanel

Altera-lhe a cor de fundo

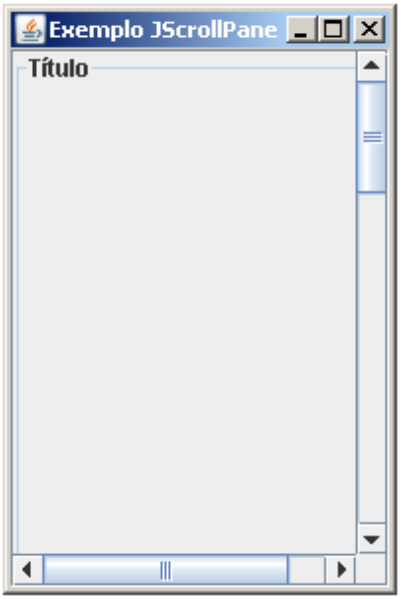
Coloca-o como container de JFrame02

```
 1 package capitulo7;
 2
 3 import java.awt.Color;
 4 import java.awt.Container;
 5 import javax.swing.JFrame;
 6 import javax.swing.JPanel;
 7
 8 public class JFrame02 extends JFrame {
 9
10    public JFrame02(String title, Container c, int width, int height) {
11       super(title);
12       this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
13       this.setContentPane(c);
14       if (width>0 && height>0) {
15          this.setSize(width, height);
16       } else {
17          this.pack();
18       }
19       this.setLocationRelativeTo(null);
20    }
21
22    public static void main(String[] args) {
23       JPanel panel = new JPanel();
24       panel.setBackground(Color.YELLOW);
25       JFrame02 f = new JFrame02("JPanel dentro de JFrame", panel, 300, 200);
26       f.setVisible(true);
27    }
28 }
```

Exercício com JScrollPane

Exercício com JScrollPane #1/2

- Criar a classe JScrollPane01 no package capitulo7;
- Colocar um JScrollPane como Container da JFrame02;
- Introduzir o código do próximo slide;



Exercício com JScrollPane #2/2

```
package capitulo7;

import java.awt.Dimension;
import javax.swing.BorderFactory;
import javax.swing.JPanel;
import javax.swing.JScrollPane;

public class JScrollPane01 {

   public static void main(String[] args) {
      JPanel panel = new JPanel();
      panel.setBorder(BorderFactory.createTitledBorder("Título"));
      Dimension dim = new Dimension(200, 1000);
      panel.setPreferredSize(dim);

      JScrollPane sp = new JScrollPane(panel);

      JFrame02 f = new JFrame02("Exemplo JScrollPane", sp, 200, 300);
      f.setVisible(true);
   }
}
```

Cria um Border dentro de JPanel para tornar visível a sua dimensão

Cria um objecto dimensão para guardar largura e altura

Indica a dimensão desejada para o panel

Cria uma JScrollPane para o JPanel

Coloca a JScrollPane como ContentPane de JFrame02

```
 1 package capitulo7;
 2
 3 import java.awt.Dimension;
 4 import javax.swing.BorderFactory;
 5 import javax.swing.JPanel;
 6 import javax.swing.JScrollPane;
 7
 8 public class JScrollPane01 {
 9
10    public static void main(String[] args) {
11       JPanel panel = new JPanel();
12       panel.setBorder(BorderFactory.createTitledBorder("Título"));
13       Dimension dim = new Dimension(200, 1000);
14       panel.setPreferredSize(dim);
15
16       JScrollPane sp = new JScrollPane(panel);
17
18       JFrame02 f = new JFrame02("Exemplo JScrollPane", sp, 200, 300);
19       f.setVisible(true);
20    }
21 }
```

Alguns componentes gráficos do package swing

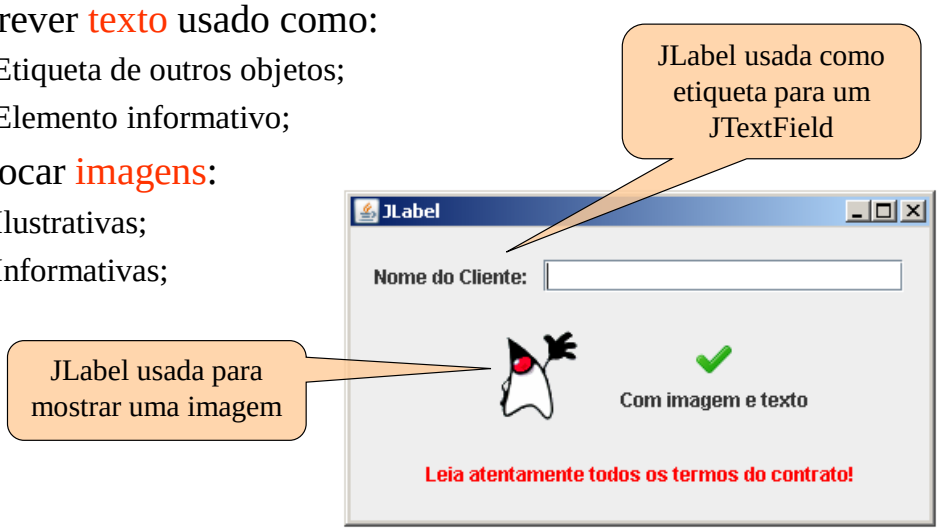
Sumário

- JFrame, JDialog e JWindow;
- JPanel e JScrollPane
- Alguns componentes gráficos do package swing:
  - JLabel;
  - JButton;
  - JTextField e JTextArea;
  - JComboBox;
  - JCheckBox e JRadioButton;
- Distribuição dos componentes;
- Gestão de eventos;

JLabel

Componentes gráficos - JLabel

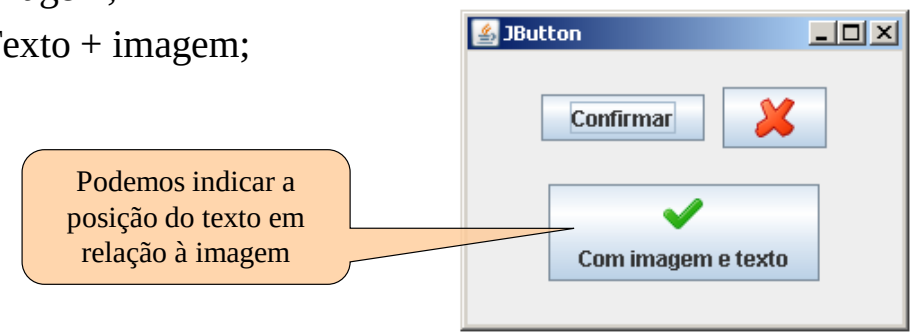
- O componente JLabel permite:
  - Escrever texto usado como:
    - Etiqueta de outros objetos;
    - Elemento informativo;
  - Colocar imagens:
    - Ilustrativas;
    - Informativas;



JButton

Componentes gráficos - JButton

- A classe JButton permite criar botões;
- Um botão pode ter:
  - Texto;
  - Imagem;
  - Texto + imagem;



JTextField

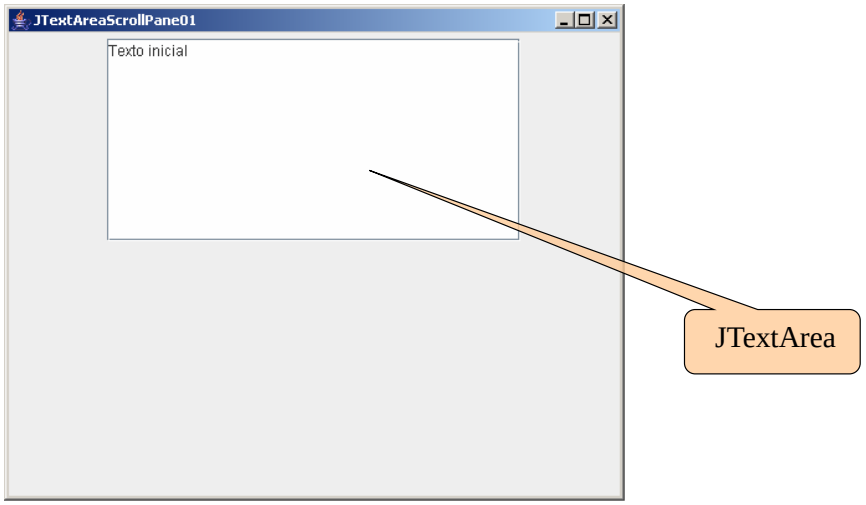
Componentes gráficos - JTextField

- O componente JTextField permite ao utilizador introduzir uma linha de texto;

JTextArea

Componentes gráficos - JTextArea

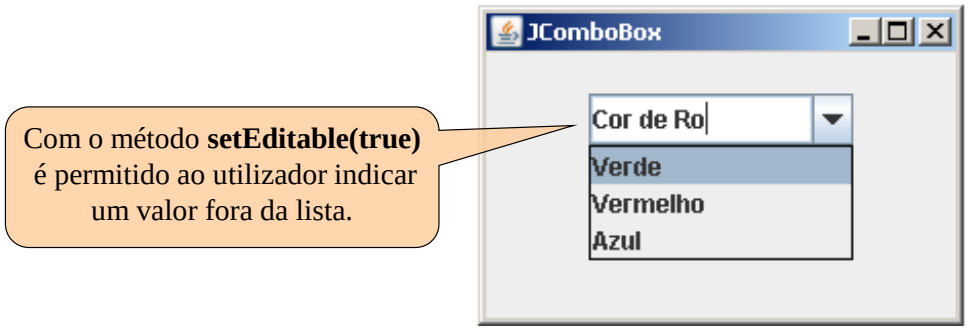
- O componente JTextField permite ao utilizador introduzir várias linhas de texto;



JComboBox

Componentes gráficos - JComboBox

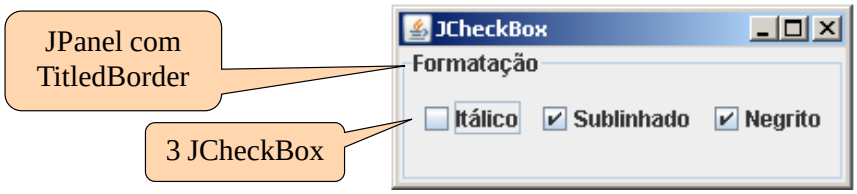
- O componente JComboBox combina as funcionalidades de uma caixa de texto com as de uma lista de valores selecionáveis;



JCheckBox

Componentes gráficos - JCheckBox

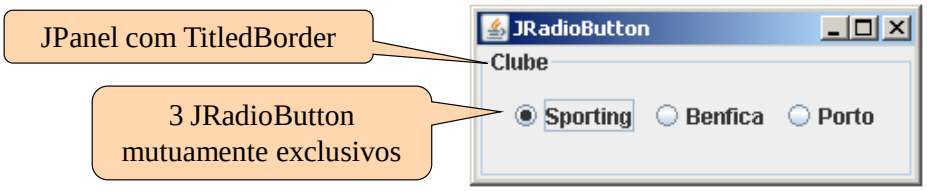
- O componente JCheckBox permite ao utilizador selecionar (marcar) ou não uma opção;
- Tem dois estados:
  - Selecionada;
  - Não selecionada;
- Havendo várias JCheckBox o utilizador pode selecionar todas, nenhuma ou só algumas;



JRadioButton

Componentes gráficos - JRadioButton

- O JRadioButton é usado para que o utilizador selecione apenas uma das opções;
- Tem dois estados:
  - Selecionado;
  - Não selecionado;
- Ao contrário da JCheckBox que pode aparecer isolada, um JRadioButton aparece sempre em grupo;



Exercício com JLabel, JButton e JTextField

Exercício com JLabel, Jbutton e JTextField #1/2

- Criar a classe JButtonJLabelJTextField01 no package capitulo7 ;
- Colocar uma JLabel, um JTextField e um JButton dentro de um JPanel;
- Adicionar o JPanel como Container de JFrame02;

- Introduzir o código do próximo slide;

Exercício com JLabel, Jbutton e JTextField #1/2

```
package capitulo7;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;

public class JButtonJLabelJTextField01 {
   public static void main(String[] args) {
      JLabel lbNome = new JLabel("Nome:");
      JTextField tfNome = new JTextField(20);
      JButton btConfirmar = new JButton("Confirmar");
      JPanel p = new JPanel();
      p.add(lbNome);
      p.add(tfNome);
      p.add(btConfirmar);

      JFrame02 f = new JFrame02("Componentes Swing 1", p, 300, 100);
      f.setVisible(true);
   }
}
```

Criar JLabel

Criar JTextField com 20 colunas visíveis

Criar JButton

O método add() de JPanel permite adicionar componentes ao Container

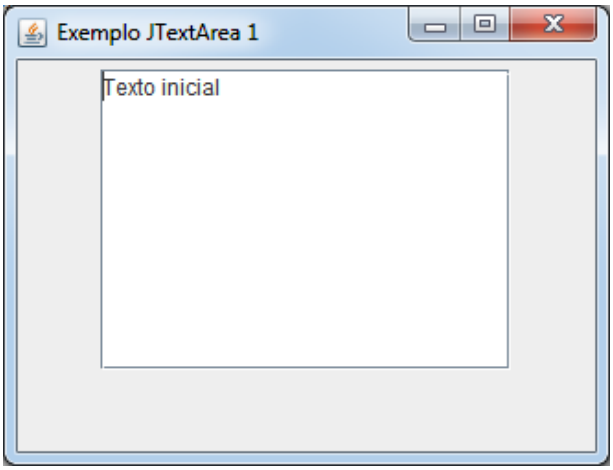
Coloca JPanel como Container de JFrame02

```
 1 package capitulo7;
 2 import javax.swing.JButton;
 3 import javax.swing.JLabel;
 4 import javax.swing.JPanel;
 5 import javax.swing.JTextField;
 6
 7 public class JButtonJLabelJTextField01 {
 8    public static void main(String[] args) {
 9       JLabel lbNome = new JLabel("Nome:");
10       JTextField tfNome = new JTextField(20);
11       JButton btConfirmar = new JButton("Confirmar");
12       JPanel p = new JPanel();
13       p.add(lbNome);
14       p.add(tfNome);
15       p.add(btConfirmar);
16
17       JFrame02 f = new JFrame02("Componentes Swing 1", p, 300, 100);
18       f.setVisible(true);
19    }
20 }
```

Exercício com JTextArea

Exercício com JTextArea #1/2

- Criar a classe JTextArea01 no package capitulo7;
- Colocar uma JTextArea dentro de um JPanel;
- Adicionar o JPanel como Container de JFrame02;
- Introduzir o código do próximo slide;



Exercício com JTextArea #1/2

```
package capitulo7;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;

public class JTextArea01 {
   public static void main(String[] args) {
      int nColunas = 20; int nLinhas = 10;
      JTextArea ta = new JTextArea("Texto inicial", nLinhas, nColunas);
      ta.setEditable(true);
      JScrollPane sp = new JScrollPane(ta);
      JPanel p = new JPanel();
      p.add(sp);

      JFrame02 f = new JFrame02("Exemplo JTextArea 1", p, 330, 250);
      f.setVisible(true);
   }
}
```

Criar JTextArea

Tornar editável

Criar um JScrollPane e adicionar o JPanel

Adicionar JScrollPane ao JPanel

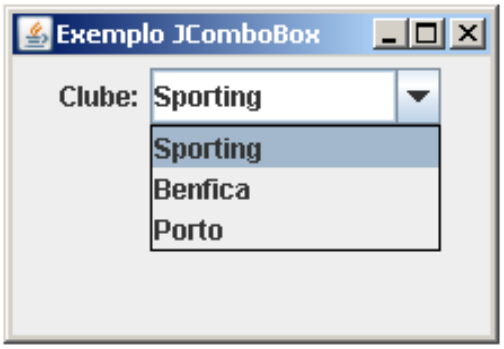
Coloca JPanel como Container de JFrame02

```
 1 package capitulo7;
 2 import javax.swing.JPanel;
 3 import javax.swing.JScrollPane;
 4 import javax.swing.JTextArea;
 5
 6 public class JTextArea01 {
 7    public static void main(String[] args) {
 8       int nColunas = 20; int nLinhas = 10;
 9       JTextArea ta = new JTextArea("Texto inicial", nLinhas, nColunas);
10       ta.setEditable(true);
11       JScrollPane sp = new JScrollPane(ta);
12       JPanel p = new JPanel();
13       p.add(sp);
14
15       JFrame02 f = new JFrame02("Exemplo JTextArea 1", p, 330, 250);
16       f.setVisible(true);
17    }
18 }
```

Exercício com JComboBox

Exercício com JComboBox #1/2

- Criar a classe JComboBox01 no package capitulo7;
- Colocar uma JComboBox num JPanel;
- Colocar o JPanel como Container de JFrame02;
- Introduzir o código do próximo slide;



Exercício com JComboBox #2/2

```
package capitulo7;
import javax.swing.JComboBox;
import javax.swing.JLabel;
import javax.swing.JPanel;

public class JComboBox01 {
   public static void main(String[] args) {
      JLabel lab = new JLabel("Clube:");
      String[] clubes = {"Sporting", "Benfica", "Porto"};
      JComboBox cbClubes = new JComboBox(clubes);
      cbClubes.setEditable(true);
      JPanel p = new JPanel();
      p.add(lab);
      p.add(cbClubes);

      JFrame02 f = new JFrame02("Exemplo JComboBox 1", p, 330, 150);
      f.setVisible(true);
   }
}
```

Array de valores para a JComboBox

Cria a JComboBox com o array indicado

permitir valores fora da lista

Adiciona ao Container

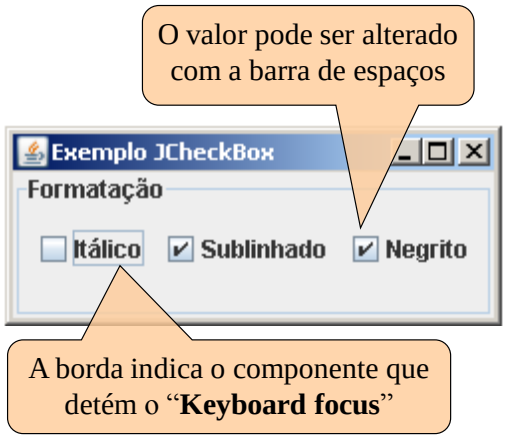
Coloca JPanel como Container de JFrame02

```
 1 package capitulo7;
 2 import javax.swing.JComboBox;
 3 import javax.swing.JLabel;
 4 import javax.swing.JPanel;
 5
 6 public class JComboBox01 {
 7    public static void main(String[] args) {
 8       JLabel lab = new JLabel("Clube:");
 9       String[] clubes = {"Sporting", "Benfica", "Porto"};
10       JComboBox cbClubes = new JComboBox(clubes);
11       cbClubes.setEditable(true);
12       JPanel p = new JPanel();
13       p.add(lab);
14       p.add(cbClubes);
15
16       JFrame02 f = new JFrame02("Exemplo JComboBox 1", p, 330, 150);
17       f.setVisible(true);
18    }
19 }
```

Exercício com JCheckBox

Exercício com JCheckBox #1/2

- Criar a classe JCheckBox01 no package capitulo7;
- Colocar uma TitledBorder e três JCheckBox num JPanel;
- Colocar o JPanel como Container de JFrame02;
- Introduzir o código do próximo slide;



Exercício com JCheckBox #2/2

```
package capitulo7;
import javax.swing.BorderFactory;
import javax.swing.JCheckBox;
import javax.swing.JPanel;

public class JCheckBox01 {
   public static void main(String[] args) {
      JCheckBox cb1 = new JCheckBox("Itálico", false);
      JCheckBox cb2 = new JCheckBox("Sublinhado", true);
      JCheckBox cb3 = new JCheckBox("Negrito", true);
      JPanel p = new JPanel();
      p.setBorder(BorderFactory.createTitledBorder("Formatação"));
      p.add(cb1);
      p.add(cb2);
      p.add(cb3);

      JFrame02 f = new JFrame02("Exemplo JCheckBox 1", p, 270, 100);
      f.setVisible(true);
   }
}
```

Criar uma JCheckBox

Etiqueta para a JCheckBox

Pré selecionada

Adiciona ao Container

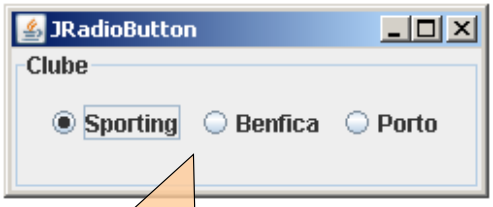
Coloca uma borda com título no JPanel

```
 1 package capitulo7;
 2 import javax.swing.BorderFactory;
 3 import javax.swing.JCheckBox;
 4 import javax.swing.JPanel;
 5
 6 public class JChecBox01 {
 7    public static void main(String[] args) {
 8       JCheckBox cb1 = new JCheckBox("Itálico", false);
 9       JCheckBox cb2 = new JCheckBox("Sublinhado", true);
10       JCheckBox cb3 = new JCheckBox("Negrito", true);
11       JPanel p = new JPanel();
12       p.setBorder(BorderFactory.createTitledBorder("Formatação"));
13       p.add(cb1);
14       p.add(cb2);
15       p.add(cb3);
16
17       JFrame02 f = new JFrame02("Exemplo JChecBox 1", p, 270, 100);
18       f.setVisible(true);
19    }
20 }
```

Exercício com JRadioButton

Exercício com JRadioButton #1/2

- Criar a classe JRadioButton01no package capitulo7;
- Colocar uma TitledBorder e três JRadioButton num JPanel;
- Adicionar um JPanel como Container de JFrame02;
- Introduzir o código do próximo slide;



Vamos adicionar os JRadioButton a um **ButtonGroup** para garantir que só um botão fica seleccionado.

Exercício com JRadioButton #2/2

```
package capitulo7;
import javax.swing.BorderFactory;
import javax.swing.ButtonGroup;
import javax.swing.JPanel;
import javax.swing.JRadioButton;

public class JRadioButton01 {
   public static void main(String[] args) {
      JRadioButton  rb1 = new JRadioButton ("Sporting", true);
      JRadioButton  rb2 = new JRadioButton ("Benfica", false);
      JRadioButton  rb3 = new JRadioButton ("Porto", false);
      ButtonGroup bg = new ButtonGroup();
      bg.add(rb1); bg.add(rb2); bg.add(rb3);
      JPanel p = new JPanel();
      p.setBorder(BorderFactory.createTitledBorder("Clube"));
      p.add(rb1); p.add(rb2); p.add(rb3);

      JFrame02 f = new JFrame02("Exemplo JRadioButton 1", p, 270, 100);
      f.setVisible(true);
   }
}
```

Cria o JRadioButton

Etiqueta para o JRadioButton

Pré selecionado

Cria um JButtonGroup

Adiciona ao Container

Adiciona ao JButtonGroup para garantir que só um fica selecionado

```
 1 package capitulo7;
 2 import javax.swing.BorderFactory;
 3 import javax.swing.ButtonGroup;
 4 import javax.swing.JPanel;
 5 import javax.swing.JRadioButton;
 6
 7 public class JRadioButton01 {
 8    public static void main(String[] args) {
 9       JRadioButton  rb1 = new JRadioButton ("Sporting", true);
10       JRadioButton  rb2 = new JRadioButton ("Benfica", false);
11       JRadioButton  rb3 = new JRadioButton ("Porto", false);
12       ButtonGroup bg = new ButtonGroup();
13       bg.add(rb1); bg.add(rb2); bg.add(rb3);
14       JPanel p = new JPanel();
15       p.setBorder(BorderFactory.createTitledBorder("Clube"));
16       p.add(rb1); p.add(rb2); p.add(rb3);
17
18       JFrame02 f = new JFrame02("Exemplo JRadioButton 1", p, 270, 100);
19       f.setVisible(true);
20    }
21 }
```

Distribuição dos componentes

Sumário

- JFrame, JDialog e JWindow;
- JPanel e JScrollPane;
- Alguns componentes gráficos do package swing:
  - JLabel;
  - JButton;
  - JTextField e JTextArea;
  - JComboBox;
  - JCheckBox e JRadioButton;
- Distribuição dos componentes;
- Gestão de eventos;

Distribuição dos Componentes #1/2

- O Java possui vários "**LayoutManagers**" (**LM**) que permitem posicionar automaticamente os componentes num Container;
- A cada Container está associado por omissão um **LayoutManager**, mas podemos trocá-lo por outro;
- Os **LMs** permitem lidar de forma automática com diferentes tamanhos e resoluções de ecrãs;
- Um JPanel pode conter outros JPanel, e cada um deles pode ter o seu próprio **LayoutManager**;

Os componentes vistos no ponto anterior têm que ser arrumados num container. A forma mais fácil de arrumar um componente é pela definição da sua posição dentro do container e da sua dimensão. Para isso usamos um referencial com coordenadas em pixel para indicar a posição e dois números inteiros em pixel para indicar a dimensão.

Mas este mecanismo tem a vantagem da precisão e o inconveniente da flexibilidade. A linguagem Java foi desenvolvida para produzir código independente do sistema operativo hospedeiro, o que implica um enorme desafio de adaptação aos diferentes ambientes de execução, pela multiplicidade de dimensões de dispositivos de output e respetivas resoluções em pixel.

Os LayoutManagers são a resposta para dar flexibilidade.

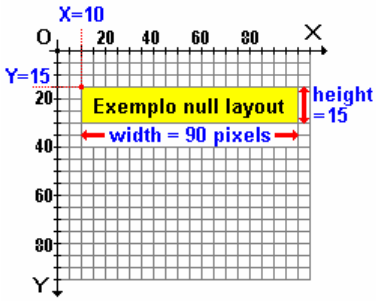
Sem LayoutManager

Sem LayoutManager

- Desativar o LayoutManager para **controlar totalmente** a posição e o tamanho do componente:
  - Indicar **null** no método **setLayout()** do Container;
  - Usar **setBounds(x, y, w, h)** no componente;
- O componente **não redimensiona** com a alteração do Container respetivo;

```
...
JPanel panel = new JPanel();
panel.setLayout(null);
JLabel label = new JLabel("Exemplo null layout");
label.setBounds(10, 15, 90, 15);
panel.add(label);
...
```

(x, y, **largura**, **altura**)

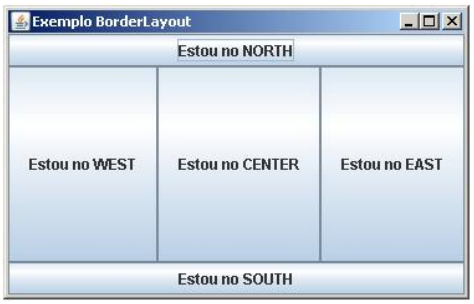


O exemplo do slide mostra como anular o LayoutManager que está definido por omissão num container e como usar o método setBounds() para definir a posição e a dimensão de um component dentro de um container. Todos os components têm este método.

LayoutManagers

LayoutManagers

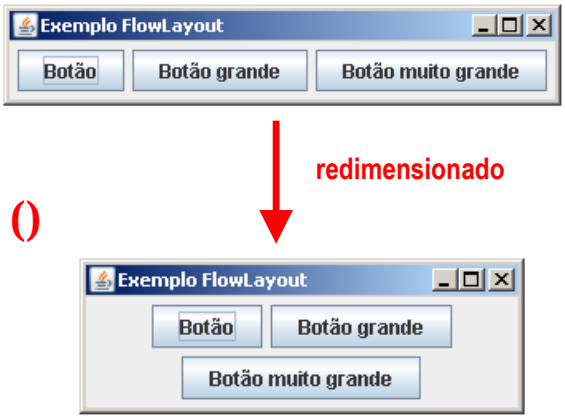
- **FlowLayout**, **BoxLayout**, **BorderLayout** e **GridLayout**, entre outros;
- Usar **setLayout()** para definir o LayoutManager do Container;



FlowLayout

LayoutManagers - FlowLayout

- Os componentes são distribuídos em linha, da esquerda para a direita;
- Por omissão os componentes ficam centrados;
- Respeita o **preferredSize ()** dos componentes (não crescem);
- É o LayoutManager por omissão do JPanel;



Exercício com FlowLayout

Exercício com FlowLayout #1/2

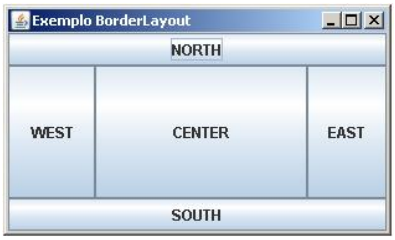
- Criar a classe FlowLayout01 no package capitulo7;
- Criar um JPanel e definir FlowLayout como LayoutManager;
- Criar 3 botões e adicionar ao JPanel ;
- Criar um objeto JFrame02 e adicionar o JPanel anterior ao seu ContentPane;

```
 1 package capitulo7;
 2 import java.awt.FlowLayout;
 3 import javax.swing.JButton;
 4 import javax.swing.JPanel;
 5
 6 public class FlowLayout01 {
 7    public static void main(String[] args) {
 8       JButton jb1 = new JButton("Botão");
 9       JButton jb2 = new JButton("Botão grande");
10       JButton jb3 = new JButton("Botão muito grande ");
11       JPanel p = new JPanel();
12       p.setLayout(new FlowLayout());
13       p.add(jb1);
14       p.add(jb2);
15       p.add(jb3);
16
17       JFrame02 f = new JFrame02("Exemplo FlowLayout", p, 400, 100);
18       f.setVisible(true);
19    }
20 }
```

BorderLayout

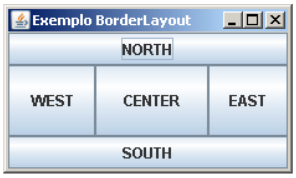
LayoutManagers – BorderLayout #1/2

- É o LayoutManager por omissão dos **top-level-containers** (JFrame, JDialog e JWindow);
- Um Container com este LayoutManager pode receber no máximo 5 componentes: **norte, sul, oeste, este e centro**;
- Os componentes que podem crescer vão ocupar o espaço disponível dentro da sua quadricula;

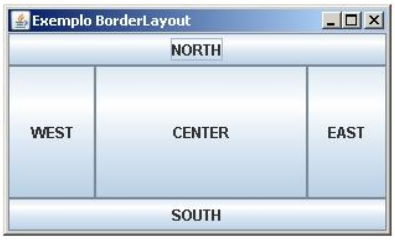


LayoutManagers - BorderLayout #2/2

- Comportamento do BorderLayout com o redimensionamento:

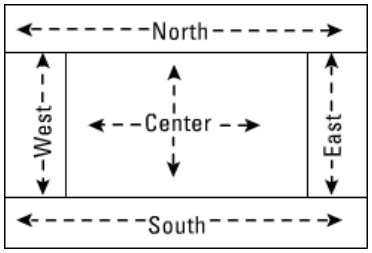


redimensionando



• Os componentes ajustam-se:

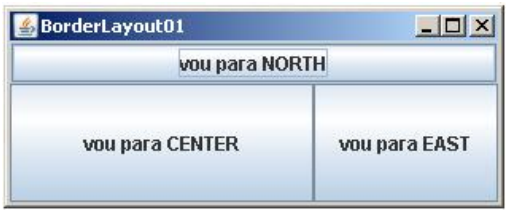
- •Em NORTH e SOUTH horizontalmente;
- •Em WEST e EAST verticalmente;
- • Em CENTER ocupa o espaço restante;



Exercício com BorderLayout

Exercício com BorderLayout #1/2

- Criar a classe BorderLayout01 no package capitulo7;
- Criar um JPanel e definir BorderLayout como LayoutManager;
- Criar 3 botões e adicionar ao JPanel nas posições NORTH, CENTER e EAST;



- Criar uma JFrame02 e adicionar o JPanel anterior ao seu ContentPane;

Exercício com BorderLayout #2/2

```
package capitulo7;
import java.awt.BorderLayout;
import javax.swing.JButton;
import javax.swing.JPanel;

public class BorderLayout01 {
   public static void main(String[] args) {
      JButton jb1 = new JButton("vou para NORTH");
      JButton jb2 = new JButton("vou para CENTER");
      JButton jb3 = new JButton("vou para EAST");
      JPanel p = new JPanel();
      p.setLayout(new BorderLayout());
      p.add(jb1,BorderLayout.NORTH);
      p.add(jb2,BorderLayout.CENTER);
      p.add(jb3,BorderLayout.EAST);

      JFrame02 f = new JFrame02("Exemplo BorderLayut", p, 300, 200);
      f.setVisible(true);
   }
}
```

Cria os JButton

Define o LayoutManager de JPanel

Adiciona o JButton à área norte do Container

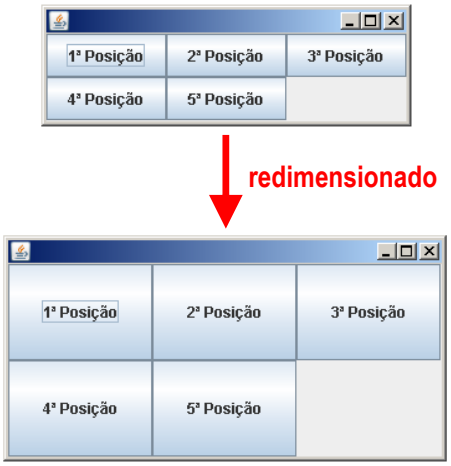
Coloca JPanel como ContentPane de JFrame02

```
 1 package capitulo7;
 2 import java.awt.BorderLayout;
 3 import javax.swing.JButton;
 4 import javax.swing.JPanel;
 5
 6 public class BorderLayout01 {
 7    public static void main(String[] args) {
 8       JButton jb1 = new JButton("vou para NORTH");
 9       JButton jb2 = new JButton("vou para CENTER");
10       JButton jb3 = new JButton("vou para EAST");
11       JPanel p = new JPanel();
12       p.setLayout(new BorderLayout());
13       p.add(jb1,BorderLayout.NORTH);
14       p.add(jb2,BorderLayout.CENTER);
15       p.add(jb3,BorderLayout.EAST);
16
17       JFrame02 f = new JFrame02("Exemplo BorderLayout", p, 300, 200);
18       f.setVisible(true);
19    }
20 }
```

GridLayout

LayoutManager - GridLayout

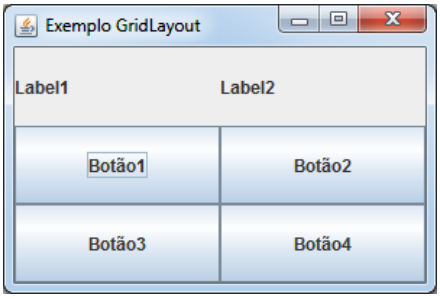
- O construtor deste LM recebe o nº de **linhas** e nº de **colunas** para criar uma **grelha** de células;
- Todas as células têm a **mesma área**;
- Os componentes são distribuídos da esquerda para a direita, de cima para baixo;
- Os componentes que podem crescer ocupam todo o espaço livre;



Exercício com GridLayout

Exercício com GridLayout #1/2

- Criar a classe GridLayout01 no package capitulo7;
- Criar um JPanel e definir GridLayout como LayoutManager. A grelha deve ter 3 linhas e 2 colunas;
- Criar 2 JLabel, 4 botões e adicionar ao JPanel;
- Introduzir o código do próximo slide;



Exercício com GridLayout #2/2

```
package capitulo7;
import java.awt.GridLayout;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JPanel;

public class GridLayout01 {
   public static void main(String[] args) {
      JLabel jla1 = new JLabel("Label1");   JLabel jla2 = new JLabel("Label2");
      JButton jb1 = new JButton("Botão1"); JButton jb2 = new JButton("Botão2");
      JButton jb3 = new JButton("Botão3"); JButton jb4 = new JButton("Botão4");
      JPanel p = new JPanel();
      p.setLayout(new GridLayout(3,2));
      p.add(jla1); p.add(jla2);
      p.add(jb1); p.add(jb2);
      p.add(jb3); p.add(jb4);

      JFrame02 f = new JFrame02("Exemplo GridLayout", p, 300, 200);
      f.setVisible(true);
   }
}
```

Criar JLabel. No GridLayout este objeto não cresce e fica encostado à esquerda

Criar Jbutton. No GridLayout este objeto cresce

Define GridLayout no JPanel

Adiciona objetos ao Container. A ordem de add() é importante

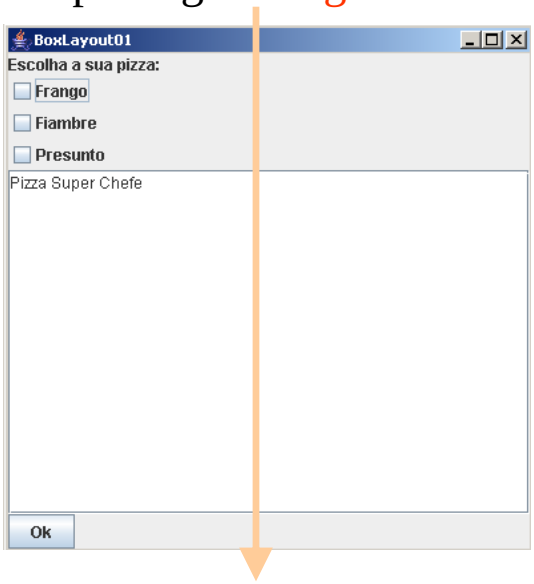
Coloca JPanel como ContentPane de JFrame02

```
 1 package capitulo7;
 2 import java.awt.GridLayout;
 3 import javax.swing.JButton;
 4 import javax.swing.JLabel;
 5 import javax.swing.JPanel;
 6
 7 public class GridLayout01 {
 8    public static void main(String[] args) {
 9       JLabel jla1 = new JLabel("Label1");   JLabel jla2 = new JLabel("Label2");
10       JButton jb1 = new JButton("Botão1"); JButton jb2 = new JButton("Botão2");
11       JButton jb3 = new JButton("Botão3"); JButton jb4 = new JButton("Botão4");
12       JPanel p = new JPanel();
13       p.setLayout(new GridLayout(3,2));
14       p.add(jla1); p.add(jla2);
15       p.add(jb1); p.add(jb2);
16       p.add(jb3); p.add(jb4);
17
18       JFrame02 f = new JFrame02("Exemplo GridLayout", p, 300, 200);
19       f.setVisible(true);
20    }
21 }
```

BoxLayout

Layout Manager - BoxLayout

- A classe BoxLayout pertence ao package swing e não ao awt;
- Distribui os objetos seguindo uma linha horizontal ou vertical (LINE_AXIS ou PAGE_AXIS);
- A sequência de add() condiciona a posição do objeto;



BoxLayout - variações no alinhamento

Objetos alinhados numa linha vertical ou horizontal

```
p.setLayout(new BoxLayout(this,BoxLayout.PAGE_AXIS));
//p.setLayout(new BoxLayout(this,BoxLayout.LINE_AXIS));
```

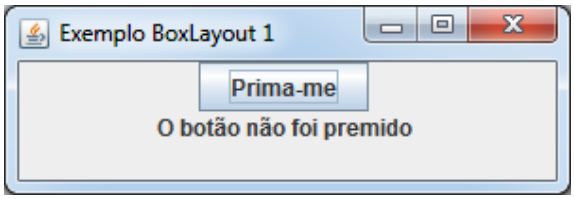
Em relação à linha anterior os objetos são alinhados à esquerda ou ao centro

```
label.setAlignmentX(Component.LEFT_ALIGNMENT);
//label.setAlignmentX(Component.CENTER_ALIGNMENT);
```

Exercício com BoxLayout

Exercício com BoxLayout #1/2

- Criar a classe GridLayout01 no package capitulo7;
- Criar um JPanel e definir BoxLayout como LayoutManager;
- Criar JButton e Jlabel e adicionar ao JPanel anterior, com distribuição vertical e alinhamento ao centro;
- Experimentar outros alinhamentos;



Exercício com BoxLayout #2/2

```
package capitulo7;
import java.awt.Component;
import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JPanel;

public class BoxLayout01 {
   public static void main(String[] args) {
      JButton jb1 = new JButton("Prima-me");
      JLabel lab1 = new JLabel("O botão não foi premido");
      JPanel p = new JPanel();
      p.setLayout(new BoxLayout(p,BoxLayout.PAGE_AXIS));
      jb1.setAlignmentX(Component.CENTER_ALIGNMENT);
      lab1.setAlignmentX(Component.CENTER_ALIGNMENT);
      p.add(jb1); p.add(lab1);

      JFrame02 f = new JFrame02("Exemplo BoxLayout 1", p, 300, 100);
      f.setVisible(true);
   }
}
```

Definir o "layout manager" para este "container": com distribuição vertical

Alinhar os objetos ao centro

Adicionar os objectos ao "container"

Coloca JPanel como ContentPane de JFrame02

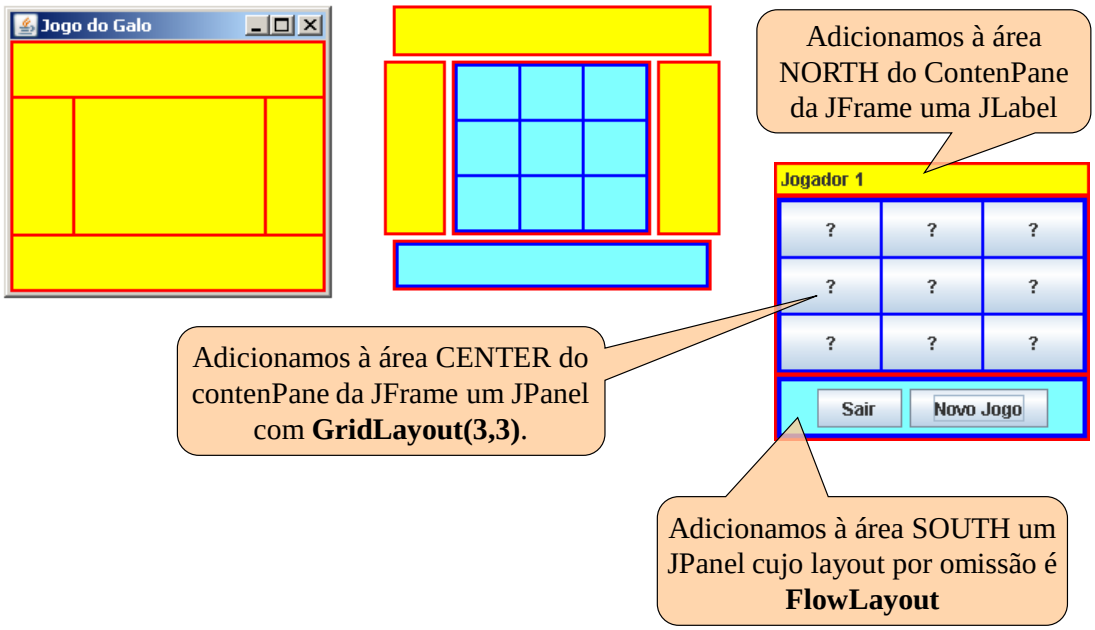
```
 1 package capitulo7;
 2 import java.awt.Component;
 3 import javax.swing.BoxLayout;
 4 import javax.swing.JButton;
 5 import javax.swing.JLabel;
 6 import javax.swing.JPanel;
 7
 8 public class BoxLayout01 {
 9    public static void main(String[] args) {
10       JButton jb1 = new JButton("Prima-me");
11       JLabel lab1 = new JLabel("O botão não foi premido");
12       JPanel p = new JPanel();
13       p.setLayout(new BoxLayout(p,BoxLayout.PAGE_AXIS));
14       jb1.setAlignmentX(Component.CENTER_ALIGNMENT);
15       lab1.setAlignmentX(Component.CENTER_ALIGNMENT);
16       p.add(jb1); p.add(lab1);
17
18       JFrame02 f = new JFrame02("Exemplo BoxLayout 1", p, 300, 100);
19       f.setVisible(true);
20    }
21 }
```

Combinando diferentes LayoutManagers

Combinando diferentes LayoutManagers #1/2

- Num container podemos adicionar componentes ou outros containers, excepto top-level-containers;
- Cada container tem o seu próprio LayoutManager que pode ser diferente do dos outros containers;

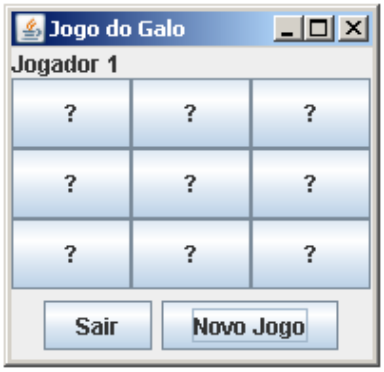
Combinando diferentes LayoutManagers #1/2



Exercício Jogo do Galo

Exercício – Jogo do Galo #1/3

- Criar a classe JogoDoGalo1 no package capitulo7;
- Criar uma JFrame que irá conter no norte a indicação do próximo jogador, ao centro as quadrículas do jogo, e no sul os botões para sair e novo jogo;
- Introduzir o código dos próximos slides;



Exercício – Jogo do Galo #2/3

```
package capitulo7;
import java.awt.BorderLayout;
import java.awt.FlowLayout;
import java.awt.GridLayout;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JPanel;

public class JogoDoGalo1 {
   private JLabel labJogador;
   private JButton bSair;
   private JButton bNovoJogo;
   private JButton quadricula[];

   public static void main(String[] args) {
      JogoDoGalo1 jogo = new JogoDoGalo1();
      jogo.labJogador = new JLabel("Jogador 1");
      jogo.bSair = new JButton("Sair");
      jogo.bNovoJogo = new JButton("Novo jogo");
      JPanel botoesBaixo = new JPanel();
      botoesBaixo.setLayout(new FlowLayout());
      botoesBaixo.add(jogo.bSair);
      botoesBaixo.add(jogo.bNovoJogo);
```

Definir a JLabel e os JButton.

JPanel com os botões de baixo

Exercício – Jogo do Galo #3/3

```
        JPanel tabuleiro = new JPanel();
        tabuleiro.setLayout(new GridLayout(3,3));
        jogo.quadricula = new JButton[9];
        for (int i=0; i<9; i++) {
           jogo.quadricula[i] = new JButton("?");
           tabuleiro.add(jogo.quadricula[i]);
        }

        JPanel pGeral = new JPanel();
        pGeral.setLayout(new BorderLayout());
        pGeral.add(jogo.labJogador, BorderLayout.NORTH);
        pGeral.add(tabuleiro, BorderLayout.CENTER);
        pGeral.add(botoesBaixo, BorderLayout.SOUTH);

        JFrame02 f = new JFrame02("Jogo do Galo", pGeral, 300, 200);
        f.setVisible(true);
    }
}
```

JPanel com os botões do Jogo

O JPanel principal recebe a JLabel e os 2 JPanels

Java Fundamentos

Capítulo 7 - Introdução ao Swing 57

```
 1 package capitulo7;
 2 import java.awt.BorderLayout;
 3 import java.awt.FlowLayout;
 4 import java.awt.GridLayout;
 5 import javax.swing.JButton;
 6 import javax.swing.JLabel;
 7 import javax.swing.JPanel;
 8
 9 public class JogoDoGalo1 {
10    private JLabel labJogador;
11    private JButton bSair;
12    private JButton bNovoJogo;
13    private JButton quadricula[];
14
15    public static void main(String[] args) {
16       JogoDoGalo1 jogo = new JogoDoGalo1();
17       jogo.labJogador = new JLabel("Jogador 1");
18       jogo.bSair = new JButton("Sair");
19       jogo.bNovoJogo = new JButton("Novo jogo");
20       JPanel botoesBaixo = new JPanel();
21       botoesBaixo.setLayout(new FlowLayout());
22       botoesBaixo.add(jogo.bSair);
23       botoesBaixo.add(jogo.bNovoJogo);
24
25       JPanel tabuleiro = new JPanel();
26       tabuleiro.setLayout(new GridLayout(3,3));
27       jogo.quadricula = new JButton[9];
28       for (int i=0; i<9; i++) {
29          jogo.quadricula[i] = new JButton("?");
30          tabuleiro.add(jogo.quadricula[i]);
31       }
32
```

```
33        JPanel pGeral = new JPanel();
34        pGeral.setLayout(new BorderLayout());
35        pGeral.add(jogo.labJogador, BorderLayout.NORTH);
36        pGeral.add(tabuleiro, BorderLayout.CENTER);
37        pGeral.add(botoesBaixo, BorderLayout.SOUTH);
38
39        JFrame02 f = new JFrame02("Jogo do Galo", pGeral, 300, 200);
40        f.setVisible(true);
41    }
42 }
```

Gestão de eventos

Sumário

- JFrame, JDialog e JWindow;
- JPanel e JScrollPane;
- Alguns componentes gráficos do package swing:
  - JLabel;
  - JButton;
  - JTextField;
  - JComboBox;
  - JCheckBox e JRadioButton;
- Distribuição dos componentes;
- Gestão de eventos;

Produtores e consumidores

Produtores e consumidores

- As **ações** dos utilizadores sobre os objetos provocam **eventos**;
- Alguns eventos têm que ser **tratados**;
- O tratamento consiste na programação da resposta;
- Em AWT/Swing a gestão de eventos baseia-se no modelo **produtor/consumidor**:
  - Os objetos da interface gráfica são **produtores** de eventos;
  - Os "listeners" são **consumidores** de eventos;

Ação, evento e reação

Ação, evento e reação

- Cada objeto **reage** a um conjunto de ações do utilizador;
- Essas ações são **desencadeadas** com o rato ou o teclado;

- Cada ação **gera** um evento;
- As propriedades deste evento **caracterizam** a ação:
  - Um click do rato memoriza as coordenadas;
  - Um click do rato sobre num botão devolve um apontador para o botão;
  - Ativar um botão com tecla ESPAÇO devolve apontador para o botão;
  - Ativar um TextField com tecla ENTER devolve apontador para o TF;
- As propriedades do evento permitem **programar a reação**;

- A reação é desencadeada num **método** da interface Listener;

Objetos, ações e eventos

Objetos, ações e eventos #1/3

Objeto	Acção realizada	Evento gerado
AbstractButton (JButton, JMenuItem)	O utilizador accionou o botão usando o rato ou o teclado	ActionEvent
JComboBox	Duplo click	ActionEvent
	Um elemento da lista foi seleccionado	ItemEvent
Document	Texto do documento foi alterado	DocumentEvent
JCheckbox	Um elemento da lista foi seleccionado	ItemEvent
CheckboxMenuItem	Um elemento da lista foi seleccionado	ItemEvent

Objetos, ações e eventos #2/3

Objeto	Acção realizada	Evento gerado
Component	O componente foi movido, escondido ou reapareceu (show)	ComponentEvent
	O componente ganhou ou perdeu o Focus (atenção) da aplicação	FocusEvent
	O utilizador premiu ou soltou uma tecla	KeyEvent
	O utilizador: • premiu ou soltou o botão do rato; • moveu o rato de forma a entrar ou sair da área do objecto; • moveu ou arrastou o ponteiro do rato; Nota: este evento tem dois Listeners	MouseEvent
Container	O componente foi adicionado ou removido do container	ContainerEvent

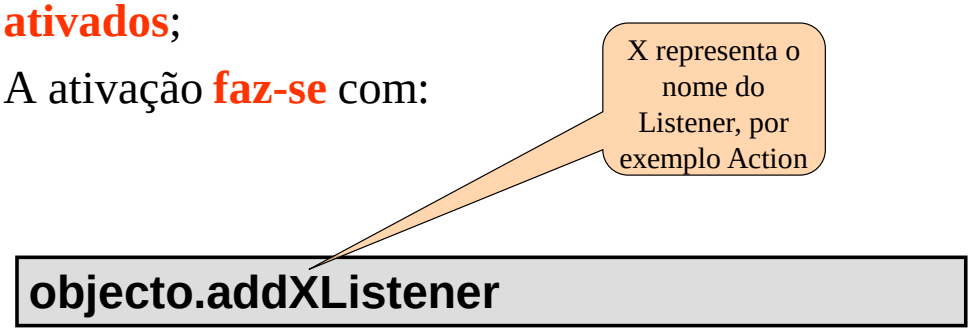
Objetos, ações e eventos #3/3

Objeto	Acção realizada	Evento gerado
JList	O utilizador fez duplo click sobre um item da lista	ActionEvent
	O utilizador marcou ou desmarcou um elemento da lista	ItemEvent
JScrollbar	O utilizador moveu a Scrollbar	AdjustmentEvent
JTextField	O utilizador terminou a introdução de texto com duplo clic ou enter	ActionEvent
Window	A janela foi aberta, fechada, minimizada, reposta ou foi feito um pedido para a fechar	WindowEvent

Ativar um listener sobre o objeto

Ativar um listener sobre o objeto

- Todos os objetos da interface gráfica **geram** eventos:
  - Ex: todos os objetos são Components;
- Alguns eventos **interessam** ao programador;
- Os listeners dos eventos que interessam devem ser **ativados**;
- A ativação **faz-se** com:

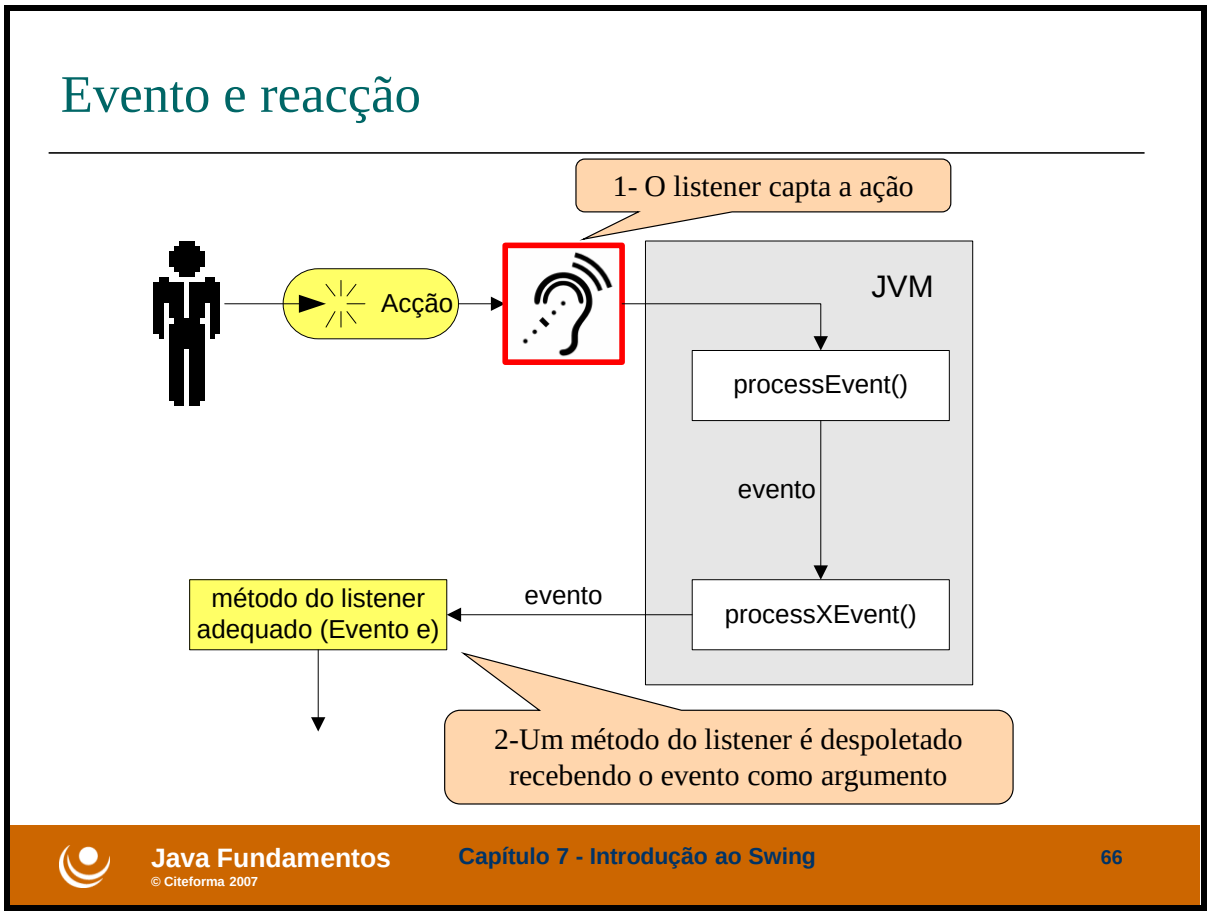


Programar a resposta

Programar a resposta

- O listener é uma **interface**;
- O listener pode definir **vários** métodos;
- Cada método é **especialista** em processar um tipo de evento;
- O método é **desencadeado** automaticamente pela JVM;
- O método recebe o evento como **argumento**;
- As **propriedades** do evento dão informação ao programador;

Evento e reação



Evento, listener e métodos do listener

Evento, listener e métodos do listener #1/2

Classe de Evento	Listener	Métodos do Listener
ActionEvent	ActionListener	actionPerformed()
AdjustmentEvent	AdjustmentListener	adjustmentValueChanged()
ComponentEvent	ComponentListener	componentHidden() componentShown() componentMoved() componentResized()
ContainerEvent	ContainerListener	componentAdded() componentRemoved()
FocusEvent	FocusListener	focusGained() focusLost()
ItemEvent	ItemListener	itemStateChanged()
KeyEvent	KeyListener	keyPressed() keyReleased() keyTyped()

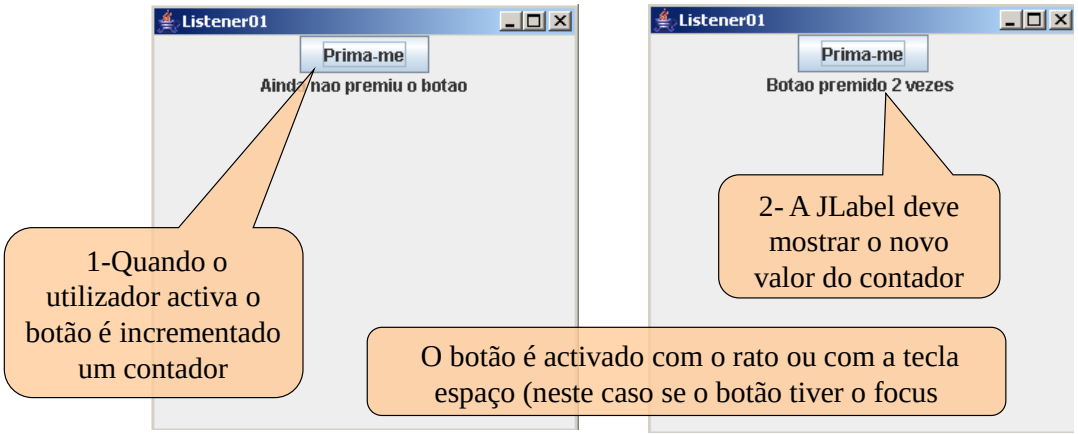
Evento, listener e métodos do listener #1/2

Classe de Evento	Listener	Métodos do Listener
MouseEvent	MouseListener	mouseEntered() mouseExited() mousePressed() mouseReleased() mouseClicked()
	MouseMotionListener	mouseDragged() mouseMoved()
DocumentEvent	DocumentListener	insertUpdate() removeUpdate() changeUpdate()
WindowEvent	WindowFocusListener	windowGainedFocus() windowLostFocus()
	WindowListener	windowActivated() windowClosed() windowClosing() windowDeactivated() windowDeiconified() windowIconified() windowOpened()

Exercício: reagir a uma ação sobre um botão

Exercício: reagir a uma ação num botão #1/3

- Vamos aproveitar a classe BoxLayout, desenvolvida num exercício anterior, que contem um botão e uma label;



Exercício: reagir a uma ação num botão #2/3

```
package capitulo7;
import java.awt.Component;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JPanel;

public class Listener01 implements ActionListener {

   private int premido = 0;
   private JPanel p;
   private JButton jb;
   private JLabel lab;

   public static void main(String[] args)
      Listener01 list = new Listener01();
      list.jb = new JButton("Prima-me");
      list.jb.addActionListener(list);
      list.lab = new JLabel("O botão não foi premido");
```

Não activamos listener sobre a label, pois os seus eventos não nos interessam

Assumir o compromisso de implementar o listener

Variável de objecto para ser usada em vários métodos

Criar o botão

Ativar o listener. Os seus métodos estão na própria classe (porque implementámos o listener)

Exercício: reagir a uma ação num botão #3/3

```
        list.p = new JPanel();
        list.p.setLayout(new BoxLayout(list.p, BoxLayout.PAGE_AXIS));
        list.jb.setAlignmentX(Component.CENTER_ALIGNMENT);
        list.lab.setAlignmentX(Component.CENTER_ALIGNMENT);
        list.p.add(list.jb);
        list.p.add(list.lab);

        JFrame02 f = new JFrame02("Exemplo gestão eventos 1", list.p, 300, 100);
        f.setVisible(true);
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        premido += 1;
        if (premido == 1) {
            this.lab.setText("Botao premido 1 vez");
        } else {
            this.lab.setText("Botao premido " + premido + " vezes ");
        }
    }
}
```

Cumprir o compromisso

```
 1 package capitulo7;
 2 import java.awt.Component;
 3 import java.awt.event.ActionEvent;
 4 import java.awt.event.ActionListener;
 5 import javax.swing.BoxLayout;
 6 import javax.swing.JButton;
 7 import javax.swing.JLabel;
 8 import javax.swing.JPanel;
 9
10 public class Listener01 implements ActionListener {
11
12    private int premido = 0;
13    private JPanel p;
14    private JButton jb;
15    private JLabel lab;
16
17    public static void main(String[] args) {
18       Listener01 list = new Listener01();
19       list.jb = new JButton("Prima-me");
20       list.jb.addActionListener(list);
21       list.lab = new JLabel("O botão não foi premido");
22       list.p = new JPanel();
23       list.p.setLayout(new BoxLayout(list.p, BoxLayout.PAGE_AXIS));
24       list.jb.setAlignmentX(Component.CENTER_ALIGNMENT);
25       list.lab.setAlignmentX(Component.CENTER_ALIGNMENT);
26       list.p.add(list.jb);
27       list.p.add(list.lab);
28
29       JFrame02 f = new JFrame02("Exemplo gestão eventos 1", list.p, 300, 100);
30       f.setVisible(true);
31    }
32
33    @Override
34    public void actionPerformed(ActionEvent e) {
35       premido += 1;
```

```
36        if (premido == 1) {
37           this.lab.setText("Botao premido 1 vez");
38        } else {
39           this.lab.setText("Botao premido " + premido + " vezes ");
40        }
41     }
42 }
```

Note que na linha 33 aparece a declaração @Override que não é requerida pela linguagem Java, sendo inserida pelo NetBeans para nos alertar que se trata de um método pertencente ao uma Interface e portanto tem que ser reescrito.

Exercício: criar um formulário passo a passo

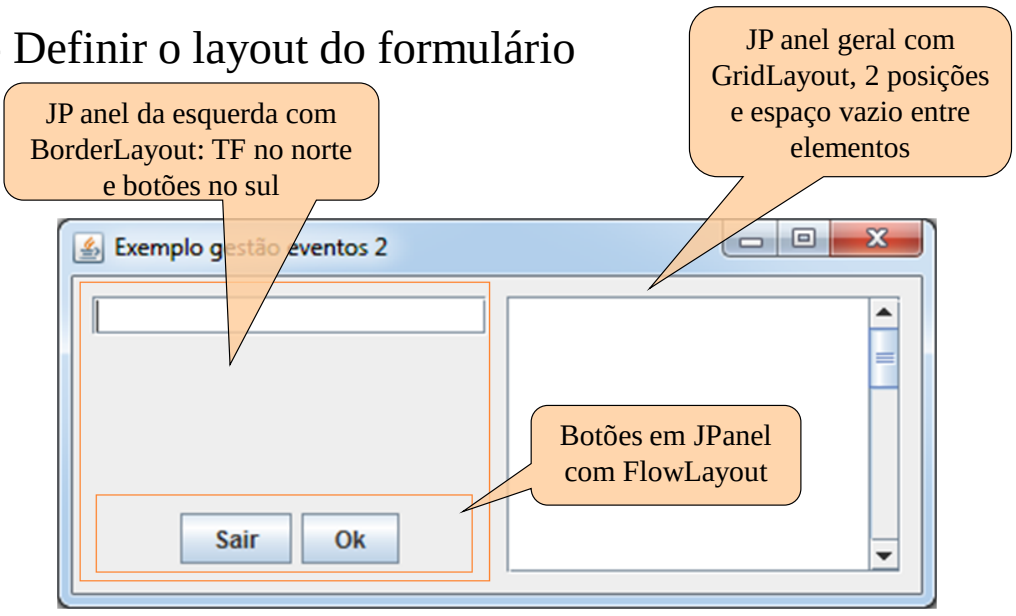
Exercício: criar formulário passo a passo #1/6

- Neste exemplo pretende-se passar texto introduzido num JTextField para uma JTextArea não editável;

- Passos a seguir:
  1. Definir o layout do formulário;
  2. Definir a funcionalidade do formulário;
  3. Identificar objetos com eventos relevantes ;
  4. Identificar o(s) listener(s) e ativar nos objetos;
  5. Definir método(s) do(s) listener(s)

Exercício: criar formulário passo a passo #2/6

1 - Definir o layout do formulário



Exercício: criar formulário passo a passo #3/6

2 - Definir a funcionalidade do formulário

- Ao ativar o botão OK o texto do JTextField passa para a JTextArea (usar espaço quando tem focus);
- Ao premir a tecla "Enter" sobre o JTextField o texto passa para a JTextArea;
- Ao ativar o botão "Sair" termina a execução do programa (usar espaço quando tem focus);
- A JFrame pode ser fechada;

Exercício: criar formulário passo a passo #4/6

3 - Identificar objetos com eventos relevantes

- Todos os objetos geram eventos;
- Vamos escolher apenas os objetos e os eventos que nos interessa tratar:
  - Botões Ok e Sair → ActionEvent;
  - TextField tf → ActionEvent;
  - JFrame → WindowEvent ;

Exercício: criar formulário passo a passo #5/6

4 - Identificar os listeners e adicionar aos objetos

- Ok, Sair → ActionEvent → ActionListener;
- tf → ActionEvent → ActionListener;

1. Na definição da classe adicionar **implements ActionListener**;
2. No construtor adicionar **addActionListener(list)** nos 3 objetos;
3. Importar as novas classes;

- **list** é usado em addActionListener para informar a JVM que os métodos da interface são definidos na própria classe;

Exercício: criar formulário passo a passo #6/6

5 - Definir método(s) do listener

- Para implementar o listener temos que definir o método actionPerformed() na própria classe;

```
public void actionPerformed (ActionEvent evt)  {
   if (evt.getSource() == sair) {
      ((JFrame) sair.getTopLevelAncestor()).dispose();
      //System.exit(0);
   } else {
      if ((evt.getSource() == ok) || (evt.getSource() == tf)) {
         ta.append(tf.getText() + "\n");
         tf.setText("");
         tf.requestFocusInWindow();
      }
   }
```

Terminar

Adicionar texto

Limpar

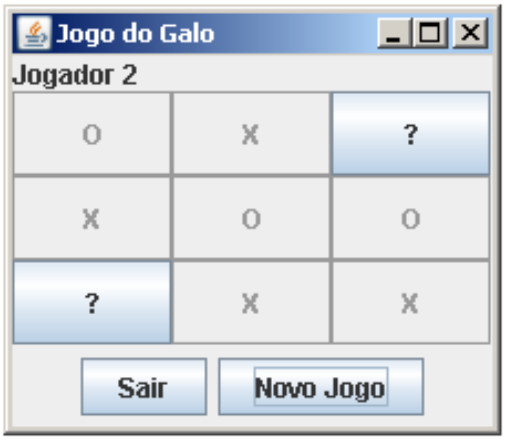
Recuperar focus

```
 1 package capitulo7;
 2
 3 import java.awt.BorderLayout;
 4 import java.awt.FlowLayout;
 5 import java.awt.GridLayout;
 6 import java.awt.event.ActionEvent;
 7 import java.awt.event.ActionListener;
 8 import javax.swing.*;
 9
10 public class Listener02 implements ActionListener{
11
12    private JButton ok;
13    private JButton sair;
14    private JTextField tf;
15    private JTextArea ta;
16
17    public static void main(String[] args) {
18       Listener02 list = new Listener02();
19       list.ok = new JButton("Ok");
20       list.ok.addActionListener(list);
21       list.sair = new JButton("Sair");
22       list.sair.addActionListener(list);
23       list.tf = new JTextField(30);
24       list.tf.addActionListener(list);
25       list.ta = new JTextArea(30,10);
26       list.ta.setEditable(false);
27       JScrollPane sp = new JScrollPane(list.ta);
28
29       JPanel pBotoes = new JPanel();
30       pBotoes.setLayout(new FlowLayout());
31       pBotoes.add(list.sair);
32       pBotoes.add(list.ok);
33
34       JPanel pEsquerda = new JPanel();
35       pEsquerda.setLayout(new BorderLayout());
36       pEsquerda.add(list.tf, BorderLayout.NORTH);
37       pEsquerda.add(pBotoes, BorderLayout.SOUTH);
38
39
40       JPanel geral = new JPanel();
41       geral.setLayout(new GridLayout(1,2,10,10));
42       geral.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));
43       geral.add(pEsquerda);
44       geral.add(sp);
45
46       JFrame02 f = new JFrame02("Exemplo gestão eventos 2", geral, 450, 200);
47       f.setVisible(true);
48    }
49
50    @Override
51    public void actionPerformed (ActionEvent evt)  {
52     if (evt.getSource() == sair) {
53         ((JFrame) sair.getTopLevelAncestor()).dispose();
54         //System.exit(0);
55     } else {
56         if ((evt.getSource() == ok) || (evt.getSource() == tf)) {
57             ta.append(tf.getText() + "\n");
58             tf.setText("");
59             tf.requestFocusInWindow();
60         }
61     }
62 }
63
64 }
```

Exercício: Jogo do Galo

Exercício: Jogo do Galo

- Aproveitando a classe desenvolvida com o tabuleiro do Jogo do Galo, adicione os mecanismos de gestão de eventos que permitam a duas pessoas jogar;



Vamos aproveitar a classe JogoDoGalo1 desenvolvida anteriormente e adicionar os gestores de eventos e métodos adequados para a funcionalidade pretendida:

```
 1 package capitulo7;
 2
 3 import java.awt.BorderLayout;
 4 import java.awt.FlowLayout;
 5 import java.awt.GridLayout;
 6 import java.awt.event.ActionEvent;
 7 import java.awt.event.ActionListener;
 8 import javax.swing.JButton;
 9 import javax.swing.JLabel;
10 import javax.swing.JPanel;
11
12 public class JogoDoGalo2 implements ActionListener {
13
14    private int jogador = 1;
15    private JLabel labJogador;
16    private JButton bSair;
17    private JButton bNovoJogo;
18    private JButton quadricula[];
19
20    public static void main(String[] args) {
21       JogoDoGalo2 jogo = new JogoDoGalo2();
22       jogo.labJogador = new JLabel("Jogador 1");
23       jogo.bSair = new JButton("Sair");
24       jogo.bSair.addActionListener(jogo);
25       jogo.bNovoJogo = new JButton("Novo jogo");
26       jogo.bNovoJogo.addActionListener(jogo);
27       JPanel botoesBaixo = new JPanel();
28       botoesBaixo.setLayout(new FlowLayout());
29       botoesBaixo.add(jogo.bSair);
30       botoesBaixo.add(jogo.bNovoJogo);
```

```
31
32        JPanel tabuleiro = new JPanel();
33        tabuleiro.setLayout(new GridLayout(3, 3));
34        jogo.quadricula = new JButton[9];
35        for (int i = 0; i < jogo.quadricula.length; i++) {
36           jogo.quadricula[i] = new JButton("?");
37           tabuleiro.add(jogo.quadricula[i]);
38           jogo.quadricula[i].addActionListener(jogo);
39        }
40
41        JPanel pGeral = new JPanel();
42        pGeral.setLayout(new BorderLayout());
43        pGeral.add(jogo.labJogador, BorderLayout.NORTH);
44        pGeral.add(tabuleiro, BorderLayout.CENTER);
45        pGeral.add(botoesBaixo, BorderLayout.SOUTH);
46
47        JFrame02 f = new JFrame02("Jogo do Galo", pGeral, 300, 200);
48        f.setVisible(true);
49     }
50
51     @Override
52     public void actionPerformed(ActionEvent e) {
53        if (e.getSource() == bSair) {
54           System.exit(0);
55        } else if (e.getSource() == bNovoJogo) {
56           limpaQuadro();
57        } else {
58           // é um botão do quadro
59           processaJogada(e);
60        }
61     }
62
63     private void processaJogada(ActionEvent e) {
64        JButton b = (JButton) e.getSource();
65        if (jogador == 1) {
66           b.setText("X");
67           jogador = 2;
68           labJogador.setText("Jogador 2");
69        } else {
70           b.setText("O");
71           jogador = 1;
72           labJogador.setText("Jogador 1");
73        }
74        b.setEnabled(false);
75     }
76
77     private void limpaQuadro() {
78        for (int i = 0; i < quadricula.length; i++) {
79           quadricula[i].setText("?");
80           quadricula[i].setEnabled(true);
81        }
82     }
83  }
```

Sumário

- JFrame, JDialog e JWindow;
- JPanel e JScrollPane
- Alguns componentes gráficos do package swing:
  - JLabel;
  - JButton;
  - JTextField;
  - JComboBox;
  - JCheckBox e JRadioButton;
- Distribuição dos componentes;
- Gestão de eventos;