Como construir aplicações gráficas e applets

Helder da Rocha
www.argonavis.com.br

AWT e Swing

- AWT ou Abstract Window Toolkit é o antigo conjunto de ferramentas para interfaces gráficas do Java
  - serve para oferecer infraestrutura mínima de interface gráfica (nivela por baixo)
  - componentes têm aparência dependente de plataforma
  - limitado em recursos devido a depender de suporte de cada plataforma para os componentes oferecidos
  - bugs e incompatibilidades entre plataformas
- JFC (Java Foundation Classes) oferece uma interface muito mais rica
  - Swing é o nome dado à coleção de componentes
  - É preciso importar java.awt e javax.swing

História do AWT

- Interface gráfica: componentes, layout, eventos
- Java 1.0
  - Interface que roda de forma medíocre em todas as plataformas ("Abominable" Window Toolkit)
  - Modelo de eventos arcaico
- Java 1.1
  - Melhora do modelo de eventos: por delegação usando design pattern Observer
- Java 1.2
  - JFC/Swing substitui totalmente componentes AWT
  - Mantém e estende a interface de eventos e layout

Java Foundation Classes

- Nativo desde Java SDK 1.2
- Biblioteca de componentes (Swing) pode ser usada em JDK 1.1.4 como extensão
  - Parte de drag&drop, cut&paste não incluída
- Swing: componentes leves, que não dependem de implementação nativa (veja Java Tutorial)
  - Uma das mais completas bibliotecas gráficas já criada
  - JavaBeans: ferramentas GUI conseguem gerar código legível e reutilizável
- Drag & drop, cut & paste, undo/redo, i18n, texto estilizado

JFC/Swing

- Coleção de objetos
- Botões, combo-boxes, frames internos (MDI), menus, árvores, tabelas, etc.
- Drag and drop, cut & paste, undo/redo, navegação via teclado, editor de texto, browser, componentes complexos
- Veja demonstração em
  - $JAVA_HOME/demo/jfc/SwingSet2/
  - java -jar SwingSet2.jar SwingSet2
- Detalhes: Java Tutorial: Swing trail (www.java.sun.com/tutorial)

Tipos de aplicações

- Há dois tipos de aplicações gráficas em Java
  - iniciadas via browser (applets)
  - iniciadas via sistema operacional
- Ambas capturam eventos do sistema e desenham-se sobre um contexto gráfico fornecido pelo sistema
- Applets são aplicações especiais que rodam a partir de um browser
  - Geralmente ocupam parte da janela do browser mas podem abrir janelas extras
  - Possuem restrições de segurança

java.awt.Component

- Raiz da hierarquia de componentes gráficos
  - Componentes Swing herdam de javax.swing.JComponent, que é "neto" de Component
- Há um Component por trás de tudo que pode ser pintado na tela
- Principais métodos:
  - void paint (java.awt.Graphics g)
  - void repaint()
  - void update(java.awt.Graphics g)

Componentes AWT

- *Há dois tipos importantes de componentes:*
- *1) descendentes diretos de java.awt.Component*
  - *"Apenas" componentes: descendentes da classe Component que não são descendentes de Container (todos os componentes da AWT)*
- *2. descendentes de java.awt.Container*
  - *Subclasse de java.awt.Component*
  - *São "recipientes." Podem conter outros componentes.*
  - *São descendentes da classe Container: Frame, Panel, Applet e JComponent (raiz da hierarquia dos componentes Swing)*

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*

Exemplo de JFrame

```
import java.awt.*;
import javax.swing.*;

public class Swinggy extends JFrame {

    public Swinggy(String nome) {
        super(nome);

        this.setSize(400,350);
        this.setVisible(true);
    }

    public static void main(String[] args) {
        new Swinggy("Swinggy 1.0");
    }

}
```

AWT thread

- *Thread que é responsável pela atualização do contexto gráfico*
  - *Chama update() (método de Component) e passa referência para o contexto gráfico como argumento sempre que for necessário redesenhá-lo.*
- *Método update(Graphics g)*
  - *Limpa a área a ser redesenhada (contexto gráfico)*
  - *Chama paint(g)*
- *Métodos update() e paint() nunca devem ser chamados diretamente a partir do thread principal*
  - *Use repaint(), que agenda uma chamada a paint() via AWT thread*
  - *Sobreponha update() se necessário*

java.awt.Graphics

- Representa o contexto gráfico de cada componente
- Passado pelo sistema quando chama update()
- Programador pode desenhar no componente usando referência recebida via paint() ao sobrepor o método:

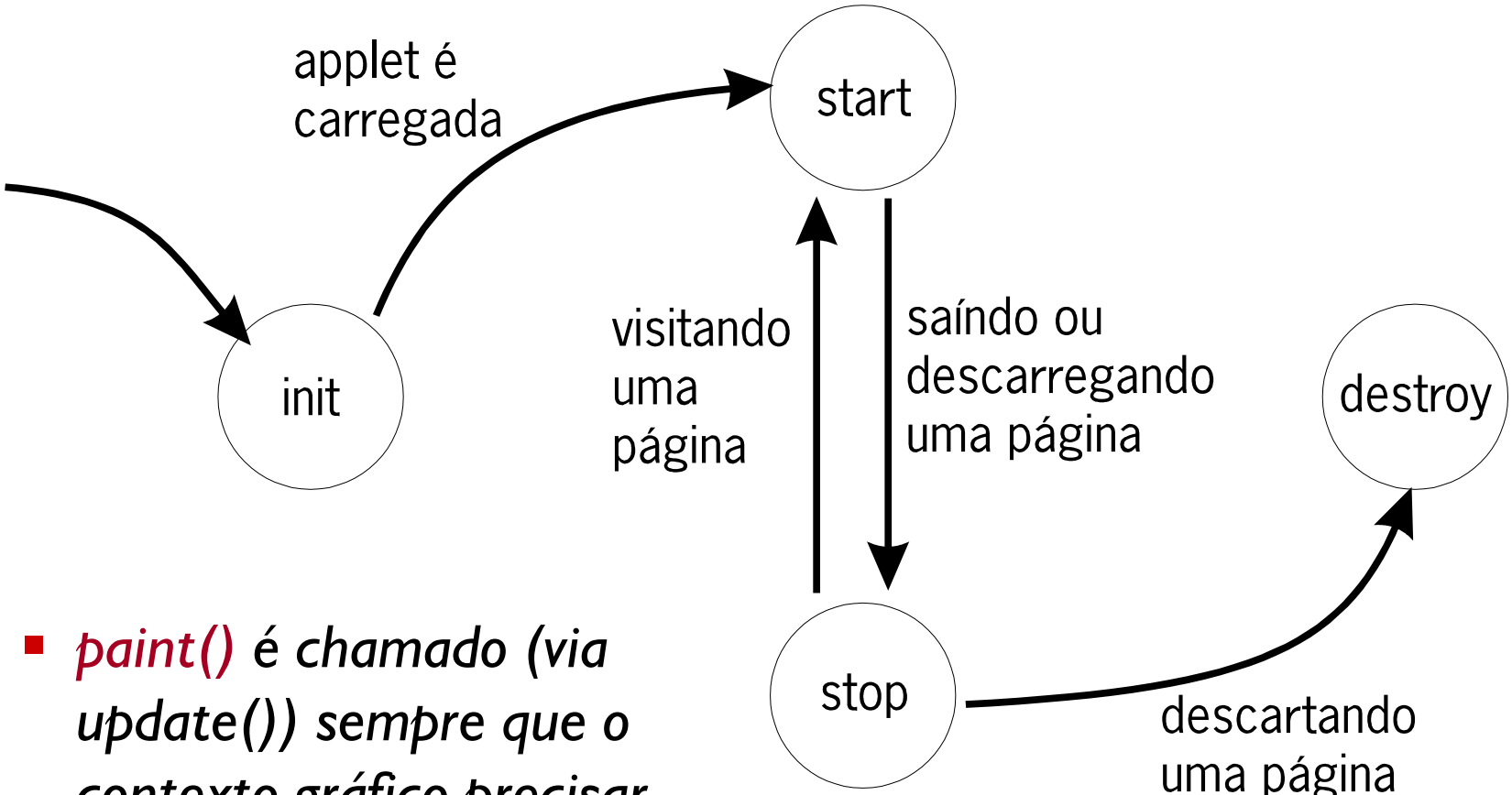
```
public void paint(Graphics g) {
      Graphics2D g2 = (Graphics2D) g;
      Shape s = new Ellipse2D.Double();
      g2.setColor(Color.red);
      g2.draw(s);
}
```

- Para definir o que será desenhado em determinado componente, sobreponha seu método paint()
  - Use Graphics2D! Mais recursos!

Applets

- *Aplicação gráfica que roda em browser*
  - *Toda a infraestrutura herdada da classe javax.swing.JApplet (ou java.applet.Applet)*
- *Métodos de JApplet, chamados automaticamente, devem ser sobrepostos*
  - *init() - inicialização dos componentes do applet*
  - *start() - o que fazer quando applet iniciar*
  - *stop() - o que fazer antes de applet parar*
  - *destroy() - o que fazer quando applet terminar*
  - *paint() - o que desenhar no contexto gráfico*

Ciclo de vida



- *paint() é chamado (via update()) sempre que o contexto gráfico precisar ser atualizado*

Como construir applets

- *Applet mínimo*

```
import javax.swing.*;
import java.awt.*;


/*
 * <applet code="HelloApplet" height="50"
 *         width="200"></applet>
 */
public class HelloApplet extends JApplet {
   public void init() {
      Container pane = this.getContentPane();
      JLabel msg = new JLabel("Hello Web");
      pane.add(msg);
   }
}
```

Comentário usado pelo appletviewer para exibir Applet:
> appletviewer HelloApplet.java

Como usar applets

- *Até Java 1.1: para incluir um applet na página Web usava-se*

```
<applet code="pacote.Classe"
        height="100" width="100">
    <param name="parametro1" value="valor">
    <param name="parametro2" value="valor">
</applet>
```

- *Java 2 (1.2 - ...) usa HTML 4.0 (que desaprovou <applet>)*

```
<object classid="XXX-XXX" ...> ... </object>
```

- *Para gerar <object> a partir de <applet> use o HTML Converter, distribuído com o SDK:*

```
java -jar $JAVA_HOME/lib/htmlconverter.jar -gui
```

Applet/Aplicação

- JApplets podem ser incluídos em JFrames e Applets podem ser incluídos em Frames
- Para criar um programa que roda tanto como applet como aplicação
  - Escreva o applet da forma convencional, implementando init(), start(), etc.
  - Crie um método main, e nele
    - crie um novo JFrame e uma instância do applet
    - adicione o applet no novo JFrame
    - torne o JFrame visível
    - chame init() e start() do applet

Exemplo

```
public class HelloApplet extends JApplet {
    public void init() {
        (...)
    }

    public static void main(String[] args) {
        HelloApplet ap = new HelloApplet();
        JFrame f = new JFrame("Applet");
        f.getContentPane().add(ap);
        f.setSize(200,50);
        ap.init();
        ap.start();
        f.setVisible(true);
    }

}
```

Restrições dos applets

- Há várias coisas que aplicações comuns podem e que um applet não pode fazer:
  - Não pode carregar bibliotecas ou definir métodos nativos
  - Não pode ler ou escrever arquivos na máquina cliente
  - Não pode fazer conexões de rede a não ser para a máquina de onde veio
  - Não pode iniciar a execução de nenhum programa no cliente
  - Não tem acesso à maior parte das propriedades do sistema
  - Janelas abertas sempre têm aviso de segurança
- Várias restrições podem ser flexibilizadas se o applet for assinado.

Applets: vantagens / desvantagens

- *Desvantagens*
  - *restrições*
  - *dependência de plug-in*
  - *tempo de download*
- *Vantagens*
  - *facilidade para realizar comunicação em rede*
  - *possibilidade de abrir janelas externas*
  - *capacidade de estender o browser em recursos de segurança, protocolos de rede, capacidade gráfica*
  - *capacidade de gerar e ler páginas Web com facilidade*
  - *capacidade de interagir com a página via JavaScript*

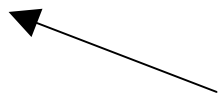
Fontes e Cores

- *Qualquer componente pode mudar a sua fonte e cor*
  - *A mudança afeta todos os componentes contidos no componente afetado*
- *Cores*
  - *instância da classe java.awt.Color*

```
componente.setBackground(new Color(255,0,0));
componente.setForeGround(Color.yellow);
```

- *Fontes*
  - *instância da classe java.awt.Font*

```
Font f = new Font("SansSerif", Font.BOLD, 24);
componente.setFont(f);
```



Há maneiras mais sofisticadas de lidar com fontes (esta é compatível com AWT)

Layouts

- *Há duas formas de acrescentar componentes em um container*
  - *Usar um algoritmo de posicionamento (layout) para dimensionar e posicionar os componentes (forma ideal e default)*
  - *Desligar o algoritmo de layout e posicionar e redimensionar os componentes diretamente*
- *Todo container tem um algoritmo de layout default*
  - *Frame e JFrame: BorderLayout (layout "geográfico")*
  - *outros Containers: FlowLayout (layout seqüencial)*

JFrame

- *Para acrescentar objetos em um JFrame ou JApplet, é preciso obter uma interface opaca chamada ContentPane*
  - *O ContentPane cobre a área útil do JFrame*
  - *O layout é definido no ContentPane*
  - *Objetos são adicionados no ContentPane*
- *Para obter o ContentPane estando dentro de um JFrame use*

```
Container pane = this.getContentPane();
```

- *Para definir um layout (diferente de BorderLayout)*

```
pane.setLayout(referência_para_layout);
```

FlowLayout, JButton, Icon

- *FlowLayout*
  - *É o layout mais simples*
  - *Dispõe os objetos um depois do outro como se fossem letras digitadas em um editor de texto*
  - *Estilo default é centralizado (pode ser alterado)*

```
this.setLayout(new FlowLayout());
```

- *JButton*
  - *Botões simples*
  - *Aceitam textos ou imagens (através da interface Icon)*

```
JButton b1 = new Button("texto");
JButton b2 = new Button("texto", icone);
```

- *Ícones de imagem*

```
Icon icone = new ImageIcon("caminho");
```

  - *Caminho deve ser relativo (de preferência) e usar sempre "/" como separador*

Componentes de texto

- *JTextField*
  - *campo de entrada de dados simples*
- *JPasswordField*
  - *campo para entrada de dados ocultos*
- *JTextArea*
  - *campo de entrada de texto multilinha*
- *JEditorPane*
  - *editor que entende HTML e RTF*
- *JTextPane*
  - *editor sofisticado com vários recursos*

- *Principais métodos*

getText()

recupera o texto contido no componente

setText(valor)

substitui o texto com outro

- *Veja documentação para mais detalhes*

Exemplo

```
public class Swinggy2 extends JFrame {

  public Swinggy2(String nome) {
     super(nome);

     Container ct = this.getContentPane();
     ct.setLayout(new FlowLayout());

     Icon icone = new ImageIcon("jet.gif");
     JButton b1 = new JButton("Sair");
     JButton b2 = new JButton("Viajar", icone);

     ct.add(b1);
     ct.add(b2);

     this.setSize(400,350);
     this.setVisible(true);
  }
}
```

Layout null

- *Algoritmos de layout podem ser combinados para obter qualquer configuração*
  - *Mais fáceis de manter e reutilizar*
  - *Layout em camadas e "orientado a objetos"*
  - *Controlam posicionamento e dimensão de componentes*
- *Algoritmos de layout podem ser criados implementando interface LayoutManager (e LayoutManager2)*
- *Para desligar layouts*

```
pane.setLayout(null);
```

- *Agora precisa definir posição e tamanho de cada componente*

```
componente.setBounds(x, y, larg, alt);
```

Exemplo

```
private JButton b1, b2, b3;
public Swinggy3(String nome) {
  Container ct = this.getContentPane();
  ct.setLayout(null);

  Icon pataverm = new ImageIcon("redpaw.gif");
  Icon pataverd = new ImageIcon("greenpaw.gif");
  Icon pataazul = new ImageIcon("bluepaw.gif");
  b1 = new JButton("Vermelha", pataverm);
  b2 = new JButton("Verde", pataverd);
  b3 = new JButton("Azul", pataazul);
  b1.setBounds(10,10,150,40);
  b2.setBounds(10,60,150,40);
  b3.setBounds(10,110,150,40);
  ct.add(b1);
  ct.add(b2);
  ct.add(b3);
}
```

Aplicação fica mais simples com vetores

```
private JButton[] b;
private String[] txt = {"Roda", "Para", "Pausa", "Sai",
                        "Olha", "Urgh!"};
private String[] img = { "greenpaw.gif", "redpaw.gif",
                         "bluepaw.gif",  "jet.gif",
                         "eye.gif",      "barata.gif"};
public Swinggy4(String nome) {
   Container ct = this.getContentPane();
   ct.setLayout(null);
   b = new JButton[txt.length];
   for (int i = 0; i < b.length; i++) {
     b[i] = new JButton(txt[i],
              new ImageIcon(img[i]));
     b[i].setBounds(10,10+(i*50),150,40);
     ct.add(b[i]);
   } // (...)
}
```

Outros layouts

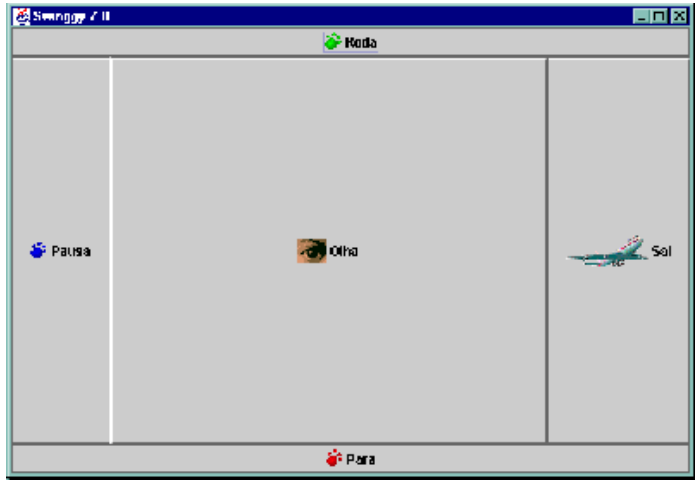
- *GridLayout(linhas, colunas)*
  - *layout que posiciona os elementos como elementos de uma tabela*
- *BorderLayout*
  - *layout que posiciona elementos em quatro posições "cardeais" e no centro*
  - *Norte e Sul têm prioridade sobre Leste e Oeste que têm prioridade sobre Centro*
  - *Constantes BorderLayout.CENTER, BorderLayout.WEST, etc.*
- *BoxLayout e GridBagLayout*
  - *permitem layouts sofisticados com amplo controle*

Exemplo com BorderLayout

```
(...)

ct.setLayout(new BorderLayout());
b = new JButton[txt.length];
String[] pos = {BorderLayout.NORTH,
                BorderLayout.SOUTH,
                BorderLayout.WEST,
                BorderLayout.EAST,
                BorderLayout.CENTER};
for (int i=0; i < Math.min(b.length, pos.length); i++) {
   b[i] = new JButton(txt[i], new ImageIcon(img[i]));
        ct.add(pos[i], b[i]);
}

(...)
```



Veja restante do código (vetores b, txt e img) em slides anteriores

Exemplo com GridLayout

```
(...)
ct.setLayout(new GridLayout(3, 2));

b = new JButton[txt.length];
for (int i = 0; i < b.length; i++) {
   b[i] = new JButton(txt[i],
                      new ImageIcon(img[i]));
       ct.add(b[i]);
}
(...)
```

Redimensione a janela e veja o resultado

Mesmo exemplo com FlowLayout

```
(...)
public Swinggy5(String nome) {
   super(nome);
   Container ct = this.getContentPane();
   ct.setLayout(new FlowLayout());
   b = new JButton[txt.length];
   for (int i = 0; i < b.length; i++) {
      b[i] = new JButton(txt[i],
                         new ImageIcon(img[i]));
      ct.add(b[i]);
   }
   this.setSize(400,350);
   this.setVisible(true);
} (...)
```

Redimensione a janela e veja o resultado

Combinação de Layouts

- *Componentes podem ser combinados em recipientes (como JPanel) para serem tratados como um conjunto*
  - ```
    JPanel p = new JPanel();
    p.setLayout(layout do JPanel);
    p.add(comp1);
    p.add(comp2);
    pane.add(BorderLayout.EAST, p);
    ```
- *Possibilita a criação de layouts complexos que consistem de várias camadas*
  - *Cada JPanel é uma camada*
    ```

Exemplo (1/2)

```
import javax.swing.*;
import java.awt.*;

public class Swinggy8 extends JFrame {

  public Swinggy8(String nome) {
    super(nome);

    Container framePane = this.getContentPane();
    framePane.setLayout(new BorderLayout());

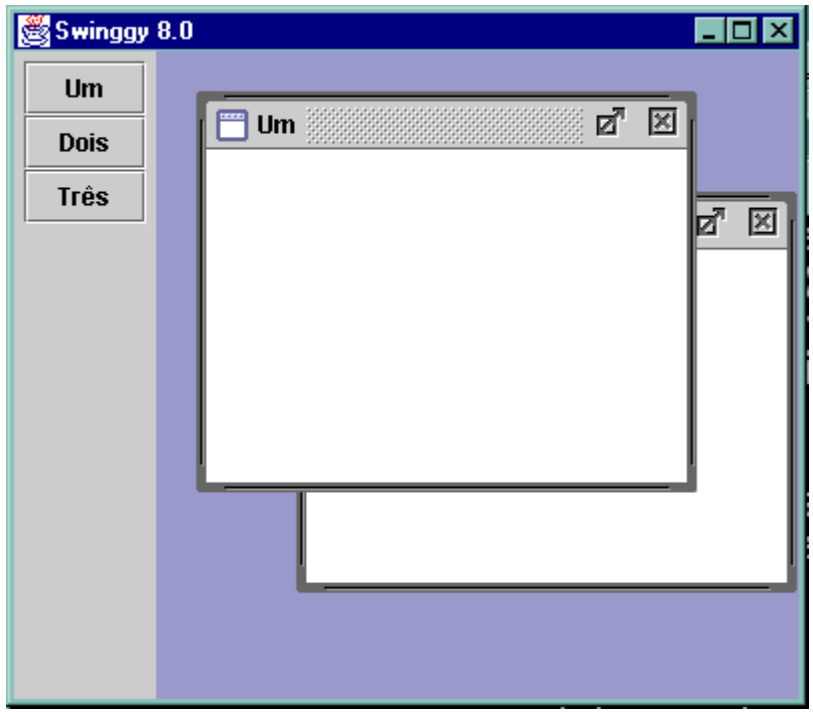
    JPanel botoes = new JPanel();
    botoes.setBackground(Color.yellow);
    botoes.setLayout(new GridLayout(3,1));
    botoes.add(new JButton("Um"));
    botoes.add(new JButton("Dois"));
    botoes.add(new JButton("Três"));
    JPanel lateral = new JPanel();
    lateral.add(botoes);
```

Exemplo (2/2)

```
    JInternalFrame if1 =
     new JInternalFrame("Um", true, true, true);
    JInternalFrame if2 =
     new JInternalFrame("Dois", true, true, true);
    if1.getContentPane().add(new JEditorPane());
    if2.getContentPane().add(new JEditorPane());
    if1.setBounds(20,20, 250,200);
    if2.setBounds(70,70, 250,200);
    if1.setVisible(true);
    if2.setVisible(true);

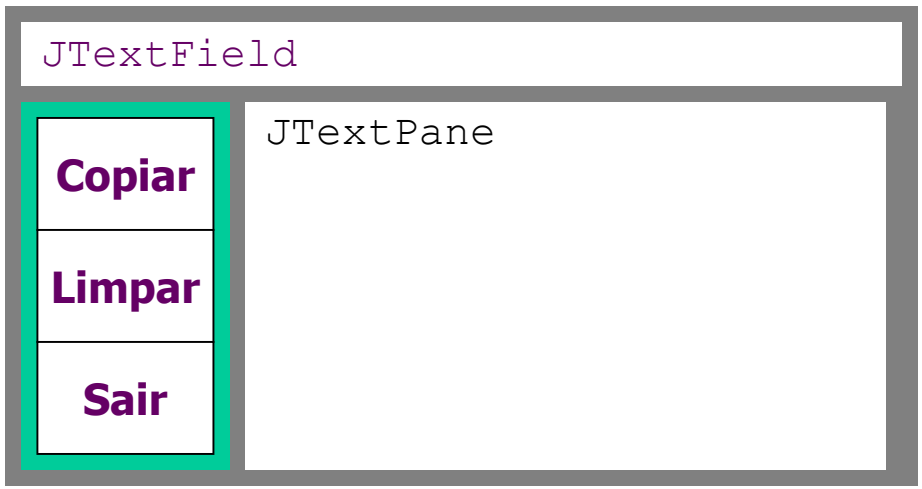
    JDesktopPane dtp = new JDesktopPane();
    dtp.add(if1); dtp.add(if2);
    framePane.add(BorderLayout.CENTER, dtp);
    framePane.add(BorderLayout.WEST, lateral);
    this.setSize(400,350);
    this.setVisible(true);
  }
}
```

Exemplo: resultado



Exercício

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



Use BorderLayout para distribuir os componentes JTextField, JEditorPane e JPanel

Use GridLayout para distribuir os botões

- *2. Experimente usar outros componentes e outros layouts (veja SwingSet demo)*

Eventos

- *Eventos em Java são objetos*
  - *Subclasses de 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 evento, método de ouvinte é chamado e evento é passado como argumento*

Eventos da Interface Gráfica

- Descendentes de java.awt.event.AWTEvent
- Divididos em categorias (java.awt.event)
  - ActionEvent (fonte: componentes de ação)
  - MouseEvent (fonte: componentes em geral 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 em geral afetados pelo teclado)
  - ...

Listeners

- *Cada evento tem uma interface Listener correspondente que possui métodos padrão para tratá-los*
  - *ActionEvent: ActionListener*
  - *MouseEvent: MouseListener e MouseMotionListener*
  - *ItemEvent: ItemListener*
  - *AdjustmentEvent: AdjustmentListener*
  - *TextEvent: TextListener*
  - *WindowEvent: WindowListener*
  - *FocusEvent: FocusListener*
  - *KeyEvent: KeyListener*
  - *XXXEvent: XXXListener*

Como ligar a fonte ao listener

- *Na ocorrência de um evento, em uma fonte, todos os listeners registrados serão notificados*
  - *É preciso antes cadastrar os listeners na fonte*
  - *fonte.add<Listener>(referência_para_listener);*
- *Exemplo:*
  - ```
    JButton button = new JButton("Fonte");
    ActionListener ouvinte1 = new OuvinteDoBotao();
    MouseListener ouvinte2 = new OuvinteDeCliques();
    button.addActionListener(ouvinte1);
    button.addMouseListener(ouvinte2);
    ```
- *É preciso implementar os listeners para cada tipo de evento tratado*
    ```

Como implementar um listener

- *Crie uma nova classe que declare implementar o(s) listener(s) desejado(s)*
  - `public class MeuListener implements ActionListener, ItemListener { ...}`
- *Implemente cada um dos métodos da(s) interface(s)*
  - `public void actionPerformed(ActionEvent e) { ...}`

    `public void itemStateChanged(ItemEvent e) {...}`
- *Veja a documentação sobre o listener usado e o evento correspondente (para saber como obter suas informações)*

Quais os listeners, métodos, eventos?

- *Consulte a documentação (veja também livro-texto)*
  - `java.awt.event.*`
- *Veja em cada listener os métodos que você deve implementar*
- *Veja em cada evento os métodos que você pode chamar dentro do listener para obter as informações desejadas (por exemplo textos, coordenadas, teclas apertadas, etc.)*
- *Veja em cada componente-fonte os métodos que você pode chamar para obter informações sobre o componente:*

```
Object fonte = evento.getSource();
if (fonte instanceof JButton)
   JButton b = (JButton) fonte;
   String label = b.getLabel();
```

Exemplo

```
import java.awt.event.*;

public class Swinggy9 extends JFrame {
   public Swinggy9(String nome) {
      (...)
        JButton b1 = new JButton("Sair");
        JButton b2 = new JButton("Viajar", icone);
        JTextField tf = new JTextField(10);
        b1.setActionCommand("Saindo");
        b2.setActionCommand("Viajando");

        ActionListener listener = new Eco();
        b1.setActionListener(listener);
        b2.setActionListener(listener);
      (...)
   }

   // Classe interna!!
   class Eco implements ActionListener {
       public void actionPerformed(ActionEvent e) {
          tf.setText(e.getActionCommand());
       }
   }
}
```

Exercício

- 3. Implemente os eventos para a aplicação
  - Copiar deve acrescentar o texto do JTextField no JEditorPane e limpar o JTextField
  - Limpar deve limpar o JTextField
  - Sair deve sair do programa
- 4. Implemente os botões como itens do menu "Operações"
  - use JMenuBar, JMenu e JMenuItem
- 5. Implemente um JToggleButton "desenhar/escrever" que troque o JTextPane por um JCanvas e permita rabiscar com o mouse (use MouseEvent)

Exercício

- *6. Implemente uma interface gráfica para a aplicação da biblioteca (capítulo 7).*
  - *Use o JTabbedPane para criar diferentes painéis: um para entrada de Agentes (autores, editores), outro para entrada de Publicações (livros, revistas, artigos) e outro para buscas.*
  - *Use um combo-box para selecionar cada tio de agente ou publicação*
  - *Desabilite campos não utilizados.*
  - *Implemente os eventos chamando os métodos da fachada (Biblioteca)*

Demo do SwingSet

- *Para ver uma demonstração dos componentes Swing que existem no Java rode a aplicação SwingSet*
  - *Mude para o diretório*

    ```
    $JAVA_HOME/demo/jfc/SwingSet2
    ```

    e rode

    ```
    java -classpath SwingSet2.jar SwingSet2
    ```
  - *Explore os exemplos e veja o código fonte*