DIN Departamento de Informática

UEM Universidade Estadual de Maringá

Interface Gráfica: Introdução

Lilian Passos Scatalon
lpscatalon2@uem.br

Interface com o Usuário (*User Interface - UI*)

A interface é a parte de um sistema com que o usuário mantém contato

Interface baseada em texto: envolve entrada/saída no modo texto por meio de um terminal

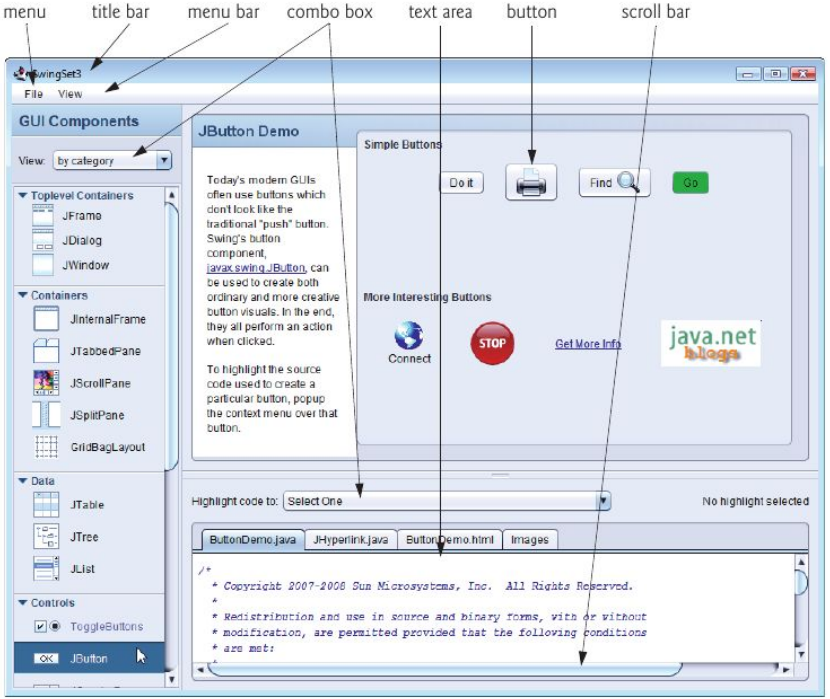
Interface gráfica: o sistema dispõe de elementos gráficos (botões, caixas de texto etc) para a interação com o usuário

Interface Gráfica do Usuário

GUI - *Graphical User Interface*

Apresenta um mecanismo *user-friendly* para interagir com o sistema

É construída com componentes GUI



Interface Gráfica em Java

Abstract Window Toolkit (AWT): API dependente de plataforma (SO), gera componentes GUI nativos.

Swing: Baseado em AWT, mas apresenta componentes GUI próprios em Java. Apresenta portabilidade, a mesma aparência ao longo de diferentes plataformas e mais funcionalidades.

JavaFX: API Java mais recente para interface gráfica e multimídia. Permite customizações com CSS. A partir do JDK 11, é usada como uma biblioteca à parte.

API Swing

Robusta, ainda muito utilizada, inclusa no JDK desde Java SE 1.2

A maioria dos pacotes fica em `javax.swing`

https://docs.oracle.com/javase/tutorial/figures/uiswing/learn/1helloworldswing.png

Conceitos básicos

Janela (*window*): Área da tela com a interface gráfica de um sistema. Container de componentes GUI. Exemplos: frame, dialog box, applet.

Componentes: Elementos gráficos contidos em uma janela. Também chamados de controles ou widgets. Exemplos: botão, caixa de texto.

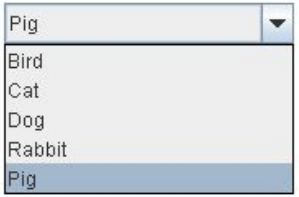
Container: um agrupamento lógico para armazenar componentes. Exemplo: painel

Basic Controls

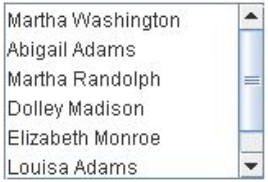
Simple components that are used primarily to get input from the user; they may also show simple state.

JButton

JCheckBox



JComboBox



JList

JMenu

JRadioButton

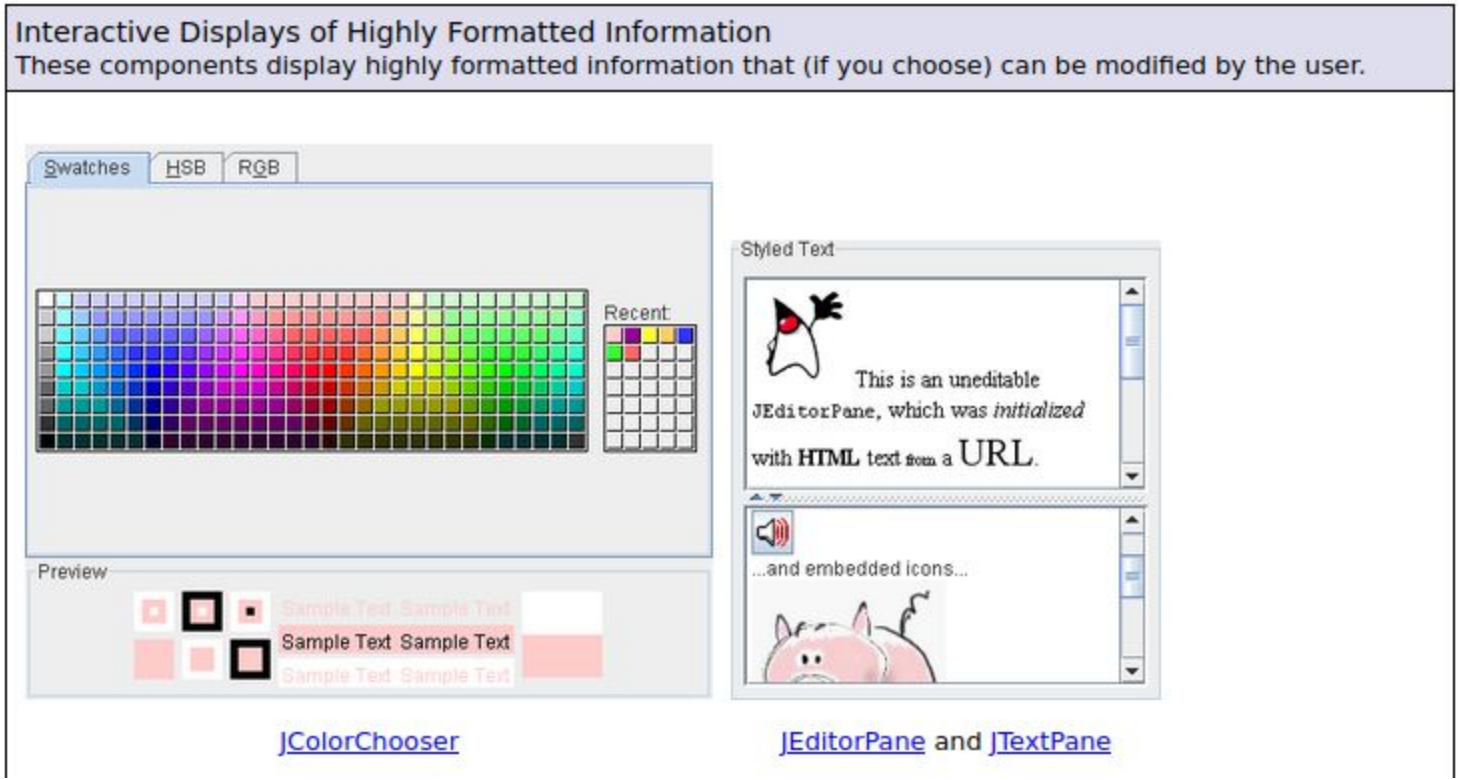
JSlider

JSpinner

JTextField

JPasswordField

https://web.mit.edu/6.005/www/sp14/psets/ps4/java-6-tutorial/components.html

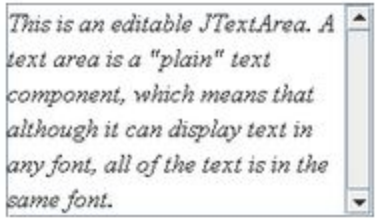


https://web.mit.edu/6.005/www/sp14/psets/ps4/java-6-tutorial/components.html

JFileChooser

Host	User	Password	Last Modified
Biocca Games	Freddy	!#asf6Awwzb	Mar 16, 2006
zabble	ichabod	Tazb!34$fZ	Mar 6, 2006
Sun Developer	fraz@hotmail.co...	AasW541!fbZ	Feb 22, 2006
Heirloom Seeds	shams@gmail....	bkz[ADF78!	Jul 29, 2005
Pacific Zoo Shop	seal@hotmail.c...	vbAf124%z	Feb 22, 2006

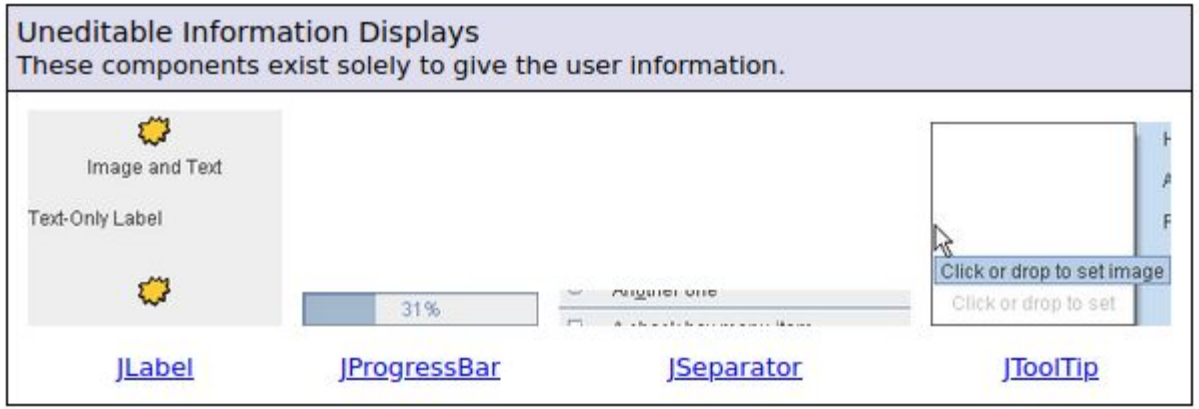
JTable



JTextArea

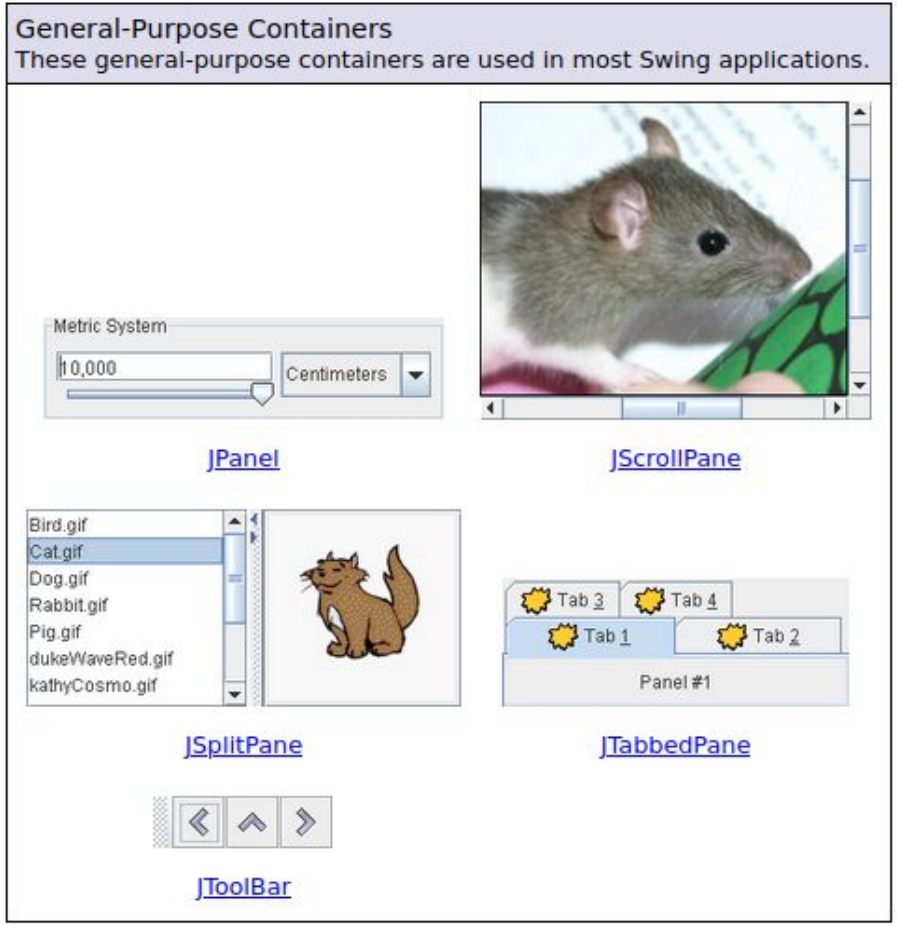
JTree

https://web.mit.edu/6.005/www/sp14/psets/ps4/java-6-tutorial/components.html



https://web.mit.edu/6.005/www/sp14/psets/ps4/java-6-tutorial/components.html

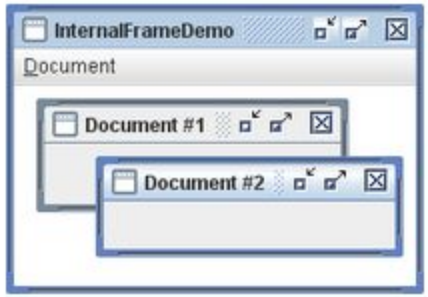
https://web.mit.edu/6.005/www/sp14/psets/ps4/java-6-tutorial/components.html

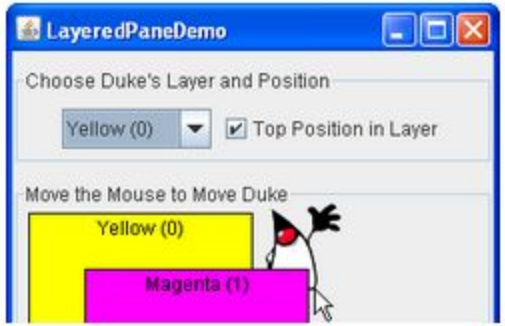


https://web.mit.edu/6.005/www/sp14/psets/ps4/java-6-tutorial/components.html

Special-Purpose Containers

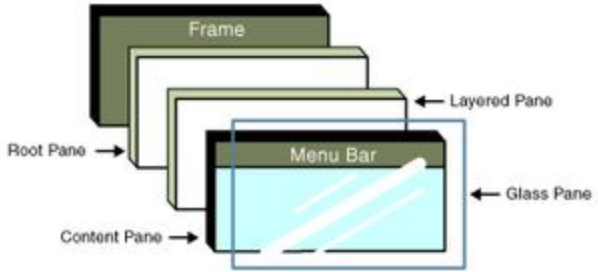
These special-purpose containers play specific roles in the UI.





JInternalFrame

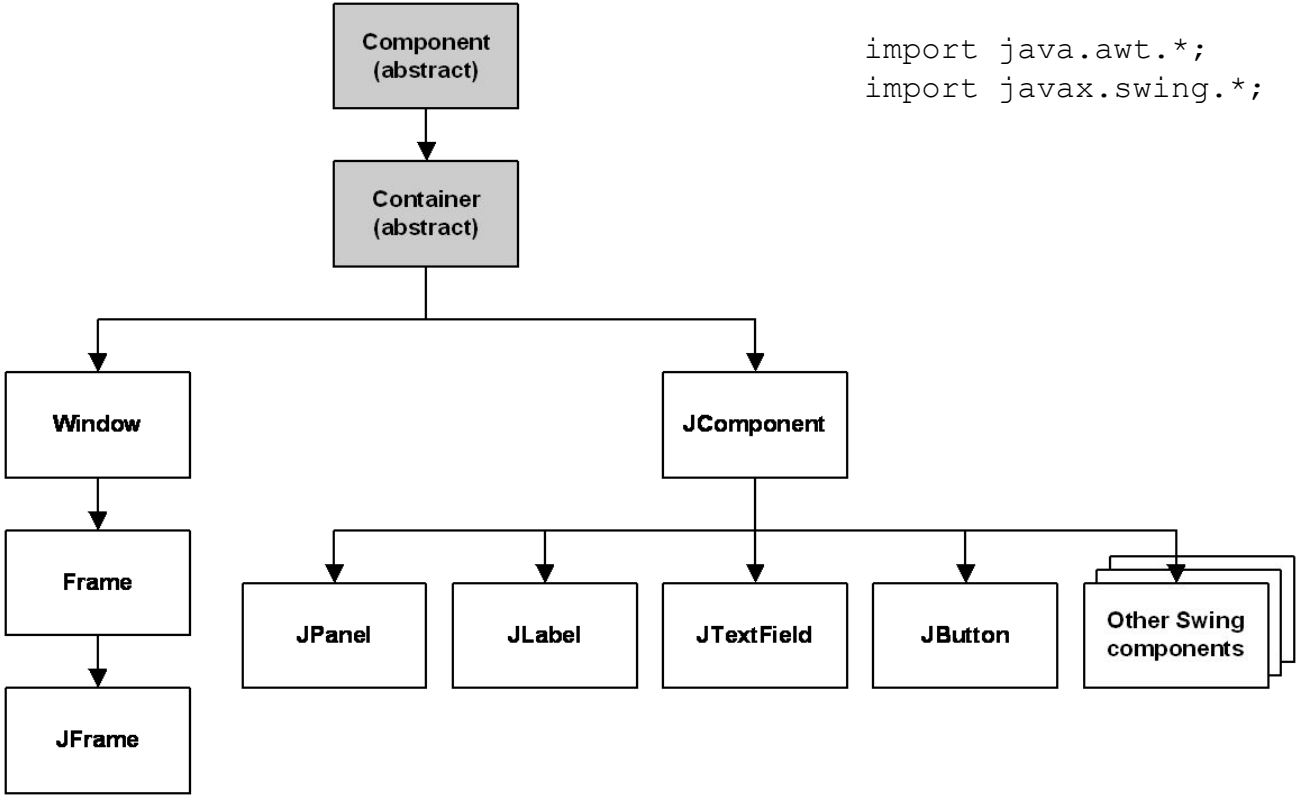
JLayeredPane



Root pane

https://web.mit.edu/6.005/www/sp14/psets/ps4/java-6-tutorial/components.html

Hierarquia Swing




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

https://voyager.deanza.edu/~hso/cis35a/lecture/java15/images/Picture2.png

Propriedades dos componentes

Cada componente tem métodos:

- de acesso (get ou is)
- modificadores (set)

Exemplo: getSize(), setSize(), getVisible(), isVisible()

Propriedade	Tipo	Descrição
background	Color	cor de plano de fundo
border	Border	linha de contorno em volta do componente
enabled	boolean	se está habilitado (é possível interagir com o comp.)
focusable	boolean	se pode receber o foco (é possível digitar texto)
font	Font	fonte usada no texto do componente
foreground	Color	cor de plano de frente
height, width	int	tamanho atual em pixels
visible	boolean	se pode ser visto
tooltip text	String	texto mostrado quando o mouse paira sobre o comp.
size, minimum / maximum / preferred size	Dimension	tamanho, limite de tamanho, tamanho desejado

JFrame

Uma janela gráfica para acomodar outros componentes

- public JFrame()
  public JFrame(String title)
    - Chamar setVisible(true) para fazer o frame aparecer depois de criá-lo

- public void add(Component comp)
  Insere o componente dentro do frame

JFrame

- public void setDefaultCloseOperation(int op)
  Define a ação a ser realizada quando o frame é fechado
  - Argumento comum: JFrame.EXIT_ON_CLOSE

- public void setSize(int width, int height)

- public void pack()
  Redimensiona o frame para acomodar seus componentes

JButton

Uma região clicável que dispara ações

- public JButton(String text)

  Cria um novo botão com o texto dado
- public String getText()

  Retorna o texto mostrado no botão
- public void setText(String text)

  Configura o texto do botão

```
import java.awt.*;    // Onde está o outro botão?
import javax.swing.*;

public class GuiExample1 {
    public static void main(String[] args) {
    JFrame frame = new JFrame();
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(new Dimension(300, 100));
    frame.setTitle("A frame");

    JButton button1 = new JButton();
    button1.setText("I'm a button.");
    button1.setBackground(Color.BLUE);
    frame.add(button1);

    JButton button2 = new JButton();
    button2.setText("Click me!");
    button2.setBackground(Color.RED);
    frame.add(button2);

    frame.setVisible(true);
    }
}
```

Tamanho e posicionamento

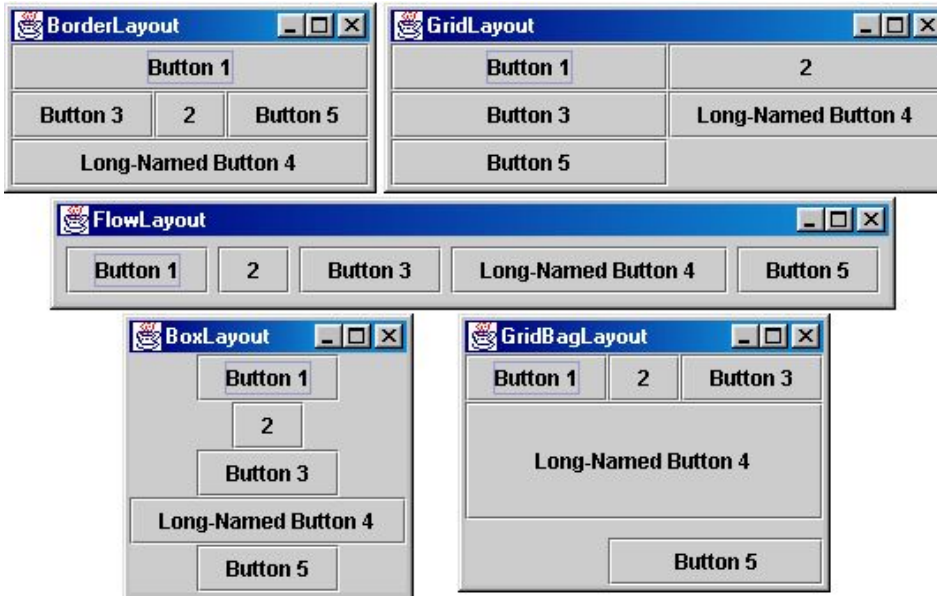
- **Posicionamento absoluto** (C++, C#, outras):
  Coordenadas exatas de cada componente
  - "Coloque esse botão em x=15, y=75, com 70x31 px de tamanho"

- **Gerenciadores de layout** (Java):
  Os objetos decidem onde posicionar cada elemento com base em regras gerais
  - "Coloque esses quatro botões em um grid 2x2 e coloque essas caixas de texto em um fluxo horizontal na parte sul do frame"

Containers e layout

Posiciona-se os componentes em um *container*; adiciona-se o container a um frame

- **container**: um objeto que armazena componentes e gerencia as posições, tamanhos e comportamento



JFrame como container

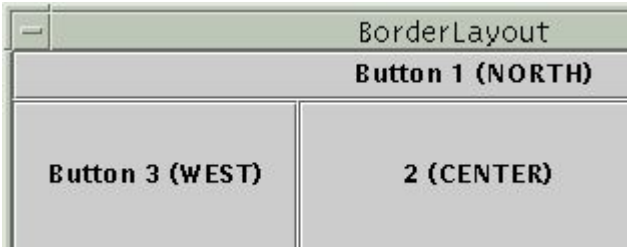
JFrame é um container. Containers têm os seguintes métodos:

- public void add(Component comp)
  public void add(Component comp, Object info)
  Adiciona um componente ao container, com informação de onde colocá-lo
- public void remove(Component comp)
- public void setLayout(LayoutManager mgr)
  Usa o gerenciador de layout dado para posicionar componentes

Preferred size

Componentes Swing têm um certo tamanho em que "preferem" estar: o tamanho suficiente para caber seus conteúdos (texto, ícones etc).

- Alguns gerenciadores de layout (p.ex. FlowLayout) adotam o preferred size dos componentes
- Outros (p.ex. BorderLayout, GridLayout) desconsideram-no e usam algum outro esquema para dimensionar os componentes



FlowLayout

public FlowLayout()

Trata o container como um "parágrafo" sendo preenchido da esquerda para a direita, de cima para baixo

Os componentes são posicionados na ordem em que são adicionados

É o layout padrão para containers (exceto JFrame)


```
myFrame.setLayout(new FlowLayout());
myFrame.add(new JButton("Button 1"));
...
```

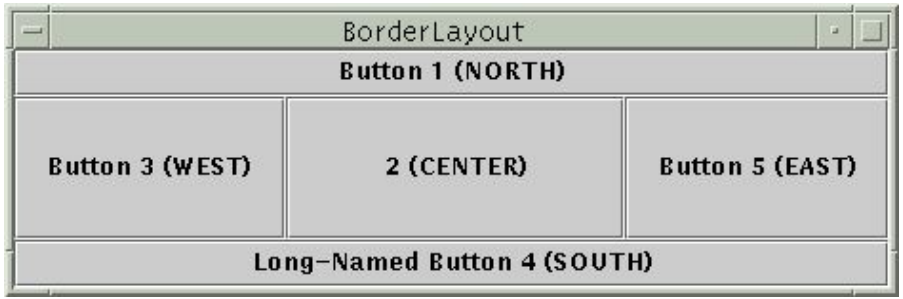
BorderLayout

public BorderLayout()

Divide o container em quatro regiões:

- as regiões NORTH e SOUTH se expandem na horizontal
- as regiões WEST e EAST se expandem na vertical
- a região CENTER usa o restante do espaço não ocupado

```
myFrame.setLayout(new BorderLayout());
myFrame.add(new JButton("Button 1"),
            BorderLayout.NORTH);
...
```



É o layout padrão para o JFrame

GridLayout

public GridLayout(int rows, int columns)

Trata o container como um grid de linhas e colunas com igual tamanho

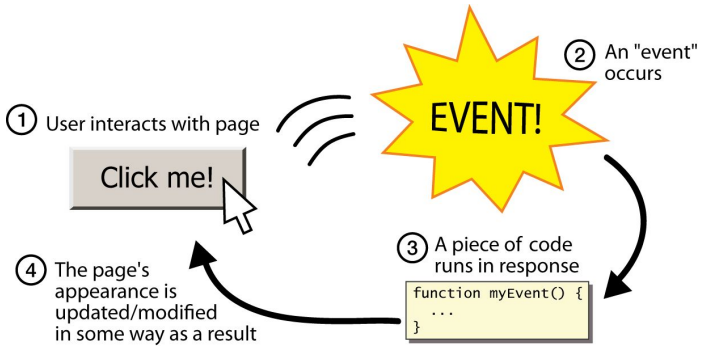
Componentes recebem o mesmo tamanho horizontal e vertical, desconsiderando seus tamanhos preferidos

Eventos gráficos

Evento: um objeto que representa uma interação do usuário com um componente GUI

Listener: um objeto que espera por eventos e responde a eles

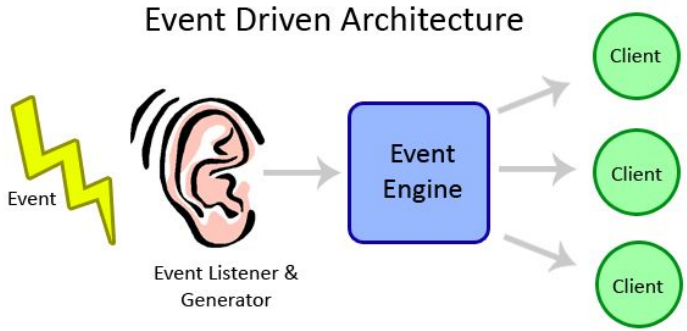
- Para manipular um evento, vincule um listener a um componente
- O listener será notificado quando um evento ocorrer



Programação orientada a eventos

O fluxo de execução geral do programa é determinado por eventos

- O programa carrega e espera por eventos de entrada do usuário
- Conforme cada evento acontece, o programa executa o respectivo código de resposta
- O fluxo de execução é determinado pela sequência de eventos que ocorrem, não é uma ordem pré-determinada



Hierarquia de eventos

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

- EventObject
  - AWTEvent (AWT)
  - ActionEvent
  - TextEvent
  - ComponentEvent
    - FocusEvent
    - WindowEvent
    - InputEvent
      - KeyEvent
      - MouseEvent

- EventListener
  - AWTEventListener
  - ActionListener
  - TextListener
  - ComponentListener
  - FocusListener
  - WindowListener

  - KeyListener
  - MouseListener

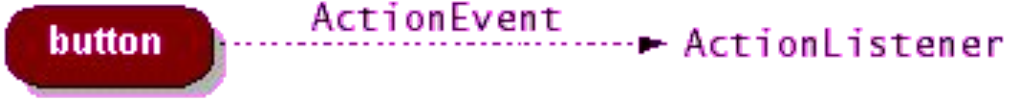
ActionEvent

Evento de ação: indicam que uma ação ocorreu em um componente GUI

É o tipo de evento mais comum, causado por:

- cliques em botões ou menus,
- marcar/desmarcar caixa de seleção,
- pressionar a tecla enter em um campo de texto etc

Manipulado por objetos que implementam a interface ActionListener



Implementando um listener

```
public class nomeClasse implements ActionListener {
    public void actionPerformed(ActionEvent event) {
    código para manipular o evento;
    }
}
```

JButton e outros componentes GUI têm o seguinte método:

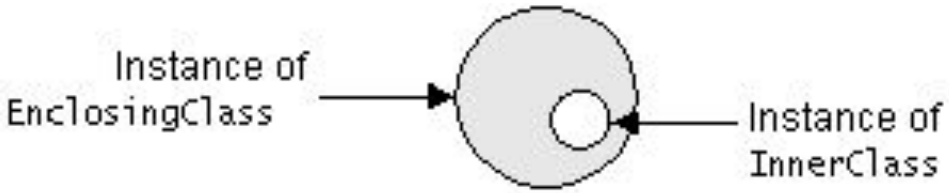
```
public void addActionListener(ActionListener al)
```

Vincula o listener dado como parâmetro para ser notificado de cliques e eventos que ocorram no componente

Classes aninhadas

Uma classe definida dentro de outra classe
A classe aninhada fica oculta de código externo (encapsulamento)
Objetos aninhados podem acessar/modificar os atributos do objeto "externo"

Os listeners de evento são frequentemente definidos como classes aninhadas



Sintaxe para classe aninhada

```
// classe externa
public class nomeClasseExterna {
    ...
    // classe aninhada
    private class nomeClasseAninhada {
        ...
    }
}
```

Apenas a classe externa pode instanciar objetos da classe aninhada
Cada objeto aninhado é associado com o objeto externo que o criou, de modo que possa acessar/modificar os métodos/atributos do objeto externo

- Se necessário, é possível se referir ao objeto externo como: **`nomeClasseExterna.`**`this`

Exemplo de evento GUI

```
public class MyGUI {
    private JFrame frame;
    private JButton repete;
    private JTextField caixaTexto;

    public MyGUI() {
        ...
        repete.addActionListener(new StutterListener());
    }
    ...

    // Quando um botão recebe um clique, o texto é duplicado
    private class RepeteListener implements ActionListener {
        public void actionPerformed(ActionEvent event) {
            String texto = caixaTexto.getText();
            caixaTexto.setText(texto + texto);
        }
    }
}
```

Bibliografia

M. Stepp. **Event-driven Programming and GUIs with Swing/AWT**. Disponível em: http://www.cs.washington.edu/331/

P. Deitel e H. Deitel. Swing GUI Components (online chapters). **Java How to Program: Early Objects**. 11th Edition, 2017.

Oracle. Creating a GUI with Swing. **The Java Tutorials**. Disponível em: https://docs.oracle.com/javase/tutorial/uiswing/index.html