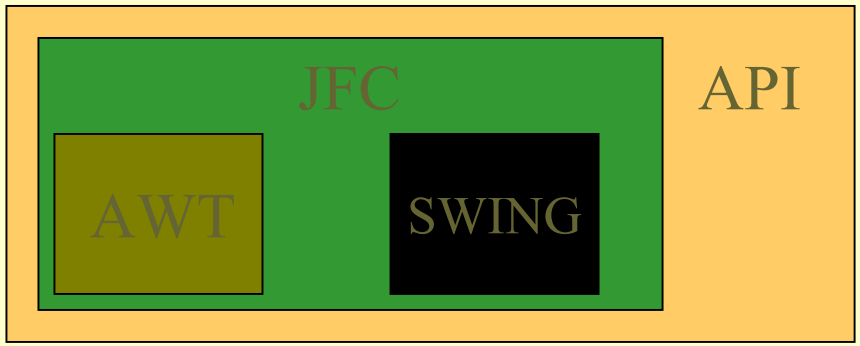
Interface Gráfica

Prof. Marcelo Cohen

1. Visão geral da interface gráfica em Java

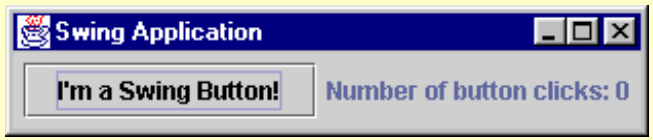
- A API (*Application Programming Interface*) para utilizar recursos gráficos é separada em dois componentes principais:
  - **AWT** (*Abstract Windowing Toolkit*): contém o conjunto de classes necessárias para a construção de programas com interfaces de janelas.
  - **SWING**: contém o conjunto de classes complementares para a construção de interfaces avançadas (somente em Java 1.2)



- JFC : *Java Foundation Classes*

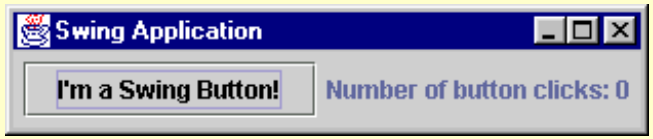
- Uma interface gráfica em Java é baseada em dois elementos:
  - *containers*: servem para agrupar e exibir outros componentes
  - *components*: botões, labels, scrollbars, etc.
- Dessa forma, todo programa que ofereça uma interface vai possuir pelo menos um *container*, que pode ser:
  - **JFrame -** janela principal do programa
  - **JDialog -** janela para diálogos
  - **JApplet -** janela para Applets
- Para fins de exemplificação, iremos utilizar um JFrame.

- Exemplo: uma janela simples



- Possui os seguintes elementos:
  - **JFrame:** armazena os demais componentes
  - JPanel: painel, serve para facilitar o posicionamento do botão e do label
  - JButton: o botão "I'm a Swing button!"
  - JLabel: o texto "Number of button clicks: 0"

- JFrame's são top-level containers: sempre estão presentes
- JPanel's são intermediate containers: podem estar ou não presentes (mas geralmente estão)
- Jbutton e Jlabel são componentes atômicos: não podem ser usados para conter outros e normalmente respondem ao usuário



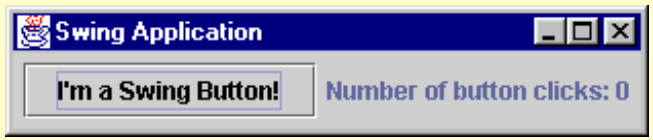

```
import javax.swing.*;

class SimpleFrame
{
  public static void main(String args[])
  {
    JFrame frame = new JFrame("Swing Application");
    JButton but = new JButton("I'm a Swing Button!");
    JLabel texto = new JLabel("Number of button clicks: 0");
    JPanel painel = new JPanel();

    painel.add(but);
    painel.add(texto);

    frame.getContentPane().add(painel);
    frame.pack();
    frame.show();
  }
}
```

Problema: como adicionar funcionalidade à janela ?

• **Solução: usando herança, para gerar uma nova classe a partir de JFrame:**

```
import javax.swing.*;

class SimpleFrame extends JFrame
{
  private JButton but;
  private Jlabel texto;
  private Jpanel painel;

  public SimpleFrame() // cria um frame com tudo dentro
  {
    super("Swing Application");
    but = new JButton("I'm a Swing Button!");
    texto = new JLabel("Number of button clicks: 0");
    painel = new JPanel();

    painel.add(but);
    painel.add(texto);

    getContentPane().add(painel);
  }

  public static void main(String args[]) {
    SimpleFrame frame = new SimpleFrame();
    frame.pack();
    frame.show();
}
```

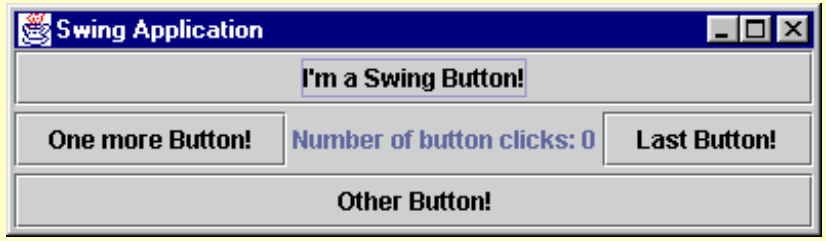
- Questões ainda não respondidas:



- Como organizar os componentes em um JPanel ?
- Java oferece diversos layouts para estruturação de componentes
- Por exemplo, para **JPanel** o layout default é **FlowLayout**, que distribui os componentes na horizontal.
- Mas e se quisermos distribuir de outro modo ?
- Basta trocar o layout por outro!

- **Box Layout:** posiciona componentes na vertical ou horizontal

```
import javax.swing.*;

class SimpleFrame extends JFrame
{
...
  public SimpleFrame() // cria um frame com tudo dentro
  {
    super("Swing Application");
...
    painel.setLayout(new BoxLayout(painel,BoxLayout.Y_AXIS));
    painel.add(but);
    painel.add(texto);

    getContentPane().add(painel);
  }
```



- **Border Layout:** organiza componentes em cinco áreas principais

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

class SimpleFrame extends JFrame
{
  Jbutton but,but2,but3,but4;
...
  public SimpleFrame() // cria um frame com tudo dentro
  {
...
    painel.setLayout(new BorderLayout(3,3));

    painel.add(but,BorderLayout.NORTH);
    painel.add(but2,BorderLayout.SOUTH);
    painel.add(but3,BorderLayout.WEST);
    painel.add(but4,BorderLayout.EAST);
    painel.add(texto,BorderLayout.CENTER);
...
  }
```

• Observe que é possível utilizar vários layouts para gerar o efeito desejado
• No exemplo acima, há dois JPanel, cada um responsável pelo layout de uma parte da interface:

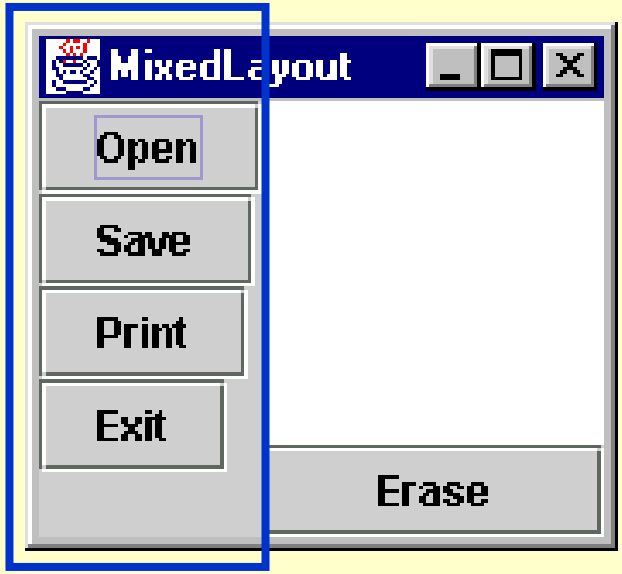
```
JPanel painelbot;   // painel para os botões
JPanel paineltexto; // painel para a área de texto e botão
```

• Há 4 botões para o primeiro painel...

```
JButton but1, but2, but3, but4;
```

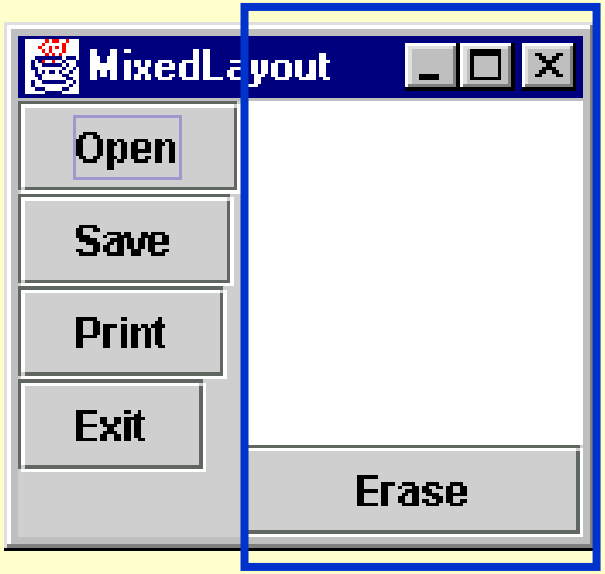
• ... e uma área de texto (JTextArea) + um botão para o segundo painel

```
JTextArea texto;
JButton apaga;
```



• O trecho de código abaixo cria e preenche o painel da esquerda:

```
but1 = new JButton("Open");
but2 = new JButton("Save");
but3 = new JButton("Print");
but4 = new JButton("Exit");

painelbot = new JPanel();
painelbot.setLayout(new BoxLayout(painelbot,BoxLayout.Y_AXIS));
painelbot.add(but1);
painelbot.add(but2);
painelbot.add(but3);
painelbot.add(but4);
```



• E o seguinte, cria e preenche o painel da direita (observe a utilização do BorderLayout):

```
texto = new JTextArea();
texto.setPreferredSize(new Dimension(100,100));
apaga = new JButton("Erase");

paineltexto = new JPanel();
paineltexto.setLayout(new BorderLayout());

paineltexto.add(texto, BorderLayout.CENTER);
paineltexto.add(apaga, BorderLayout.SOUTH);
```

• Falta apenas adicionar ambos ao content pane - detalhe: o **layout padrão** para content panes é BorderLayout, por isso não é necessário especificá-lo.

```
getContentPane().add(painelbot,BorderLayout.WEST);
getContentPane().add(paineltexto,BorderLayout.CENTER);
```

• Observe a especificação de posicionamento central da área de texto, para que esta ocupe efetivamente a maior parte do painel.