POO
Interface Gráfica - Swing

Adaptado de: http://189.80.82.102/matdidatico/eter/

Interface Gráfica

- Os elementos básicos necessários para criar um GUI (*Graphical User Interface* - Interface Gráfica do Usuário) residem em dois pacotes:
  - java.awt: Abstract Windowing Toolkit
  - javax.swing: Swing

AWT (JDK 1.0 – 1.1)

- Primeiro conjunto de classes Java para construir GUIs
- Conjunto de controles básicos
- Usa componentes do sistema operacional (pesados).
- Layout diferente em diferentes Oss.

Swing (JDK 1.2 ...)

- Mantem os conceitos de AWT, e tenta resolver seus problemas.
- Mais controles e opções flexíveis que em AWT.
- Componentes proprios (livianos).
- Layout consistente sobre todos os Oss.

- Diversos nomes de componentes AWT somente foram renomeados em Swing, ex:
  - AWT: `Button b = new Button ("OK");`
  - Swing: `JButton b = new JButton("OK");`

* OS: Sistema operativo

- Swing X AWT:
  - AWT é o conjunto original de componentes visuais de Java.
    - É implementada sobre componentes nativos do sistema operacional.
    - Programas Java rodando em diferentes plataformas possuem aparência diferente.
  - Swing é uma extensão da AWT.
    - Implementado inteiramente em Java.
    - Swing possui a mesma estrutura que os componentes AWT.
    - O pacote Swing apresenta componentes mais flexíveis.
  - As classes Swing são parte de um conjunto mais genérico de capacidades gráficas, chamado de *Java Foundation Classes* (JFC).

- JFC suporta:
  - definição de botões, menus, etc
  - desenho 2D (java.awt.geom)
  - funcionalidades drag-and-drop (java.awt.dnd)
  - API com acessibilidade a usuários (javax.accessibility)

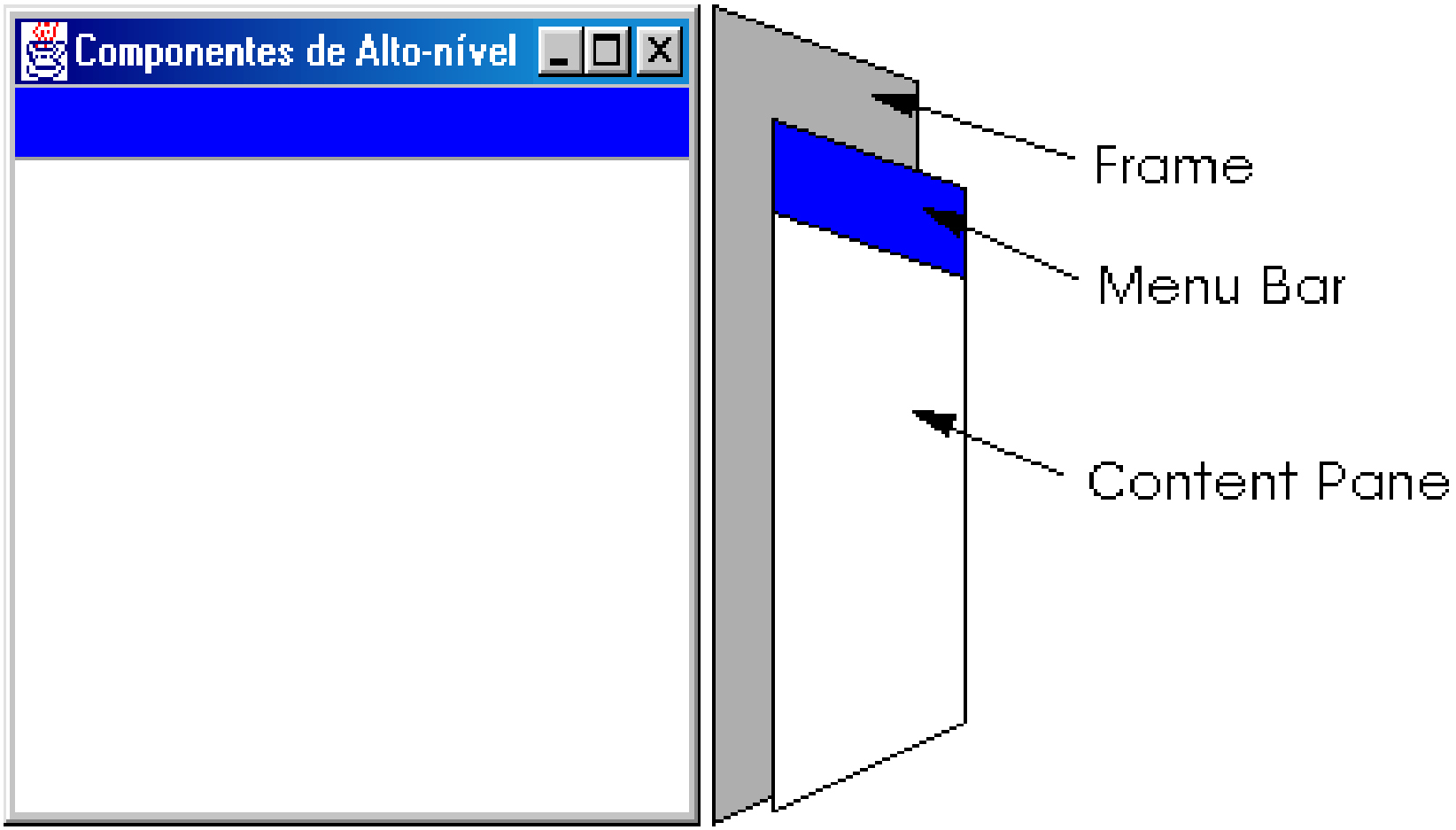
Swing

- Uma interface gráfica em Java é baseada em dois elementos:
  - Containers
    - Servem para agrupar e exibir outros componentes.
  - Componentes
    - São os botões, caixas de texto, barras de rolagem, labels, etc.
- Pacote javax.swing
  - `import javax.swing.*;`
  - composto de várias classes;
  - estudaremos as principais classes.

Containers

- Todo programa que oferece uma interface vai possuir pelo menos um container de mais alto nível, que pode ser:
  - JFrame: janela principal de um aplicativo
  - JDialog: janela para diálogos
  - JApplet: janela para Applets

- JFrame: Um objeto desta classe possui uma barra de título e características para receber menus e outros componentes.
- JDialog: Usada para definir janelas de diálogo para entrada de dados. Normalmente usada em resposta a uma opção de menu selecionada. Definida em função de um objeto JFrame.
- JApplet: Classe base para applets Java. É possível desenhar e adicionar menus e outros componentes em um JApplet.


Componentes de Alto-nível
Frame
Menu Bar
Content Pane

Exemplo:

```
import javax.swing.*;
public class TestaJanela{
  public static void main (String args[]){
    // Cria o objeto gráfico para a janela
    JFrame janela = new JFrame("Título da janela");
    // Seta posição e tamanho
    janela.setBounds(50, 100, 400, 150);
    // Ação ao fechar a janela = sair do programa
    janela.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE);
    // Exibe a janela
    janela.setVisible(true);
  }
}
```

Define a ação ao fechar a janela. Neste caso, indica o fim do aplicativo via System.exit(0).

Exercícios:

1) Recompile o programa e teste o funcionamento da janela com os seguintes parâmetros do método setDefaultCloseOperation():
   - DISPOSE_ON_CLOSE - Destrói a janela.
   - DO_NOTHING_ON_CLOSE - Desabilita opção default, o programador deve implementar o que acontece ao fechar a janela.
   - HIDE_ON_CLOSE - Apenas torna a janela não visível (opção padrão do JFrame).

   Obs: estas constantes estão definidas na interface WindowConstants.

2) O que acontece se retirarmos a linha com o método setVisible(true)?

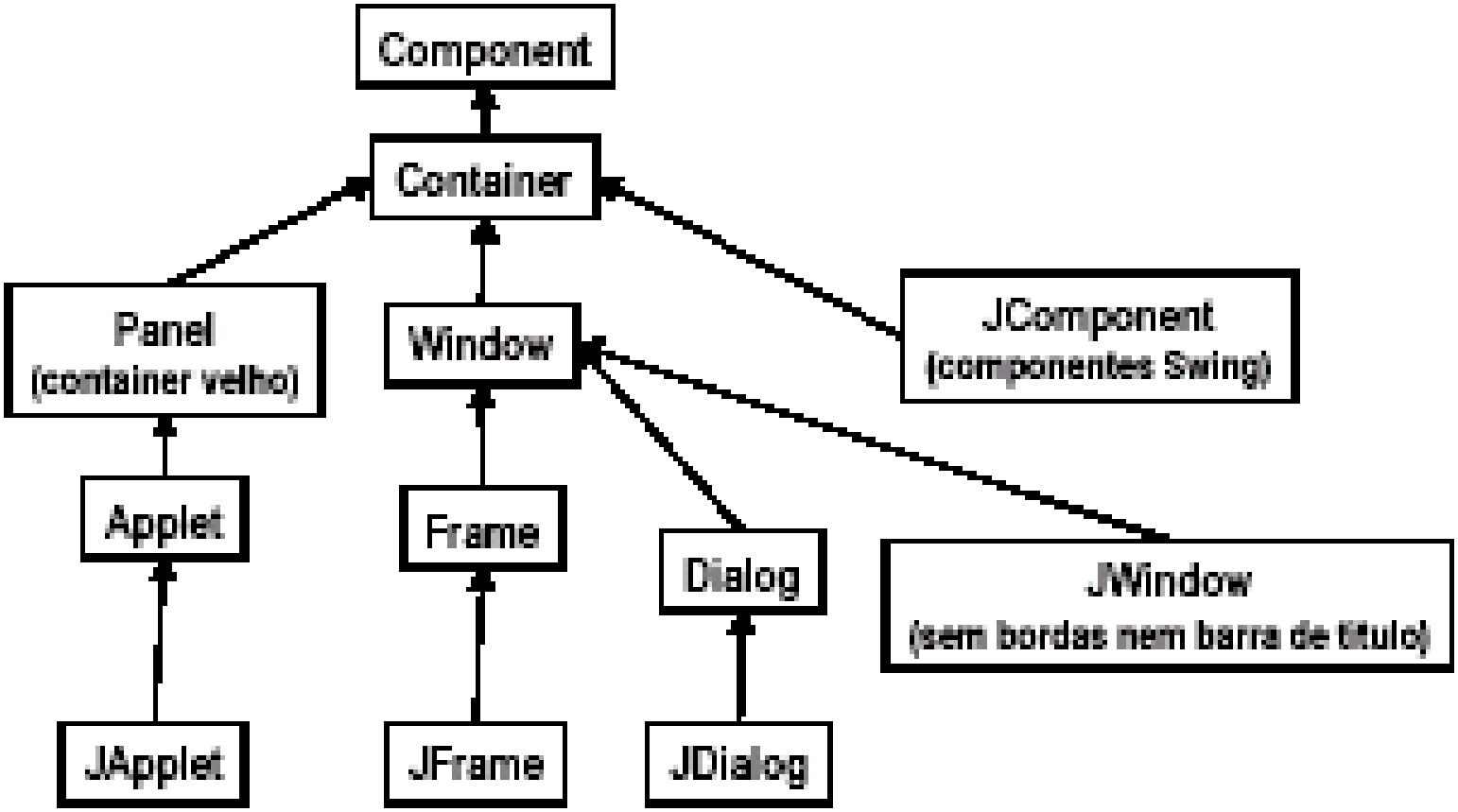
3) Faça um trecho de programa que anime uma janela, variando sua posição e tamanho.

4) Altere o programa do exemplo, acrescentando a seguinte linha de comando antes do método setVisible(true):

```
janela.setLocationRelativeTo(null);
```

A janela será criada centralizada na tela do computador!!!

- Hierarquia de classes:

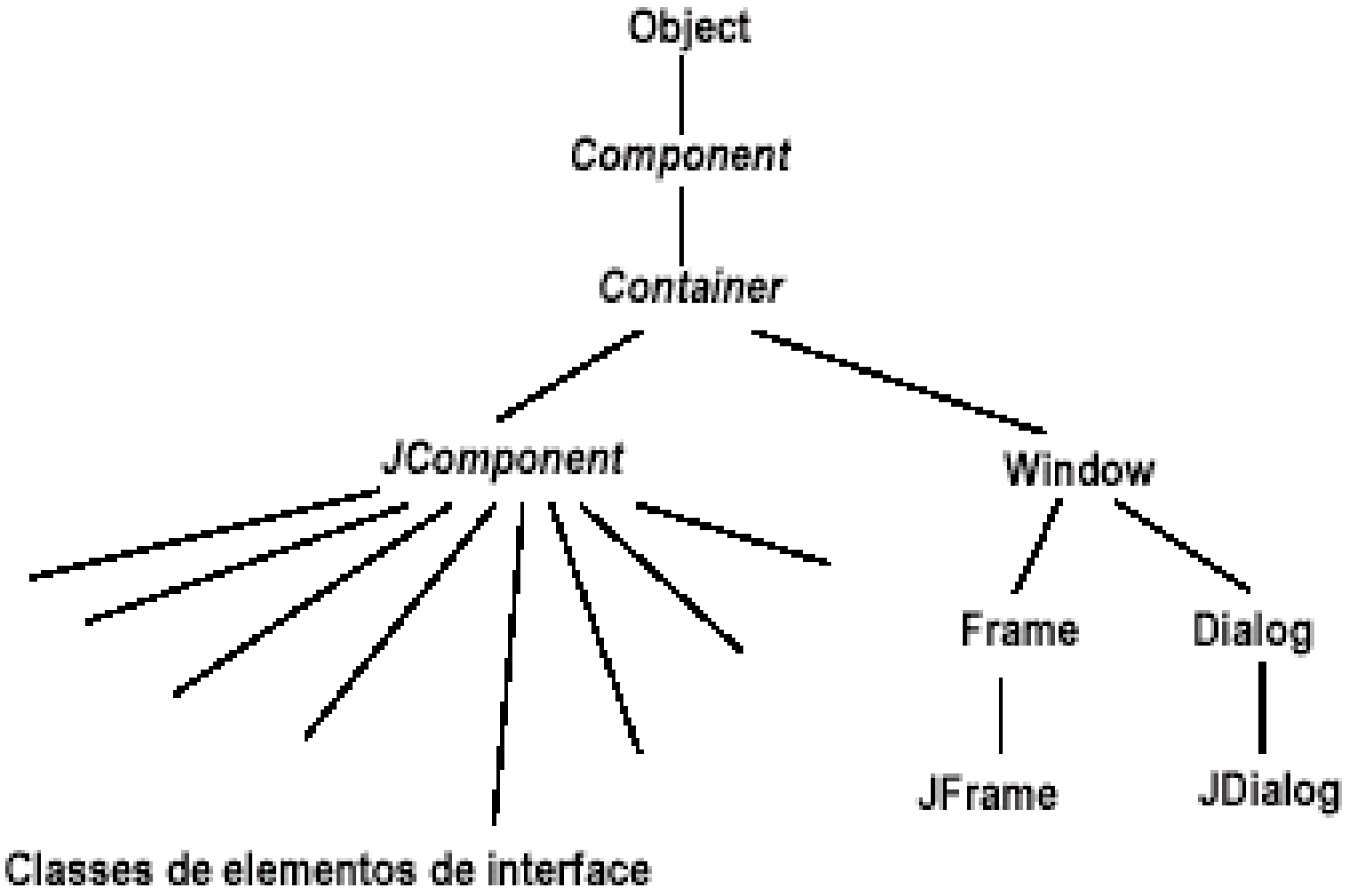


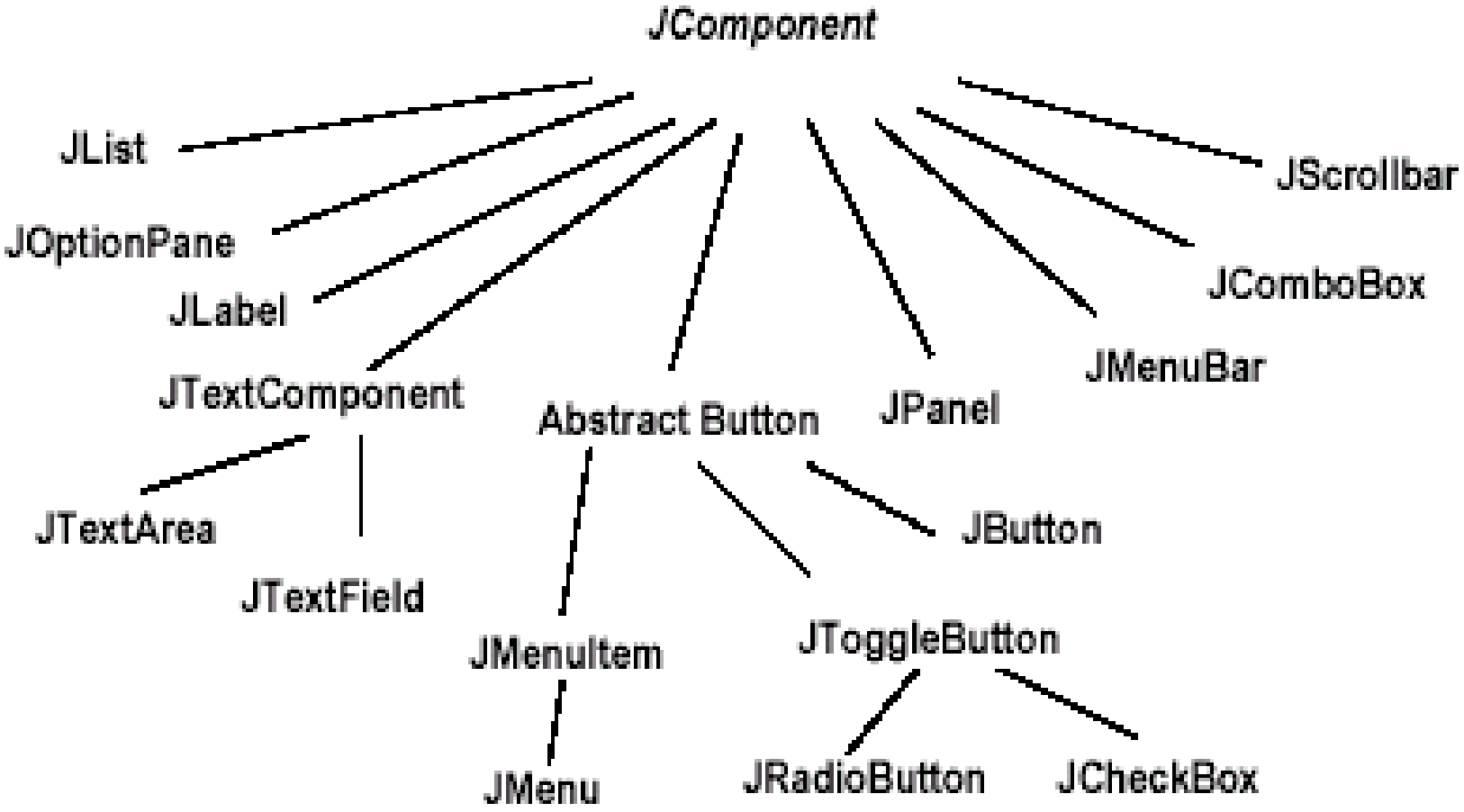
- Algumas classes da hierarquia são bastante simples, outras são complexas.
- JFrame, por exemplo:
  - 2 construtores, 236 métodos
    - 23 métodos declarados
    - 17 métodos herdados de Frame
    - 19 métodos herdados de Window
    - 44 métodos herdados de Container
    - 124 métodos herdados de Component
    - 9 métodos herdados de Object

Componentes

- Para construirmos uma interface gráfica em JAVA, adicionamos componentes (Botões, Menus, Textos, Tabelas, Listas, etc.) sobre a área da janela.
  - Por essa razão a área da janela é um *container*, ou seja, um elemento capaz de armazenar uma lista de componentes.
- JComponent: As subclasses de JComponent definem um conjunto de componentes standard (menus, botões, checkboxes, etc).

- Hierarquia de classes:





- Veja mais em:
  - http://java.sun.com/docs/books/tutorial/uiswing/components/components.html

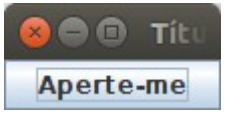
- Exemplo:


```
import javax.swing.*;
import java.awt.*;
public class TestaContainer{
  public static void main (String args[]){
    JFrame janela = new JFrame("Título da janela");
    janela.setBounds(50, 100, 400, 150);
    janela.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE);
    Container caixa = janela.getContentPane(); // Define o
   container
    caixa.add(new JButton("Aperte-me")); // Adiciona um botão
    janela.setVisible(true); // Exibe a janela
  }
}
```

Exercícios

1) Redimensione interativamente a janela da aplicação e observe o comportamento do botão da interface.

2) Adicione ao programa do exemplo, a chamada janela.pack() antes de janela.setVisible(true). Esse método redimensiona a janela para o tamanho mínimo para suportar os componentes visuais. O que aconteceu?



Exemplo2:

```
import javax.swing.*;
public class TestaContainer2{
  public static void main(String args[]){
    JFrame janela = new JFrame(”Testa Container 2");
    janela.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE);
    JButton botao = new JButton("Clique");
    JLabel texto = new JLabel("Número de cliques: 0");
    JPanel painel = new JPanel( ); // Container intermediário
    painel.add(botao);//Adiciona botão ao container
    painel.add(texto);//Adiciona label ao container
    janela.getContentPane().add(painel); //Adiciona painel ao
     container da janela
    janela.pack();//Redimensiona a janela
    janela.show();//Mostra a janela
  }
}
```

- No Exemplo2 temos os seguintes componentes:
  - JFrame: janela do aplicativo;
  - JPanel: painel intermediário, serve para facilitar o posicionamento do botão e do label;
  - JButton: o botão;
  - JLabel: o texto.
- JFrame são *top-level containers*: sempre estão presentes.
- JPanel são *intermediate containers*: podem estar ou não presentes (mas geralmente estão).
- JButton e JLabel são *componentes atômicos*: não podem ser usados para conter outros componentes e normalmente respondem a uma ação do usuário.

- 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!

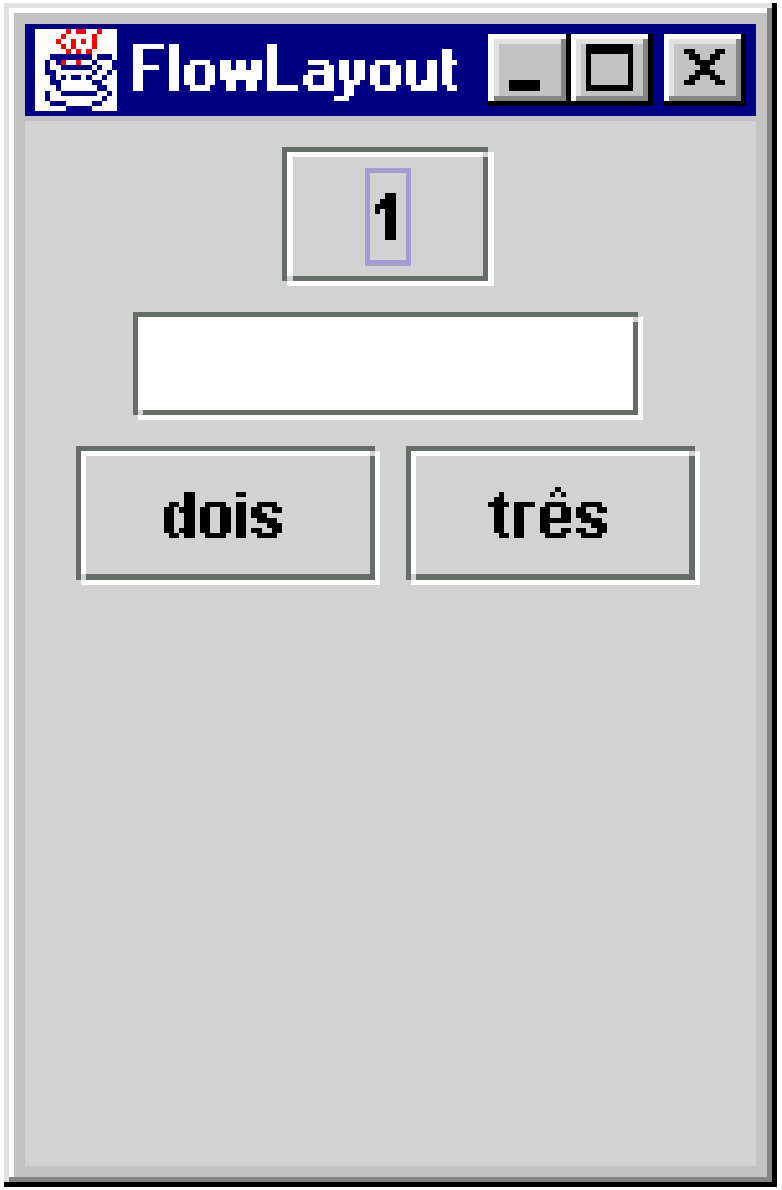
Layouts

- Como a filosofia da linguagem JAVA é de que os programas sejam extremamente portáveis, a filosofia da interface visa também ser extremamente adaptável.
- Por essa razão a disposição dos Componentes sobre o Container não é indicada por um par ordenado (x,y) como na maioria das bibliotecas de construção de interface com o usuário.

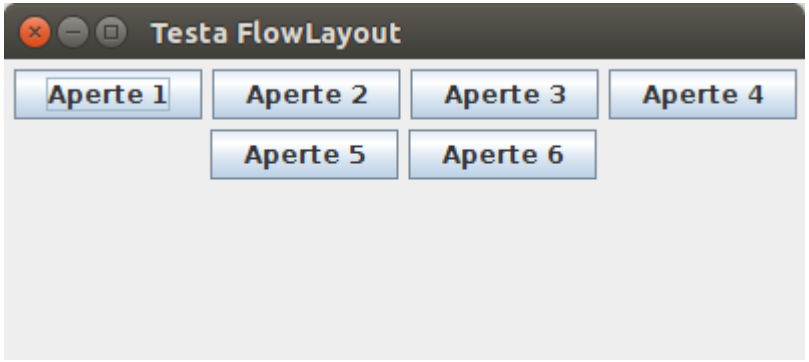
- O arranjo dos componentes no container é gerenciado por um *LayoutManager*.
  - A vantagem da existência de um LayoutManager é que a apresentação dos componentes se adapta quando do redimensionamento da janela.
  - A desvantagem é o pouco domínio que o programador tem da posição dos componentes com alguns dos gerenciadores de layout.
- É possível definir seus próprios layouts, mas a linguagem oferece um conjunto de layouts básicos que simplificam o trabalho.

- FlowLayout:
  - Coloca os componentes em uma fila da esquerda superior do container para a direita.
  - Respeita o tamanho preferido dos componentes mesmo quando não houver espaço suficiente no container.
  - É o padrão do JPanel.

```
JPanel c =new JPanel();
c.add(new JButton("1"));
c.add(new JTextField(9));
c.add(new JButton("dois"));
c.add(new JButton("três"));
```



Exemplo:




```
import javax.swing.*;
import java.awt.*;
public class TestaFlowLayout{
  public static void main (String args[]){
    int i;
    JFrame janela = new JFrame("Testa FlowLayout");
    janela.setBounds(50, 100, 400, 150);
    janela.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE);
    FlowLayout flow = new FlowLayout(); // Define o layout do container
    Container caixa = janela.getContentPane(); // Define o container
    caixa.setLayout(flow); // Seta layout do container
    for (i=1; i<=6; i++) caixa.add(new JButton("Aperte " + i));
    janela.setVisible(true);
  }
}
```

Exercícios

1) Redimensione interativamente a janela da aplicação e observe o comportamento dos botões da interface.
2) Troque o argumento do construtor FlowLayout(), para FlowLayout (FlowLayout.LEFT) e observe.

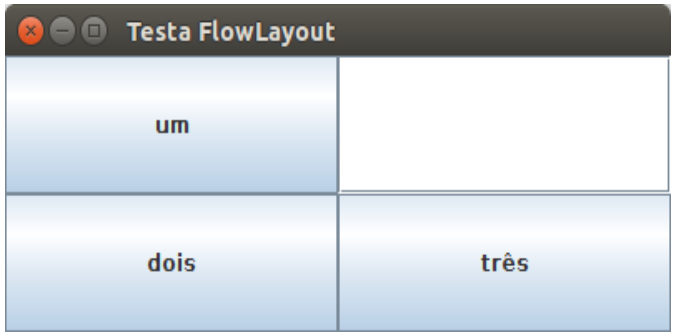
   O alinhamento dos componentes pode ser:

   - centralizado FlowLayout.CENTER (padrão)
   - esquerda FlowLayout.LEFT
   - direita FlowLayout.RIGHT

3) Adicione no programa acima, os seguintes componentes:
   - JLabel label = new JLabel("Exemplo de texto:");
   - caixa.add(label);
   - JTextField campo = new JTextField(15);
   - caixa.add(campo);
   - janela.pack();

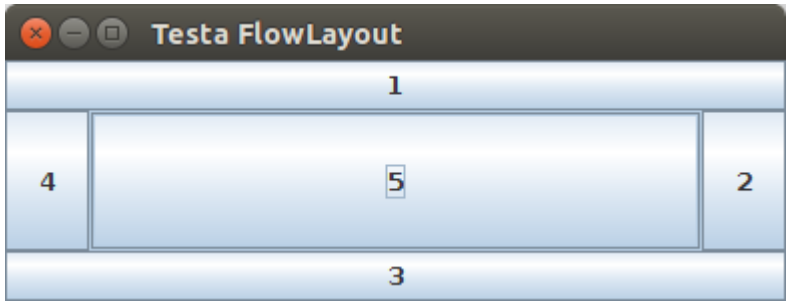
- GridLayout:
  - Divide o container em linhas e colunas.
  - Coloca os componentes da esquerda para a direita, de cima para baixo.
  - Todos os componentes terão o mesmo tamanho.
  - Permite definir um vetor ou matriz de células onde os componentes serão colocados.
  - Não respeita o tamanho original dos componentes.

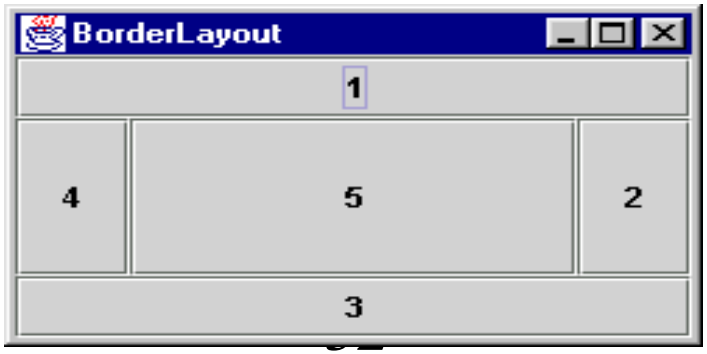
```
public class TestaGridLayout {
     public static void main (String args[]){
          JPanel c =new JPanel();
          c .setLayout(new GridLayout(2,2));
          c.add(new JButton("um"));
          c.add(new JTextField(5));
          c.add(new JButton("dois"));
          c.add(new JButton("três"));
          JFrame janela = new JFrame("Testa FlowLayout");
          janela.add(c);
          janela.setVisible(true);
          janela.setBounds(50, 100, 400, 150);
janela.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE);
     }
}
```



- BorderLayout:
  - Layout mais utilizado para a maioria das aplicações gráficas.
  - Divide um container em cinco regiões:
    - BorderLayout.CENTER
    - BorderLayout.NORTH
    - BorderLayout.EAST
    - BorderLayout.SOUTH
    - BorderLayout.WEST
  - Quando se adiciona um componente, é necessário especificar em qual das áreas ele deve ser adicionado.
    - Ex.: `add(butOK, BorderLayout.WEST);`
    - Componente irá ocupar todo o espaço!

```
public class TestaBorderLayout {
     public static void main (String args[]){
          JPanel c =new JPanel(new BorderLayout());
          JButton b1=new JButton("1");
          c.add(b1,BorderLayout.NORTH);
          JButton b2=new JButton("2");
          c.add(b2,BorderLayout.EAST);
          JButton b3=new JButton("3");
          c.add(b3,BorderLayout.SOUTH);
          JButton b4=new JButton("4");
          c.add(b4,BorderLayout.WEST);
          JButton b5=new JButton("5");
          c.add(b5,BorderLayout.CENTER);

          JFrame janela = new JFrame("Testa FlowLayout");
          janela.add(c);
          janela.setVisible(true);
          janela.setBounds(50, 100, 400, 150);
          janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
     }
}
```

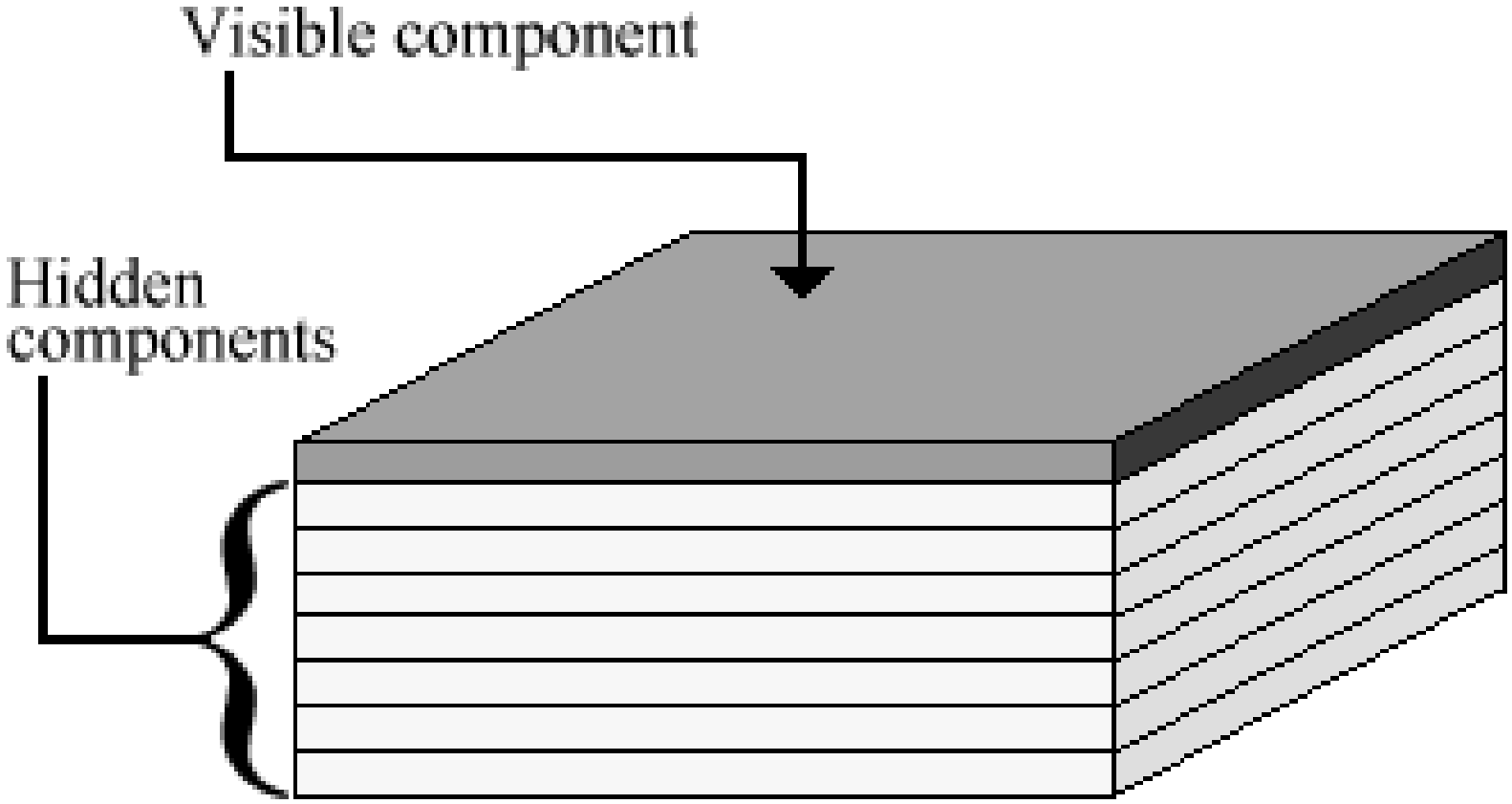




- BoxLayout:
  - Respeita o tamanho preferido dos componentes
  - Coloca os componentes em uma linha ou coluna.
    - BoxLayout.X_AXIS para componentes em linha
    - BoxLayout.Y_AXIS para componentes em coluna

```
public class TestaBoxLayout {
    public static void main (String args[]){
        JPanel c =new JPanel();
        c.setLayout(new BoxLayout(c, BoxLayout.Y_AXIS));
        c.add(new JButton("um"));
        c.add(new JButton("dois"));
        c.add(new JButton("três"));
        c.add(new JButton("quatro"));

        JFrame janela = new JFrame("Testa BoxLayout");
        janela.add(c);
        janela.setVisible(true);
        janela.setSize(200, 200);
        janela.setLocationRelativeTo(null);
        janela.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE);
    }
}
```

- CardLayout:
  - Comporta-se como a uma pilha, somente o objeto que estiver no topo será visível.
  - Métodos:
    - first(Container)
    - last(Container)
    - next(Container)
    - previous(Container)
    - show(Container, String)


Visible component
Hidden components

```java
public class CardLayoutExample extends JFrame {
	private JPanel pnlCard;
	private CardLayout cl;

	public CardLayoutExample() {
		// Carateristicas da Janela
		setTitle("Card Layout Example");
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setVisible(true);
		// Estabelecendo Layout de pnlCard
		pnlCard = new JPanel();
		cl = new CardLayout();
		pnlCard.setLayout(cl);
		// Criaçao de vários JPanels,
		//foram adicionados ao pnlCard
		JPanel p1 = new JPanel();
		JPanel p2 = new JPanel();
		JPanel p3 = new JPanel();
		JPanel p4 = new JPanel();
		p1.add(new JLabel("Card 1"));
		p2.add(new JLabel("Card 2"));
		p3.add(new JLabel("Card 3"));
		p4.add(new JLabel("Card 4"));
		pnlCard.add("Panel1", p1);
		pnlCard.add("Panel2", p2);
		pnlCard.add("Panel3", p3);
		pnlCard.add("Panel4", p4);
```

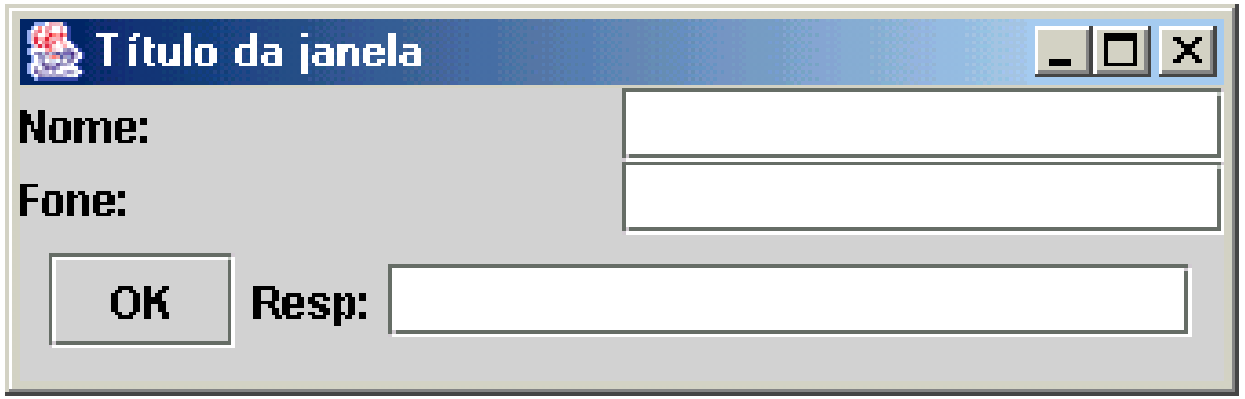

```java
// Criação de PnlButton
		JPanel pnlButton = new JPanel();
		JButton btnPrev = new JButton("Previous");
		JButton btnNext = new JButton("Next");
		pnlButton.add(btnPrev);
		pnlButton.add(btnNext);
		// Eventos para adiministrar os componentes do panel
com CardLayout
		btnPrev.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent arg0) {
				cl.previous(pnlCard);
			}
		});
		btnNext.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent arg0) {
				cl.next(pnlCard);
			}
		});
		getContentPane().add(pnlCard, BorderLayout.NORTH);
		getContentPane().add(pnlButton, BorderLayout.SOUTH);
	}

	public static void main(String[] args) {
		CardLayoutExample cl = new CardLayoutExample();
		cl.setSize(300, 150);
	}
}
```

- GridBagLayout:
  - Semelhante ao GridLayout, porem as divisões podem ter tamanhos diferentes.
  - Utiliza a classe GridBagConstraints para dimensionar e posicionar os componentes.
  - Um layout flexível e complicado usado quando nenhum dos outros se aplica.

```
public class FrmGridBag extends JFrame{
     public FrmGridBag(){
          super("GridBagLayout");
          JPanel p = new JPanel();
          p.setLayout(new GridBagLayout());
          // creates a constraints object
          GridBagConstraints c = new GridBagConstraints();
          c.insets = new Insets(2, 2, 2, 2); // insets for all components
          c.gridx = 0; // column 0
          c.gridy = 0; // row 0
          c.ipadx = 5; // increases components width by 10 pixels
          c.ipady = 5; // increases components height by 10 pixels
          p.add(new JButton("Java"), c); // constraints passed in
          c.gridx = 1; // column 1
          // c.gridy = 0; // comment out this for reusing the obj
          c.ipadx = 0; // resets the pad to 0
          c.ipady = 0;
          p.add(new JButton("Source"), c);
          c.gridx = 0; // column 0
          c.gridy = 1; // row 1
          c.gridwidth= 2;
          p.add(new JButton("and Support"), c);
          //c.gridx = 1; // column 1
          setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
          getContentPane().add(p);
     }
     public static void main(String[] args) {
          FrmGridBag frm = new FrmGridBag();
          frm.setSize(300, 15j0);
          frm.setVisible(true);
     }
}
```

- Layouts Compostos
  - A classe JPanel é derivada de Container e pode ser usada para agrupar Componentes de maneira a criar Layouts compostos.
  - Por ser também um Container, um JPanel pode ter seu Layout próprio.
  - Como um Container pode conter outros Containers, podemos agrupar vários Componentes em um JPanel e passar a considerá-los como um único Componente para efeitos de layout.
  - Dessa forma, utilizamos um Container com vários JPanel, cada um com seu layout.

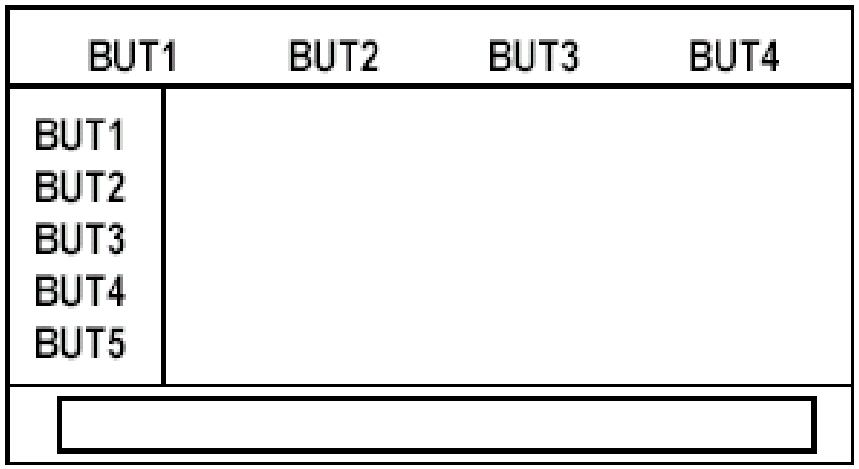
```
Container caixa = janela.getContentPane();
JPanel painel1 =new JPanel();
JPanel painel2 =new JPanel();
caixa.setLayout(new GridLayout(2,1));
painel1.setLayout(new GridLayout(2,2));
painel2.setLayout(new FlowLayout(FlowLayout.CENTER));
```



Exercícios

1) Altere o layout da janela do último exemplo de maneira que o botão de OK fique centrado na parte inferior da janela juntamente com um botão de CANCEL.

2) Crie o layout de uma janela que tenha 4 botões na parte superior, 5 botões no lado esquerdo, um campo de entrada de dados na parte inferior e deixa a área central livre para a edição de gráficos como no esquema abaixo:



Definindo Janelas

- Existem várias maneiras de definir as janelas:
  - Toda a definição dentro de um método `main`. (Péssima prática de programação)
  - Definir uma classe que herda de `JFrame` e utilizar um objeto dessa classe como a janela.
  - Definir uma classe que herda de `JPanel` e utilizar um objeto dessa classe dentro de um objeto `JFrame`.

- Situação 1:

```
public class Situacao1{
  public static void main(String[] args) {
    JFrame janela = new JFrame();
    ...//adicionar containers e componentes
    janela.pack();
    janela.show();
  }
}
```

Situação 2:

```
public class Situacao2 extends JFrame{
  public Situacao2(){
    super("Titulo"); //Construtor de Jframe
    ... //adicionar containers e components
  }
}
public class TesteSituacao2{
  public static void main(String[] args) {
    Situacao2 janela = new Situacao2();
    janela.pack();
    janela.show();
  }
}
```

- Situação 3:

```
public class Situacao3 extends JPanel{
  public Situacao3(){
    ...//adicionar containers e components
  }
}
public class TesteSituacao3{
  public static void main(String[] args) {
    JFrame janela = new JFrame("Titulo");
    janela.getContentPane().add(new Situacao3());
    janela.pack();
    janela.show();
  }
}
```

Exercícios:

1) Escolha um dos exemplos trabalhados e codifique utilizando as outras duas formas de definir a janela da interface gráfica.