Programação II Introdução a Gerenciadores de Layout

Ficha(s) Pratica(s)	*13,14*

Na aula passada verificamos que ao executar a nossa primeira janela, qualquer componente que era adicionado ao *JFrame* ocupada a tela inteira, se tivéssemos dois componentes, apenas o ultimo a ser adicionado é que era visualizado.

Então chegou-se a conclusão que os componentes por si próprio não se organizam, então durante este capitulo debruçar-se-á sobre o uso dos *LAYOUT* **.**

Gerenciadores de Layout

Uma das principais características da linguagem Java é a sua portabilidade que dispensa preocupações com aspetos relacionados com *hardware*. Imagine um programa que foi codificado para rodar numa máquina que possui um monitor com a resolução 1366 x 768 pixels sendo executado em um monitor de 800 x 600 pixels, isso provavelmente iria fazer com que os componentes sejam mal posicionados ou ate mesmo que o componente não seja visto completamente ao nível da tela. As linguagens compiladas como C++ exigem que o programador saiba de antemão para qual tipo de máquina esta desenvolvendo o programa, ou então o programa deve carregar as características na hora de execução, esse tipo de funcionamento agrega bastante complexidade ao algoritmo e reduz a portabilidade.

Em java não existe o posicionamento e dimensionamento dos componentes gráficos, isso é feito através de processos chamados Gerenciadores de layouts, isso torna o programa com portabilidade, porque o mesmo pode ser executado em um sistema operacional Windows com qualquer resolução ou então em outros sistemas operacionais como Linux.

Os principais gerenciadores de layout que serão analisados são:

- *FlowLayout*
- *GridLayout*
- *BorderLayout*
- *CardLayout*

FlowLayout

O *FlowLayout* é o gerenciador com as funções mais simples, isso porque os componentes são visualizados da esquerda para a direita na ordem da inserção. Quando não existe mais espaço ele quebra a linha colocando o componente numa outra linha, isso também pode ser testado usando o cursor do *mouse* para diminuir/aumentar o comprimento da tela.

Contrutores da classe FlowLayout

1. FlowLayout()
2. FlowLayout(int align)
3. FlowLayout(int align, int h, int v)

O parâmetro **align** permite especificar um novo alinhamento para os objetos, os possíveis valores são:

1. *FlowLayout.LEFT* – alinhamento a esquerda;
2. *FlowLayout.CENTER* – alinhamento centralizado;
3. *FlowLayout.RIGHT* – alinhamento a direita.

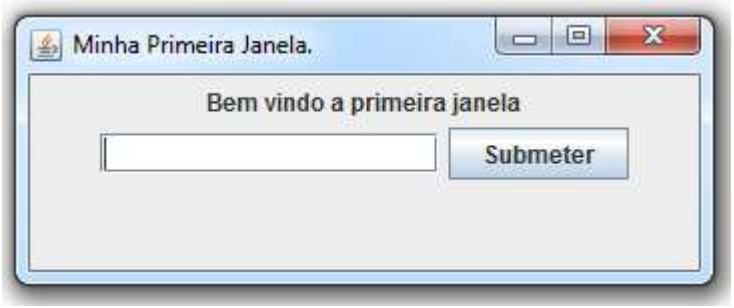
Os parâmetros **h** e **v** permitem indicar o espaçamento horizontal e vertical.

Sintaxe:

Dentro do construtor da classe, e antes de adicionar os componentes ao *JFrame*, deve-se indicar qual o layout que o *JFrame* vai seguir:

- `frame.setLayout(new FlowLayout());`

Agora implemente a seguinte tela:



```
==========================================================================
```

```
package GUI;

import java.awt.FlowLayout;

import javax.swing.JFrame;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JTextField;

public class FirstFrame {

	JFrame frame = new JFrame();
	JButton jb = new JButton("Submeter");
	JLabel jl = new JLabel("Bem vindo a primeira janela");
	JTextField jtf = new JTextField(15);


	public FirstFrame()
	{
		frame.setTitle("Minha Primeira Janela.");
		frame.setSize(350,150);
		frame.setLocation(50,50);
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		frame.setLayout(new FlowLayout());
		frame.add(jl);
		frame.add(jtf);
		frame.add(jb);
		frame.setVisible(true);
	}
}
```

```
==========================================================================
```

```
package GUI;

public class ArrancaPrimeiraJanela {
	public static void main(String[] args) {
		new FirstFrame();
	}
}
```

```
==========================================================================
```

GridLayout

O *GridLayout* organiza os componentes em uma grade imaginaria, todos os componentes desta grade são redimensionados junto com as células e as células possuem o mesmo tamanho.

Sintaxe

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

GridLayout(linhas, colunas, espaçamento-horizontal, espaçamento-vertical)
```

Crie agora uma janela simulando uma tela de autenticação, a tela ira possuir 1 titulo ex: “Autenticação”, uma caixa escrito utilizador e o respetivo lugar para preencher o valor um campo para escrito senha e o respetivo lugar para colocar a senha, e um botão com o nome submeter.

Nota: Como o *GridLayout* trabalha com células, quando tiver necessidade que uma célula fique

vazia, deve criar um *JLabel* vazio e adicionar ao *JFrame*, assim ficara um espaço vazio, assim como pode ver na imagem abaixo logo a seguir a "autenticação"

Veja o código necessário para gerar a tela acima:

```
==========================================================================
package GUI;

import java.awt.GridLayout;
import javax.swing.JFrame;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JTextField;

public class FirstFrame {

	JFrame frame = new JFrame();

	JLabel jl = new JLabel("Autenticação");
	JLabel jl_vazio = new JLabel();
	JLabel jl_vazio1 = new JLabel();
	JLabel jl_utilizador = new JLabel("Utilizador");
	JLabel jl_senha = new JLabel("Senha");
	JTextField jt_utilizador = new JTextField(15);
	JTextField jt_senha = new JTextField(15);
	JButton jb_submeter = new JButton("Submeter");


	public FirstFrame()
	{
		frame.setTitle("Minha Primeira Janela.");
		frame.setSize(350,150);
		frame.setLocation(50,50);
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		frame.setLayout(new GridLayout(4, 2));
		frame.add(jl);
		frame.add(jl_vazio);
		frame.add(jl_utilizador);
		frame.add(jt_utilizador);
		frame.add(jl_senha);
		frame.add(jt_senha);
		frame.add(jb_submeter);
		frame.setVisible(true);
	}
}
==========================================================================
```

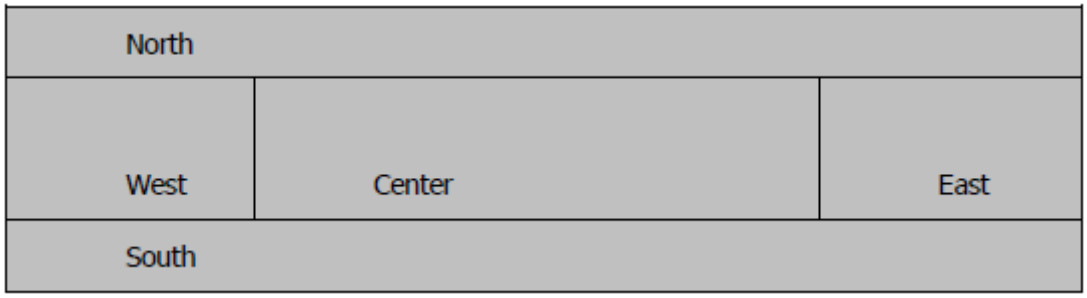
```
package GUI;

public class ArrancaPrimeiraJanela {
	public static void main(String[] args) {
		new FirstFrame();
	}
}
```

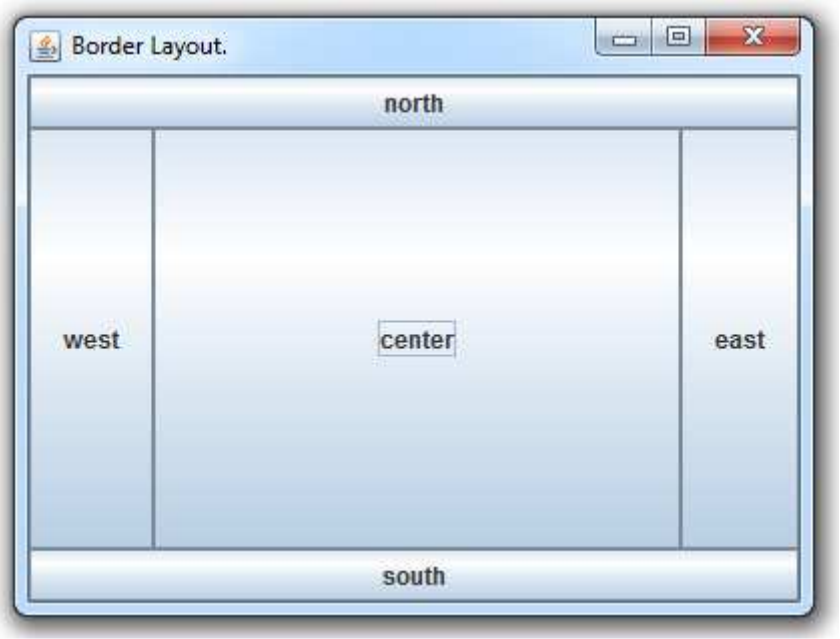
==========

BorderLayout

O *BorderLayout* é um gerenciador que divide um *container* em 5 partes nomeadamente uma parte superior (**North**), uma parte inferior (**South**), uma parte a esquerda (**West**), uma parte a direita (**East**) e uma parte na central (**Center**).



Em cada região é possível apenas colocar um componente, então como esse gerenciador possui cinco regiões, apenas cinco componentes poderão existir no máximo, veja o seguinte exemplo: cinco botões serão introduzidos no *container*, um em cada região.



Veja que em cada região apenas um componente foi colocado, mas os componentes não organizam-se aleatoriamente, deve-se indicar qual componente deve estar em qual região, isso deve ser indicado na hora em que o componente estiver a ser adicionado ao *container*.

```
frame.add("North", jb_b1);
frame.add("South", jb_b2);
```

```
frame.add("West", jb_b3);

frame.add("East", jb_b4);
frame.add("Center", jb_b5);
```

Veja o código que gerou a tela cima, agora faça algumas modificações, remova a região “EAST” e execute o programa, o que aconteceu? Agora remove a região “SOUTH”. Qual a sua conclusão para o que acontece quando um componente é removido do *container*?

```
================================================================================
package GUI;

import java.awt.BorderLayout;
import javax.swing.JButton;
import javax.swing.JFrame;

public class BorderLayoutManager {

	JFrame frame = new JFrame();
	JButton jb_b1 = new JButton("north");
	JButton jb_b2 = new JButton("south");
	JButton jb_b3 = new JButton("west");
	JButton jb_b4 = new JButton("east");
	JButton jb_b5 = new JButton("center");

	public BorderLayoutManager()
	{
		frame.setTitle("Border Layout.");
		frame.setSize(400,300);
		frame.setLocation(50,50);
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		frame.setLayout(new BorderLayout());
		frame.add("North", jb_b1);
		frame.add("South", jb_b2);
		frame.add("West", jb_b3);
		frame.add("East", jb_b4);
		frame.add("Center", jb_b5);
		frame.setVisible(true);

	}
}

================================================================================
```

O facto de apenas poder adicionar um componente em cada região poderia ser o suficiente para esse gerenciador ser menos usado pelos programadores, mais o Java tem um conceito de **Paineis**, quer dizer que ao invés de adicionarmos um componente em cada região, adicionamos um **Painel** em cada região que pertentemos colocar mais de um componente, e como cada **Painel** tem a capacidade de ter um próprio Layout logo podemos adicionar quantos componentes quisermos disposto de maneira que bem necessitarmos. Para isso vamos introduzir o conceito de Painel.

JPanel

Em alguns casos é necessário separar o *JFrame* em áreas para adicionar outros componentes, o Painel assim como o *JFrame* permitem que outros objetos sejam adicionados nele, em uma aplicação Java um ou mais Painéis podem estar contidos em um *JFrame*.

Sintaxe: `JPanel panel= new JPanel();`

Então continuando com o *BorderLayout*, agora pode-se usar esse gerenciador associado a vários *JPanel* formando assim uma interface robusta.


```
==========================================================================
package GUI;

import java.awt.BorderLayout;
import java.awt.GridLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

public class BorderLayoutManager {

	JFrame frame = new JFrame();

	JPanel panel1= new JPanel();
	JPanel panel2 = new JPanel();

	JLabel jl_titulo = new JLabel("======Pagina Inicial do Programa usando
	Border Layout======");
	JLabel jl_rodape = new JLabel("Direitos reservados @ Mujhahid");
	JLabel jl_label1 = new JLabel("Texto 1");
	JLabel jl_label2 = new JLabel("Texto 2");
```

```
	JLabel jl_label3 = new JLabel("Texto 3");
	JButton jb_b1 = new JButton("B1");
	JButton jb_b2 = new JButton("B2");
	JButton jb_b3 = new JButton("B3");
	JButton jb_b4 = new JButton("east");
	JButton jb_b5 = new JButton("center");

	public BorderLayoutManager()
	{
		frame.setTitle("Border Layout.");
		frame.setSize(400,300);
		frame.setLocation(50,50);
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		//Esse sera o gerenciador de layout para o frame
		frame.setLayout(new BorderLayout(2,2));
		frame.add("North", jl_titulo);

		/***
		 * Esse sera o gerenciador de layout
		 * para o panel1 que situa-se na zona "West"
		 */
		panel1.setLayout(new GridLayout(3,1,2,2));
		//agora adicionamos a esse panel os componentes
		panel1.add(jb_b1);
		panel1.add(jb_b2);
		panel1.add(jb_b3);

		/**
		 * Finalmente, levamos o panel com os componentes
		 * e adicionamos em alguma região pretentida do frame
		 */
		frame.add("West", panel1);

		/***
		 * Esse sera o gerenciador de layout
		 * para o panel2 que situa-se na zona "East"
		 */
		panel2.setLayout(new GridLayout(3,1));
		//agora adicionamos a esse panel os componentes
		panel2.add(jl_label1);
		panel2.add(jl_label2);
		panel2.add(jl_label3);

		/**
		 * Finalmente, levamos o panel com os componentes
		 * e adicionamos em alguma região pretentida do frame
		 */
		frame.add("East", panel2);

		frame.add("South", jl_rodape);
		frame.add("Center", jb_b5);
		frame.setVisible(true);
	}
}
```

==========

```
package GUI;

public class ArrancaPrimeiraJanela {
	public static void main(String[] args) {
		new BorderLayoutManager();
	}
}
```

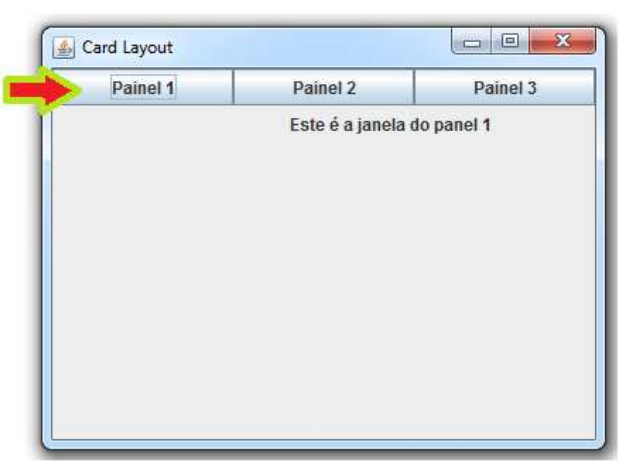
==========

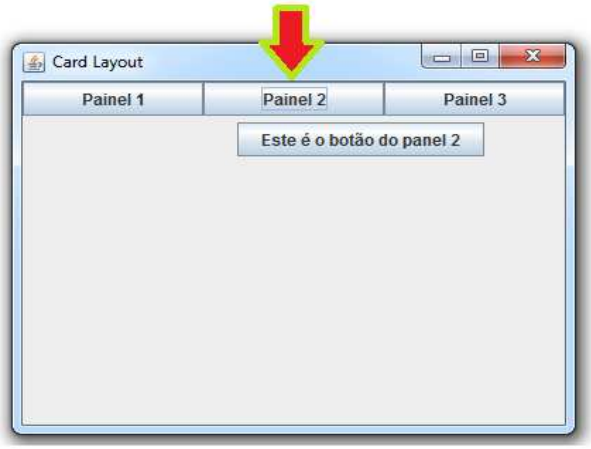
CardLayout

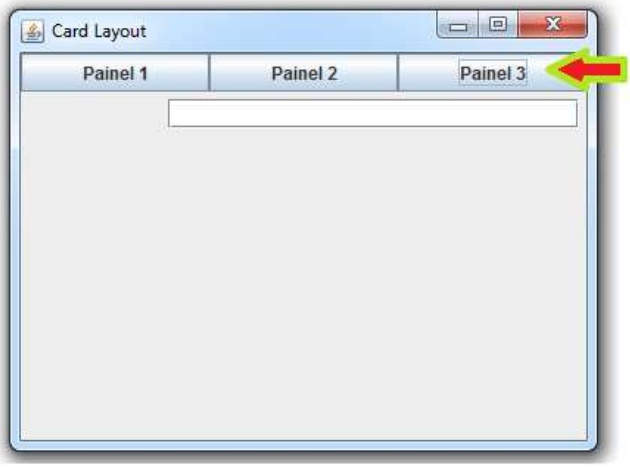
O **CardLayout** é um gerenciador de cartões, ele atua como se existisse uma pilha de cartões, onde cada cartão é representado por um *container*, cada um desses *container* pode ter um *layout* próprio. Cada cartão tem um nome e o método para trocar de cartão é *show()*.

Para usarmos o *CardLayout* irá se usar conceitos novos como tratamentos de eventos, para esta aula não se preocupe com esses conceitos apenas use para conseguir completar o exemplo do *CardLayout*, nas próximas aulas esse tópico será tratado com mais detalhes.

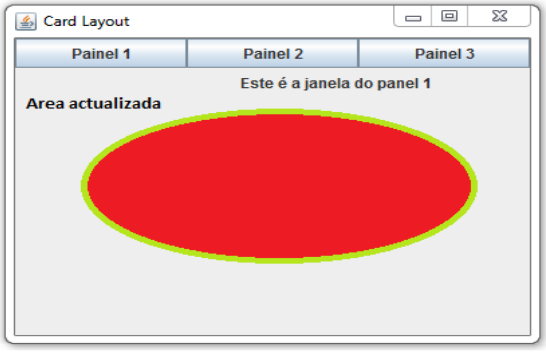
Veja a iteração do *CardLayout* ao clicar nos botões:







Note que criamos três painéis e quando o utilizador clica nos botões acima, o respetivo painel é invocado e atualizando a parte inferior da tela



```
================================================================================
```

```java
package GUI;

import java.awt.BorderLayout;
import java.awt.CardLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;

public class CardLayoutManager implements ActionListener{
    //criar um frame
      JFrame frame = new JFrame();
      //criar 3 paineis
      JPanel panel1= new JPanel();
      JPanel panel2 = new JPanel();
      JPanel panel3 = new JPanel();
      //criando rotulos
      JLabel jl_janela1 = new JLabel("Este é a janela do panel 1");
      JButton jb_janela2 = new JButton("Este é o botão do panel 2");
      JTextField jtf_janela3 = new JTextField(25);
      //criar dois botões
      JButton bt_1 = new JButton("Painel 1");
      JButton bt_2 = new JButton("Painel 2");
      JButton bt_3 = new JButton("Painel 3");

      JPanel selectPanel = new JPanel();
      JPanel cardLayoutPanel = new JPanel();


      public CardLayoutManager()
      {
            frame.setTitle("Card Layout");
            frame.setSize(400,300);
            frame.setLocation(50,50);
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            frame.setLayout(new BorderLayout());

            selectPanel.setLayout(new GridLayout(1,3));
            selectPanel.add(bt_1);
            selectPanel.add(bt_2);
            selectPanel.add(bt_3);

            panel1.add(jl_janela1);
            panel2.add(jb_janela2);
            panel3.add(jtf_janela3);

            cardLayoutPanel.setLayout(new CardLayout());

            cardLayoutPanel.add(panel1, "P1");
            cardLayoutPanel.add(panel2, "P2");
            cardLayoutPanel.add(panel3, "P3");

            frame.add("North", selectPanel);
            frame.add("East", cardLayoutPanel);
```

```
            bt_1.addActionListener(this);
            bt_2.addActionListener(this);
            bt_3.addActionListener(this);

            frame.setVisible(true);
      }


      @Override
      public void actionPerformed(ActionEvent action) {
            CardLayout cardLayout = (CardLayout) cardLayoutPanel.getLayout();
            if(action.getSource() == bt_1)
            {
                  cardLayout.show(cardLayoutPanel, "P1");
            }
            if(action.getSource() == bt_2)
            {
                  cardLayout.show(cardLayoutPanel, "P2");
            }
            if(action.getSource() == bt_3)
            {
                  cardLayout.show(cardLayoutPanel, "P3");
            }

      }
}
```

==========

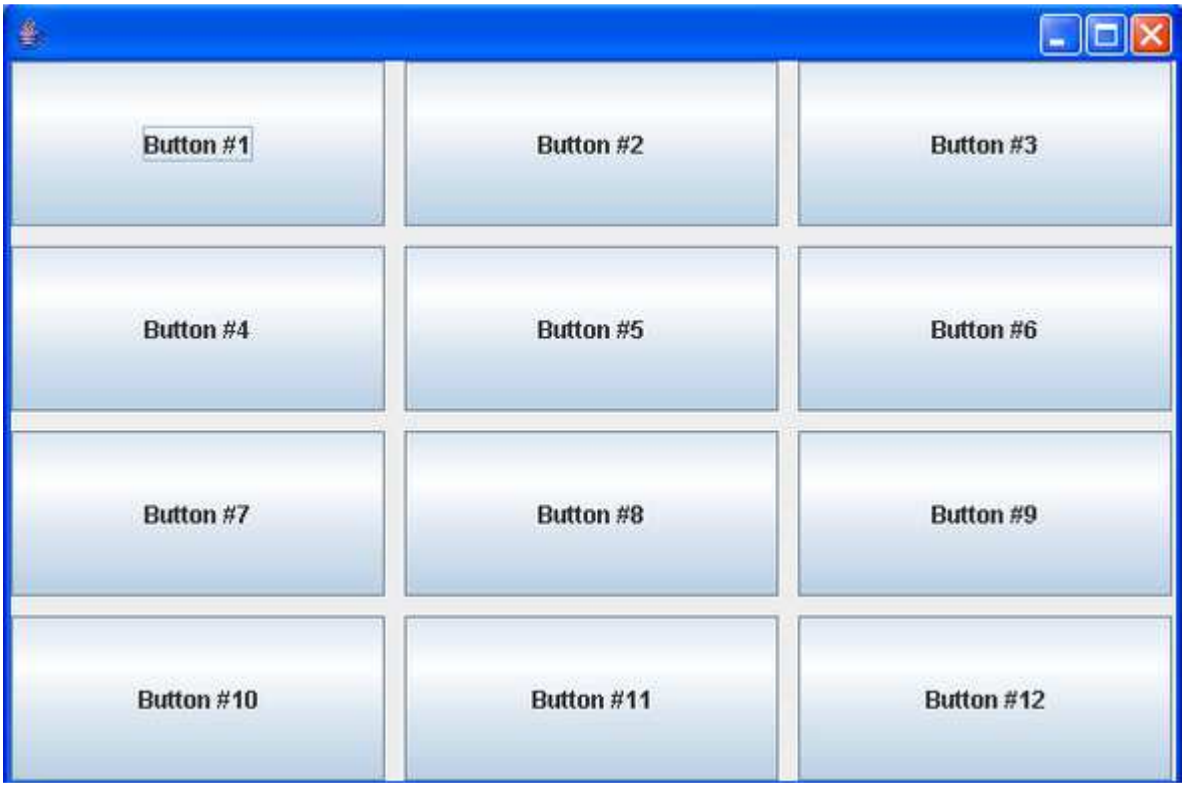
```
package GUI;

public class ArrancaPrimeiraJanela {
      public static void main(String[] args) {
            new CardLayoutManager();
      }
}
```

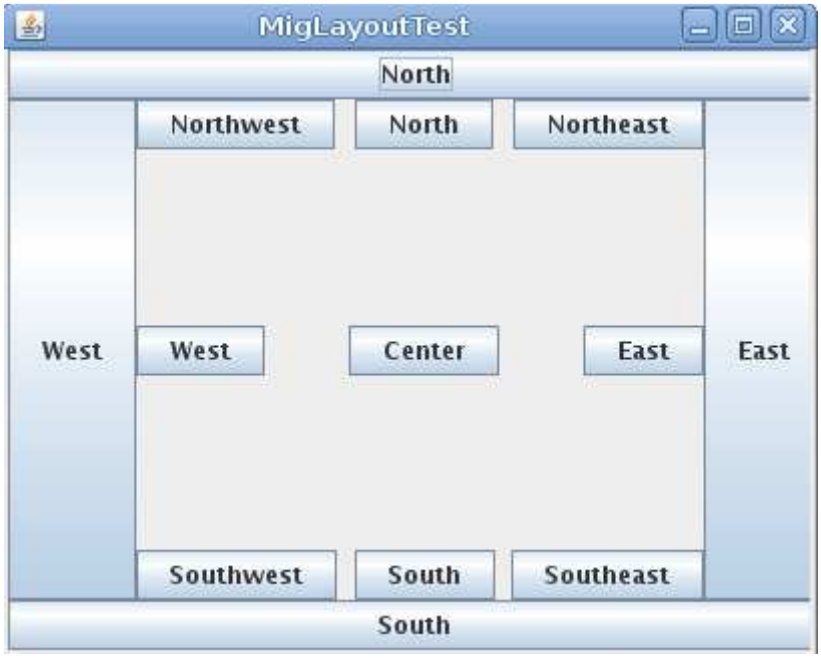
==========

Nota: A parte sublinhada a amarelo, será uma matéria para próximas aulas, por enquanto vai usando de modo a completarmos o exemplo do *GridLayout*.

Exercício 1
Analise o melhor gerenciador de Layout para desenhar a seguinte tela abaixo:



Exercício 2
Analise o melhor gerenciador de Layout para desenhar a seguinte tela abaixo:



Investigação:

Investigue os seguintes componentes e use dentro do seu *JFrame* para verificar o efeito:

1. setResizable
2. JPasswordField

--- FIM DO DOCUMENTO ---