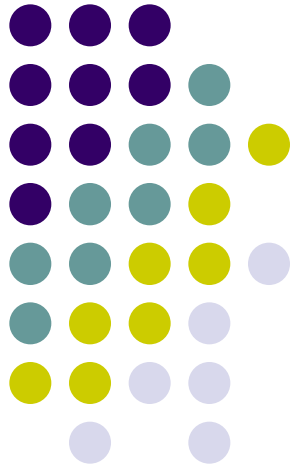
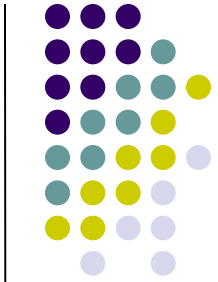
COM220

Aula 21: Gerenciadores de Layout Parte 2

Prof. Laércio Baldochi

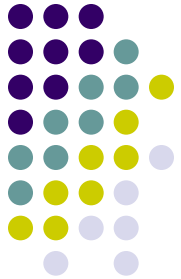


Gerenciadores de Layout
Parte 2



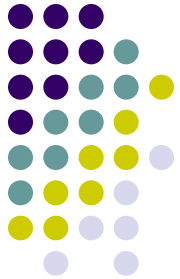
- CardLayout
- BoxLayout
- GridBagLayout

CardLayout



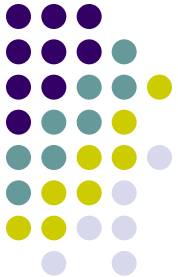
- Gerencia múltiplos painéis, que podem ser sobrepostos, compartilhando o mesmo espaço no container
- Funciona como uma pilha de cartões
  - Cartão que está no topo é o único visível
- Semelhante a JTabbedPane
  - mas... deve ser explicitamente programado

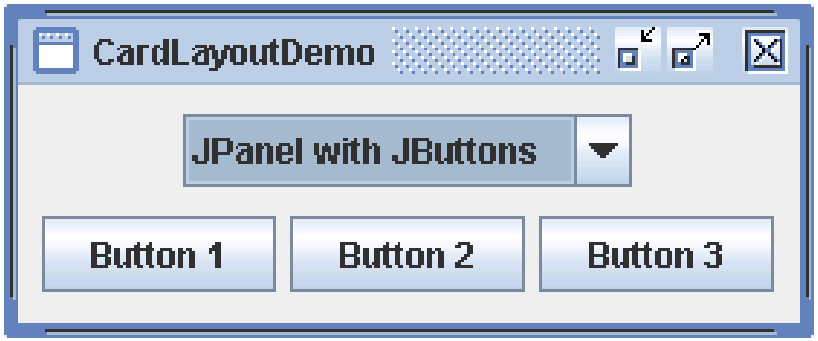
CardLayout

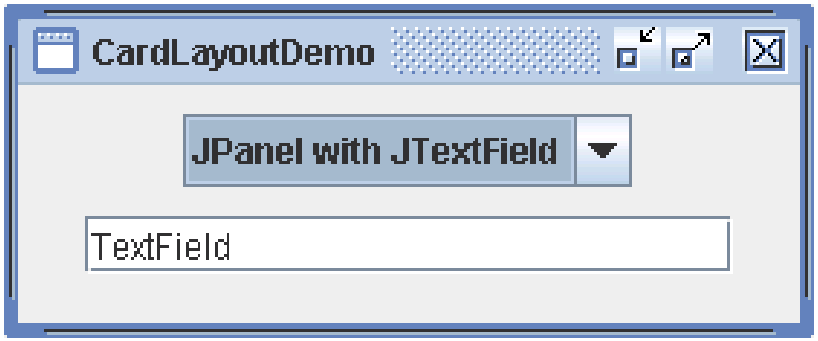


- Seleção dos painéis pode ser efetuada de 3 maneiras
  - Escolhendo o primeiro ou o último cartão por meio dos métodos first() e last()
  - Percorrendo os cartões para frente e para trás, através dos métodos next() e previous()
  - Especificando o cartão pelo nome, por meio do método show(parent,name)
    - parent -> container no qual o layout será aplicado

CardLayout
Exemplo

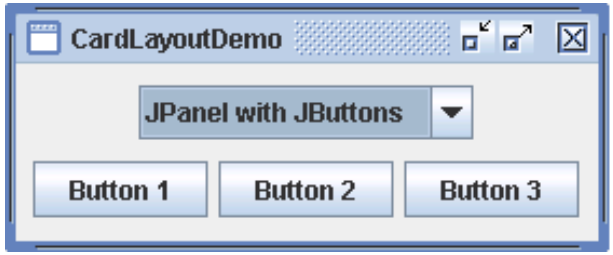


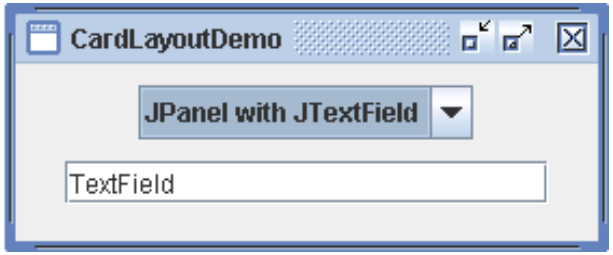




JComboBox é usado para escolher o painel a ser exibido

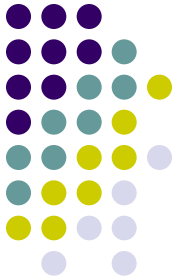
CardLayout Exemplo



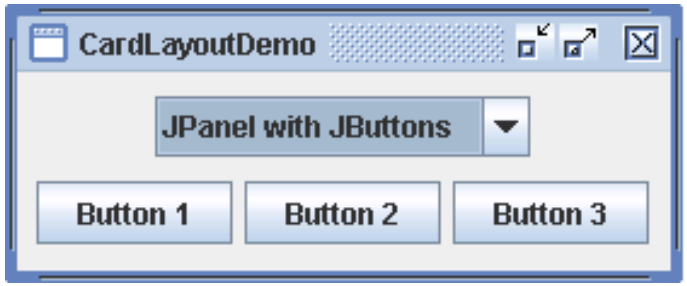


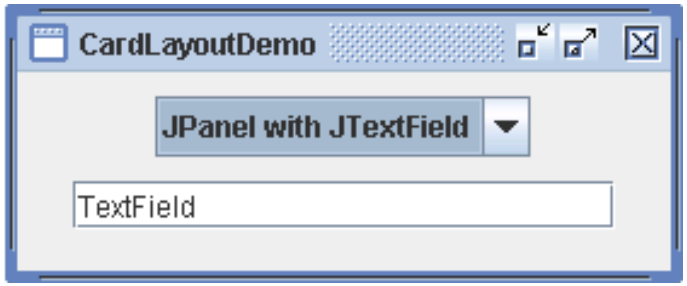

```
private final String BUTTON_PANEL =
  "JPanel with JButtons";
private final String TEXT_PANEL =
  "JPanel with JTextField";

private JPanel cards;
private JComboBox cbox;
...
// Cria o primeiro painel com três botões
JPanel p1 = new JPanel();
p1.add(new JButton("Button 1"));
p1.add(new JButton("Button 2"));
p1.add(new JButton("Button 3"));

// Cria o segundo painel com o campo de
  texto
JPanel p2 = new JPanel();
p2.add(new JTextField("TextField", 20));

// Inicializa o container e seta o layout
cards = new JPanel();
cards.setLayout(new CardLayout());
// Adiciona os painéis e associa um nome
  (string) a eles
cards.add(BUTTON_PANEL, p1);
cards.add(TEXT_PANEL, p2);
// Cria um combobox para selecionar os
  cartões
cbox = new JComboBox();
cbox.addItem(BUTTON_PANEL);
cbox.addItem(TEXT_PANEL);
cbox.addActionListener(this);
// Adiciona o combo em um painel
JPanel panel = new JPanel();
panel.add(cbox);
// Adiciona o painel de seleção e o painel
  com os cartões
setLayout(new BorderLayout());
add(panel, BorderLayout.PAGE_START);
add(cards, BorderLayout.CENTER);
```

CardLayout Exemplo



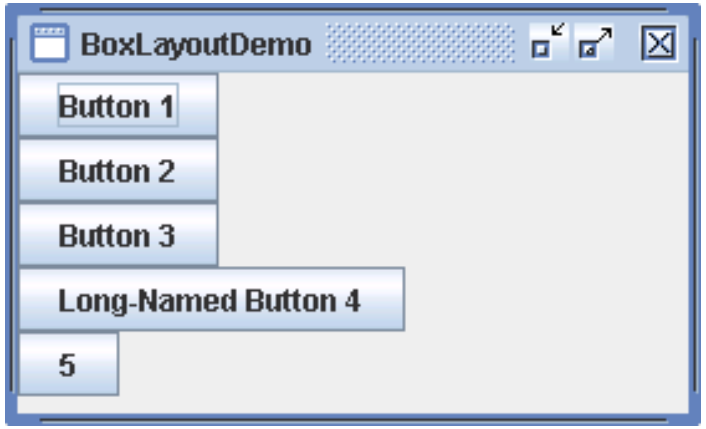
```
...
// Implementa o listener do combobox
  public void actionPerformed(ActionEvent e) {
     if (e.getSource() == cbox) {
        // Obtém o nome selecionado e mostra o cartão
        String option = (String) cbox.getSelectedItem();
        CardLayout layout = (CardLayout) cards.getLayout();
        layout.show(cards, option);
     }
  }
```



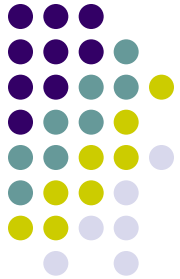


BoxLayout

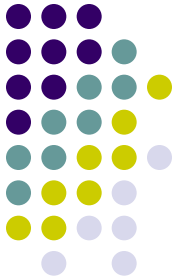
- Permite que múltiplos componentes sejam empilhados verticalmente ou enfileirados horizontalmente
- Semelhante ao FlowLayout, mas...
  - componentes não são reposicionados quando o container tem seu tamanho modificado



BoxLayout



- Construtor recebe como parâmetro
  - container em que será aplicado o layout
  - eixo em torno do qual os componentes serão posicionados
- Valores para eixo
  - X_AXIS
    - Horizontalmente, da esquerda para a direita
  - Y_AXIS
    - Verticalmente, de cima para baixo
  - LINE_AXIS
    - Posiciona os componentes como palavras em uma linha
  - PAGE_AXIS
    - Posiciona os componentes como linhas em um texto



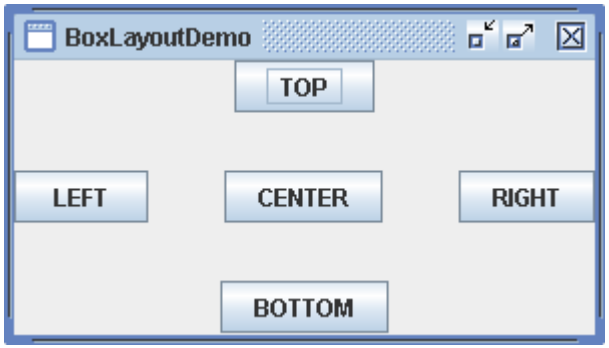
BoxLayout

- Componentes ficam "colados" um ao outro
- Alternativa
  - Inserir caixas invisíveis entre eles, para definir espaçamento ou posicionar corretamente um componente
- Classe Box permite criar 4 tipos de caixas invisíveis

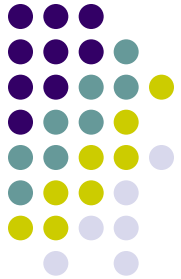
Tipo	Método	Efeito
Rigid Area	`Box.createRigidArea(size)`	Cria uma área de tamanho fixo entre os componentes
Vertical Glue	`Box.createVerticalGlue()`	Cria um espaço vertical flexível entre os componentes
Horizontal Glue	`Box.createHorizontalGlue()`	Cria um espaço horizontal flexível entre os componentes
Filler	`new Box.Filler(minSize, prefSize, maxSize )`	Cria um espaço com tamanho mínimo, preferido e máximo desejado

BoxLayout Exemplo

```
public BoxLayoutDemo() {
    // Cria os painéis e seta os layouts
    JPanel p1 = new JPanel();
    p1.setLayout(new BoxLayout(p1,
    BoxLayout.Y_AXIS));

    JPanel p2 = new JPanel();
    p2.setLayout(new BoxLayout(p2,
    BoxLayout.X_AXIS));
    p2.setAlignmentX(0.5f);

    // Cria os componentes e seta o alinhamento
    no centro (0.5f)
    JButton b1 = new JButton("  TOP  ");
    b1.setAlignmentX(0.5f);
    JButton b2 = new JButton("BOTTOM");
    b2.setAlignmentX(0.5f);

    JButton b3 = new JButton(" LEFT ");
    b3.setAlignmentY(0.5f);
    JButton b4 = new JButton("CENTER");
    b4.setAlignmentY(0.5f);
    JButton b5 = new JButton("RIGHT");
    b5.setAlignmentY(0.5f);

    // Adiciona os componentes
    p2.add(b3);
    p2.add(Box.createHorizontalGlue());
    p2.add(b4);
    p2.add(Box.createHorizontalGlue());
    p2.add(b5);

    p1.add(b1);
    p1.add(Box.createVerticalGlue());
    p1.add(p2);
    p1.add(Box.createVerticalGlue());
    p1.add(b2);
    // Inicializa o frame
    add(p1);
    setSize(300, 170);
    setTitle("BoxLayoutDemo");
    setVisible(true);
}
```



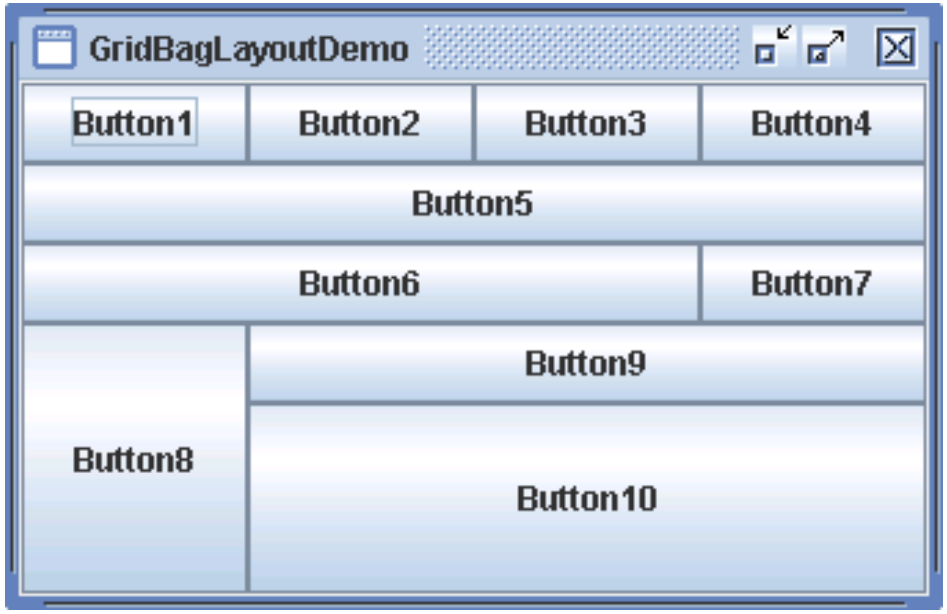
Exercício 1
Entrega: 18/06



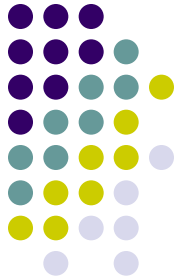
- Adicione à aplicação anterior quatro botões: TOP-LEFT, TOP-RIGHT, BOTTOM-LEFT e BOTTOM-RIGHT, utilizando o gerenciador BoxLayout.

GridBagLayout

- Variação do GridLayout
  - Cada componente pode ocupar uma ou mais células
  - Células não precisam ter o mesmo tamanho



GridBagLayout

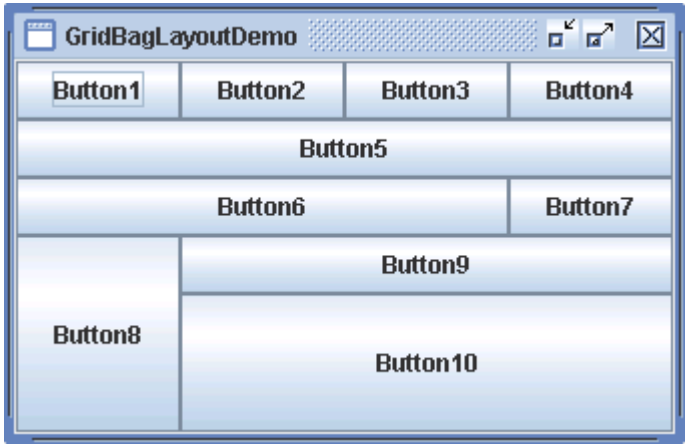


- GridBagConstraints
  - Classe usada para definir o tamanho e as características dos componentes inseridos no container
  - Utilizado no processo de adição de componentes

Variável	Função
gridx, Gridy	Indica a linha (gridy) e a coluna (gridx) na qual o componente será inserido (com início em zero)
gridwidth, gridheight	Especifica o número de colunas (gridwidth) e o número de linhas (gridheight) que o componente irá ocupar
Fill	Usado quando o componente é menor que a área que irá ocupar, determinando como esse irá preenchê-la
ipadx, ipady	Especifica o espaçamento interno do componente em relação às suas bordas verticais (ipadx) e horizontais (ipady)
Insets	Especifica o espaçamento externo do componente em relação às bordas da área que irá ocupar (por *default* é zero)
Anchor	Usado quando o componente é menor que a área que irá ocupar, determinando onde esse deve ser colocado
weightx, weighty	Determina como o gerenciador deve distribuir o espaço do container entre as linhas (weighty) e colunas (weightx)

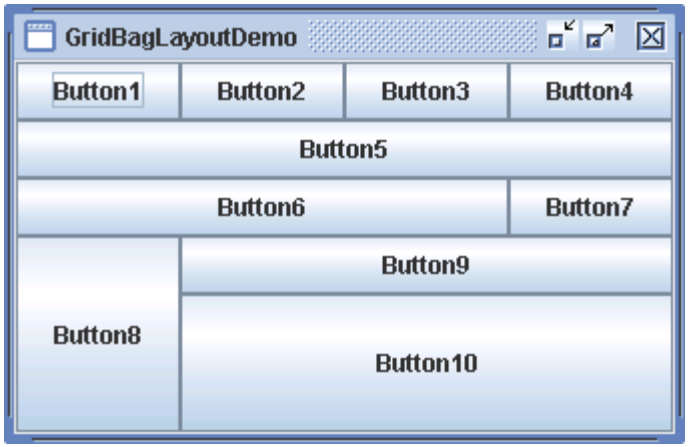
GridBagLayout Exemplo

```
// Seta o layout e cria o objeto com as restrições
setLayout(new GridBagLayout());
GridBagConstraints c = new GridBagConstraints();

// Define as restrições comuns a todos os componentes
c.fill = GridBagConstraints.BOTH;     // Preencher nas duas direções
c.weightx = 1.0;                      // Colunas devem ocupar todo o espaço
c.weighty = 1.0;                      // Linhas devem ocupar todo o espaço

// Cria os componentes, seta as restrições de cada um e
// adiciona ao container
JButton b1 = new JButton("Button1");     // Linha 0, Coluna 0
add(b1, c);
JButton b2 = new JButton("Button2");     // Linha 0, Coluna 1
c.gridx = 1;
add(b2, c);
JButton b3 = new JButton("Button3");     // Linha 0, Coluna 2
c.gridx = 2;
add(b3, c);
JButton b4 = new JButton("Button4");     // Linha 0, Coluna 3
c.gridx = 3;
add(b4, c);
JButton b5 = new JButton("Button5");     // Linha 1, Coluna 0
c.gridy = 1;
c.gridx = 0;
c.gridwidth = 4;    // Largura 4
add(b5, c);
```

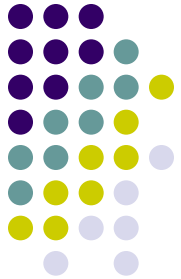


GridBagLayout - Exemplo

```
JButton b6 = new JButton("Button6");    // Linha 2, Coluna 0
c.gridy = 2;
c.gridx = 0;
c.gridwidth = 3;    // Largura 3
add(b6, c);

JButton b7 = new JButton("Button7");    // Linha 2, Coluna 3
c.gridx = 3;
c.gridwidth = 1;    // Largura 1
add(b7, c);

JButton b8 = new JButton("Button8");    // Linha 3, Coluna 0
c.ipady = 20;    // Torna o componente mais alto
c.gridy = 3;
c.gridx = 0;
c.gridheight = 2;    // Altura 2
add(b8, c);

JButton b9 = new JButton("Button9");    // Linha 3, Coluna 1
c.ipady = 0;    // Reseta os valores default
c.gridy = 3;
c.gridx = 1;
c.gridwidth = 3;    // Largura 3
c.gridheight = 1;    // Altura 1
add(b9, c);

JButton b10 = new JButton("Button10");    // Linha 4, Coluna 1
c.ipady = 40;    // Torna o componente mais alto
c.gridy = 4;
add(b10, c);
```



Exercícios
Entrega: 19/06



2) Implemente a aplicação do **slide 6** utilizando um JTabbedPane, ao invés de um gerenciador CardLayout.

3) Construa um formulário contendo quatro JLabels e quatro JTextFields para entrada de nome, endereço, telefone e e-mail do usuário. Isso deve ser feito utilizando o gerenciador de *layout* SpringLayout.