Algoritmos III
Swing: Layout

Raphael de Souza, Karen Figueiredo

IC/UFMT

Roteiro

1. Swing: Layout
   - java.awt.FlowLayout
   - java.awt.BorderLayout
   - java.awt.GridLayout

Layout

- java.awt.FlowLayout
- java.awt.BorderLayout
- java.awt.GridLayout
- java.awt.GridBagLayout
- java.awt.CardLayout
- java.awt.BoxLayout
- java.awt.SpringLayout
- Etc...

java.awt.FlowLayout

Definição

Os componentes são colocados sequencialmente da esquerda para direita na ordem em que foram adicionados

java.awt.FlowLayout

Definição

Os componentes são colocados sequencialmente da esquerda para direita na ordem em que foram adicionados

java.awt.FlowLayout

- Admite três opções de alinhamento, através do método setAlignment():
  - FlowLayout.CENTER
  - FlowLayout.LEFT
  - FlowLayout.RIGHT
- Quando falta espaço no container os componentes caem para a próxima linha

java.awt.FlowLayout

- Admite três opções de alinhamento, através do método setAlignment():
  - FlowLayout.CENTER
  - FlowLayout.LEFT
  - FlowLayout.RIGHT
- Quando falta espaço no container os componentes caem para a próxima linha

java.awt.FlowLayout

- Admite três opções de alinhamento, através do método setAlignment():
  - FlowLayout.CENTER
  - FlowLayout.LEFT
  - FlowLayout.RIGHT
- Quando falta espaço no container os componentes caem para a próxima linha

java.awt.FlowLayout

- Admite três opções de alinhamento, através do método setAlignment():
  - FlowLayout.CENTER
  - FlowLayout.LEFT
  - FlowLayout.RIGHT
- Quando falta espaço no container os componentes caem para a próxima linha

java.awt.FlowLayout

- Admite três opções de alinhamento, através do método setAlignment():
  - FlowLayout.CENTER
  - FlowLayout.LEFT
  - FlowLayout.RIGHT
- Quando falta espaço no container os componentes caem para a próxima linha

java.awt.FlowLayout

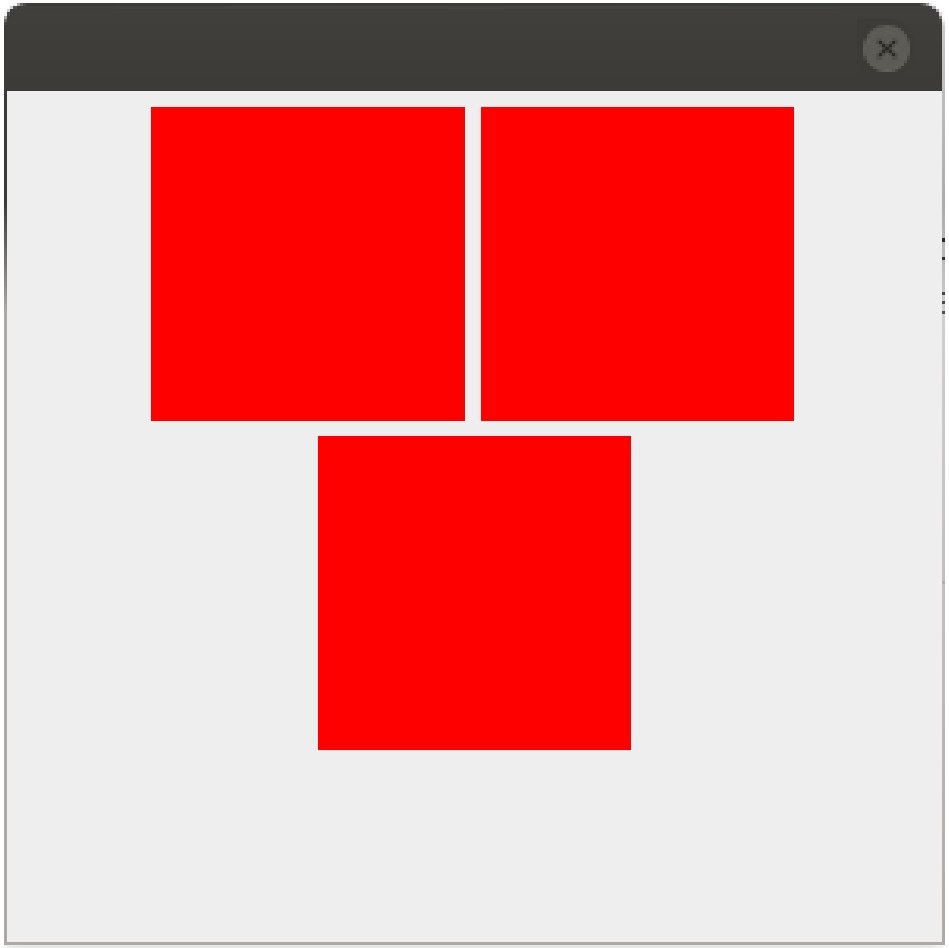
- Admite três opções de alinhamento, através do método setAlignment():
  - FlowLayout.CENTER
  - FlowLayout.LEFT
  - FlowLayout.RIGHT
- Quando falta espaço no container os componentes caem para a próxima linha

java.awt.FlowLayout

```
import javax.swing.JPanel;
...
public class Tela extends javax.swing.JFrame {
  public Tela() {
    setLayout(new java.awt.FlowLayout());
    setSize(300, 300);
    setDefaultCloseOperation(EXIT_ON_CLOSE);

    for (int i = 0; i < 3; i++) {
      JPanel panel = new JPanel();
      panel.setBackground(java.awt.Color.red);
      panel.setSize(100, 100);
      panel.setPreferredSize(panel.getSize());
      add(panel);
    }
  }
}
```

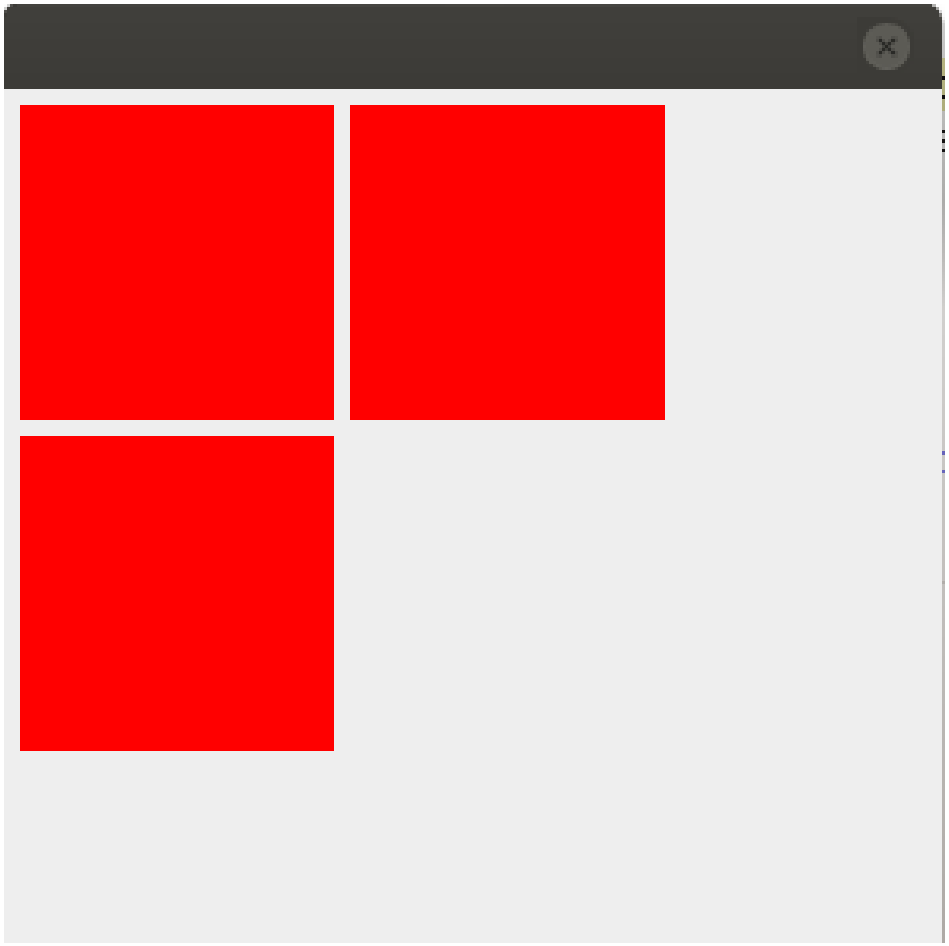
java.awt.FlowLayout



java.awt.FlowLayout

Colocando o alinhamento para a esquerda.

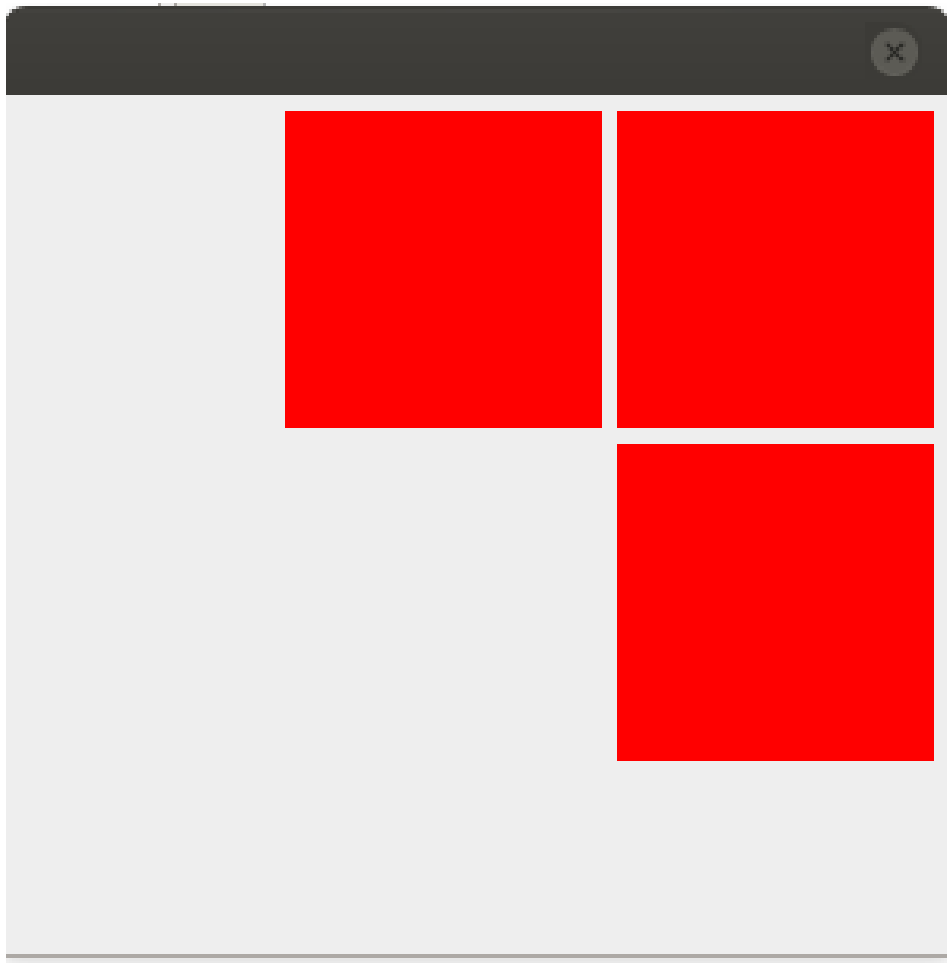
```
setLayout(new FlowLayout(FlowLayout.LEFT));
```



java.awt.FlowLayout

Colocando o alinhamento para a direita.

```
setLayout(new FlowLayout(FlowLayout.RIGHT));
```



java.awt.BorderLayout

Definição

Organização de componentes em cinco áreas de tela: centro e os pontos cardeais

java.awt.BorderLayout

Definição

Organização de componentes em cinco áreas de tela: centro e os pontos cardeais

java.awt.BorderLayout

- Admite cinco posições na tela:
  - BorderLayout.CENTER
  - BorderLayout.EAST
  - BorderLayout.WEST
  - BorderLayout.NORTH
  - BorderLayout.SOUTH
- Apenas um componente pode ser adicionado em cada região

java.awt.BorderLayout

- Admite cinco posições na tela:
  - BorderLayout.CENTER
  - BorderLayout.EAST
  - BorderLayout.WEST
  - BorderLayout.NORTH
  - BorderLayout.SOUTH
- Apenas um componente pode ser adicionado em cada região

java.awt.BorderLayout

- Admite cinco posições na tela:
  - BorderLayout.CENTER
  - BorderLayout.EAST
  - BorderLayout.WEST
  - BorderLayout.NORTH
  - BorderLayout.SOUTH
- Apenas um componente pode ser adicionado em cada região

java.awt.BorderLayout

- Admite cinco posições na tela:
  - BorderLayout.CENTER
  - BorderLayout.EAST
  - BorderLayout.WEST
  - BorderLayout.NORTH
  - BorderLayout.SOUTH
- Apenas um componente pode ser adicionado em cada região

java.awt.BorderLayout

- Admite cinco posições na tela:
  - BorderLayout.CENTER
  - BorderLayout.EAST
  - BorderLayout.WEST
  - BorderLayout.NORTH
  - BorderLayout.SOUTH
- Apenas um componente pode ser adicionado em cada região

java.awt.BorderLayout

- Admite cinco posições na tela:
  - BorderLayout.CENTER
  - BorderLayout.EAST
  - BorderLayout.WEST
  - BorderLayout.NORTH
  - BorderLayout.SOUTH
- Apenas um componente pode ser adicionado em cada região

java.awt.BorderLayout

- Admite cinco posições na tela:
  - BorderLayout.CENTER
  - BorderLayout.EAST
  - BorderLayout.WEST
  - BorderLayout.NORTH
  - BorderLayout.SOUTH
- Apenas um componente pode ser adicionado em cada região

java.awt.BorderLayout

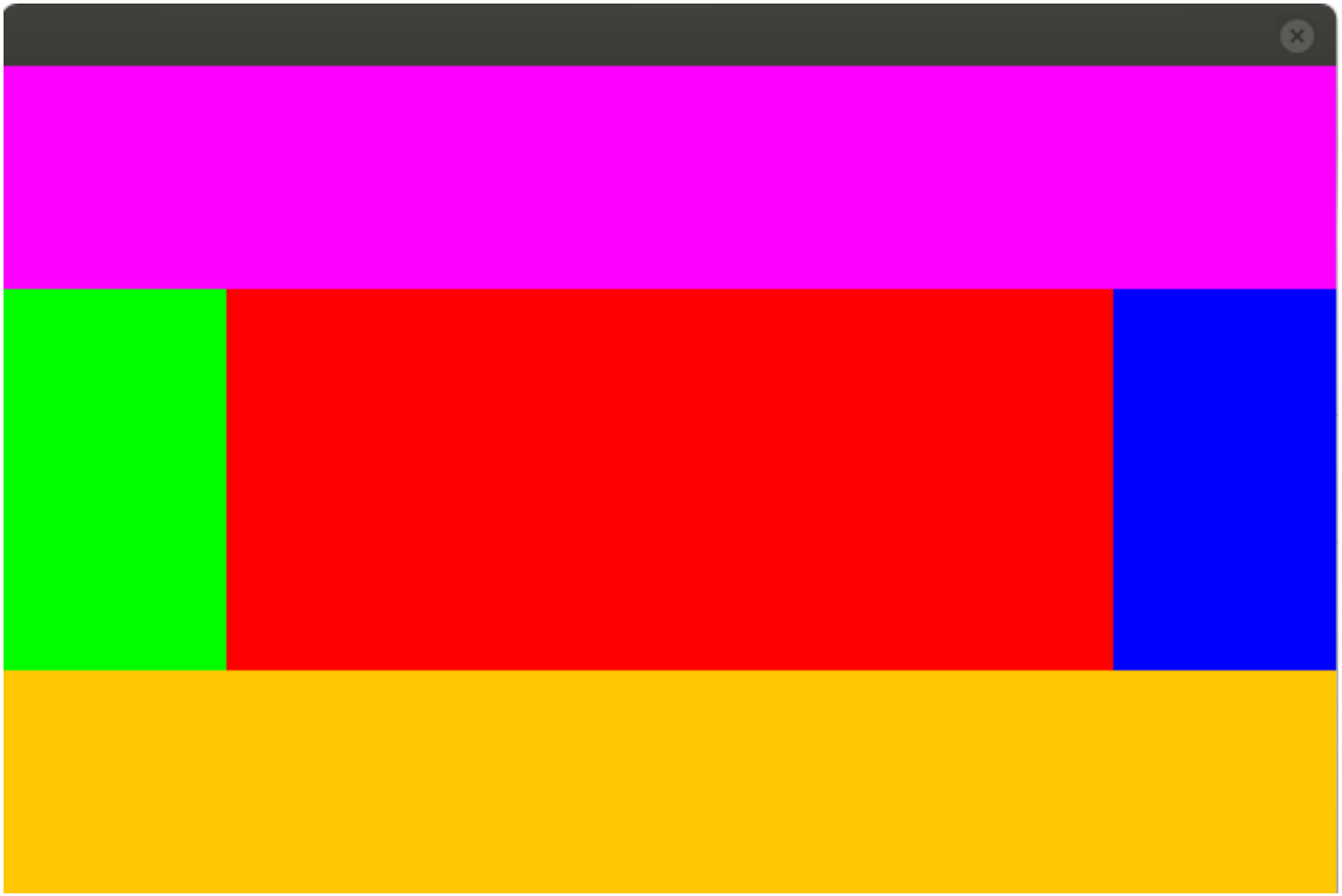
- Admite cinco posições na tela:
  - BorderLayout.CENTER
  - BorderLayout.EAST
  - BorderLayout.WEST
  - BorderLayout.NORTH
  - BorderLayout.SOUTH
- Apenas um componente pode ser adicionado em cada região

java.awt.BorderLayout

```
setLayout(new BorderLayout());
setSize(600, 400);
String[] borders = new String[]{BorderLayout.
    CENTER, BorderLayout.EAST, BorderLayout.WEST,
    BorderLayout.NORTH, BorderLayout.SOUTH};

Color[] colors = new Color[]{Color.red, Color.
    blue, Color.green, Color.magenta, Color.ORANGE
    };
for (int i = 0; i < 5; i++) {
   JPanel panel = new JPanel();
   panel.setBackground(colors[i]);
   panel.setSize(100, 100);
   panel.setPreferredSize(panel.getSize());
   add(panel, borders[i]);
}
```

java.awt.BorderLayout



java.awt.GridLayout

Definição

Organização de componentes em linhas e colunas que formam uma grade

java.awt.GridLayout

Definição

Organização de componentes em linhas e colunas que formam uma grade

java.awt.GridLayout

- Os componentes são adicionados da esquerda para direita e de cima para baixo
- Depois que enche a primeira linha vai para a segunda
- GridLayout(int rows, int cols, int lgap, int cgap)
  - rows: quantidade de linhas que terá a grid
  - cols: quantidade de colunas que terá a grid
  - lgap: espaçamento entre linhas
  - cgap: espaçamento entre colunas

java.awt.GridLayout

- Os componentes são adicionados da esquerda para direita e de cima para baixo
- Depois que enche a primeira linha vai para a segunda
- GridLayout(int rows, int cols, int lgap, int cgap)
  - rows: quantidade de linhas que terá a grid
  - cols: quantidade de colunas que terá a grid
  - lgap: espaçamento entre linhas
  - cgap: espaçamento entre colunas

java.awt.GridLayout

- Os componentes são adicionados da esquerda para direita e de cima para baixo
- Depois que enche a primeira linha vai para a segunda
- GridLayout(int rows, int cols, int lgap, int cgap)
  - rows: quantidade de linhas que terá a grid
  - cols: quantidade de colunas que terá a grid
  - lgap: espaçamento entre linhas
  - cgap: espaçamento entre colunas

java.awt.GridLayout

- Os componentes são adicionados da esquerda para direita e de cima para baixo
- Depois que enche a primeira linha vai para a segunda
- GridLayout(int rows, int cols, int lgap, int cgap)
  - rows: quantidade de linhas que terá a grid
  - cols: quantidade de colunas que terá a grid
  - lgap: espaçamento entre linhas
  - cgap: espaçamento entre colunas

java.awt.GridLayout

- Os componentes são adicionados da esquerda para direita e de cima para baixo
- Depois que enche a primeira linha vai para a segunda
- GridLayout(int rows, int cols, int lgap, int cgap)
  - rows: quantidade de linhas que terá a grid
  - cols: quantidade de colunas que terá a grid
  - lgap: espaçamento entre linhas
  - cgap: espaçamento entre colunas

java.awt.GridLayout

- Os componentes são adicionados da esquerda para direita e de cima para baixo
- Depois que enche a primeira linha vai para a segunda
- GridLayout(int rows, int cols, int lgap, int cgap)
  - rows: quantidade de linhas que terá a grid
  - cols: quantidade de colunas que terá a grid
  - lgap: espaçamento entre linhas
  - cgap: espaçamento entre colunas

java.awt.GridLayout

- Os componentes são adicionados da esquerda para direita e de cima para baixo
- Depois que enche a primeira linha vai para a segunda
- GridLayout(int rows, int cols, int lgap, int cgap)
  - rows: quantidade de linhas que terá a grid
  - cols: quantidade de colunas que terá a grid
  - lgap: espaçamento entre linhas
  - cgap: espaçamento entre colunas

java.awt.GridLayout

- Os componentes são adicionados da esquerda para direita e de cima para baixo
- Depois que enche a primeira linha vai para a segunda
- GridLayout(int rows, int cols, int lgap, int cgap)
  - rows: quantidade de linhas que terá a grid
  - cols: quantidade de colunas que terá a grid
  - lgap: espaçamento entre linhas
  - cgap: espaçamento entre colunas

java.awt.GridLayout

```
setLayout(new GridLayout(0, 2, 3, 3));
setSize(600, 400);
setDefaultCloseOperation(EXIT_ON_CLOSE);

Color[] colors = new Color[]{Color.red, Color.
    blue, Color.green, Color.magenta, Color.ORANGE
    };

for (int i = 0; i < 5; i++) {
  JPanel panel = new JPanel();
  panel.setBackground(colors[i]);
  panel.setSize(100, 100);
  panel.setPreferredSize(panel.getSize());
  add(panel);
}
```

java.awt.GridLayout

Referências Bibliográficas

Aula baseada em slides preparados pelos seguintes professores:

- Prof. Bruno B. Boniati - www.cafw.ufsm.br/ bruno - Universidade Federal de Santa Maria
- Prof. José Gustavo de Souza Paiva - http://www.facom.ufu.br/ jgustavo/disc/poo.html - Faculdade de Computação da Universidade Federal de Uberlândia
- Prof. Vítor E. Silva Souza - http://www.inf.ufes.br/ vitorsouza/pt/ - Departamento de Informática da Universidade Federal do Espírito Santo

Livro:

- Guia de Estudo: Certificação Sun para Programador Java 6. Kathy Sierra e Bert Bates