Layouts Java

Prof. Hugo de Paula

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Departamento de Ciência da Computação

Sumário

Entendendo Layouts

- Note que a disposição dos Components sobre o Container não foi indicada por um par ordenado (x,y).
- *Layout* define a maneira pela qual os componentes irão se distribuir sobre o Container.
- É possível definir seus próprios layouts, mas a linguagem oferece um conjunto de LayoutManagers que facilitam o trabalho.
- Para especificar um LayoutManager para um Container, basta chamar o seguinte método do Container:

```
public void setLayout(LayoutManager manager)
```

Por exemplo:

```
JPanel painel = new JPanel();
painel.setLayout(new BorderLayout());
```

FlowLayout

- FlowLayout: os Components são distribuídos da esquerda para a direita e de cima para baixo.
- Ele é o layout default do JPanel.
- Você pode definir a dimensão dos componentes com o comando panel.setPreferredSize(**new** Dimension(100, 100));
- Os métodos setBounds e setSize só têm efeito quando o LayoutManager é **null**.
- Ele possui três construtores:

```
public FlowLayout();
public FlowLayout(int align);
public FlowLayout(int align, int hgap, int vgap);
```

- *align* pode receber: FlowLayout.LEFT (ou FlowLayout.LEADING), FlowLayout.RIGHT (ou FlowLayout.TRAILING) ou FlowLayout.CENTER.

BorderLayout

- BorderLayout: Divide o Container em 5 áreas: North, South, East, West e Center.
- É o layout default para a maioria das aplicações com janelas (p. ex. JFrame).
- Construtores:

```
public BorderLayout();
public BorderLayout(int hgap, int vgap);
```

- Quando se adiciona um componente é necessário especificar em qual das áreas ele deve ser adicionado.
- Ex:

```
add(butOK, BorderLayout.WEST);
```

GridLayout

- GridLayout é similar ao FlowLayout, mas cada Component é alocado em uma célula de igual tamanho.
- Permite definir um vetor ou matriz de células nas quais os componentes são alocados.
- Construtores:

```
public GridLayout(int rows, int columns);
public GridLayout(int rows, int columns, int hgap, int vgap);
```

- Componentes são alocados na ordem em que são adicionados, da esquerda para a direita, de cima para baixo.
- Se o número de colunas ou o número de linhas é zero, ele é definido pelo quantidade de componentes adicionados. Se ambos os valores são setados, o número de colunas é ignorado.

Outros layouts

- CardLayout: usado para exibir um componente de cada vez como em uma pilha de cartas.
- BoxLayout: empilha os componentes, um em cima do outro, ou em uma fila. Uma espécie de FlowLayout mais flexível.
- GridBagLayout: um layout flexível e complicado usado quando se deseja um layout complexo e com muitos componentes.
- GroupLayout e SpringLayout são extremamente complexos e não devem ser usados manualmente.
- GroupLayout agrupa os componentes hierarquicamente e os grupos podem ser posicionados sequencialmente ou em paralelo.
- SpringLayout permite misturar características de diversos LayoutManagers para compor um novo layout.

Compondo layouts usando painéis

- A classe JPanel é derivada de Container e, por isso, possui seu próprio LayoutManager.
- Pode ser usada para compor interfaces complexas.

```
public class AplicacaoGrafica extends JFrame{
  private JButton butOK;
  private JTextField campo1,campo2,campoR;
  private JLabel texto1,texto2,textoR;
  private JPanel p1 = new JPanel();
  private JPanel p2 = new JPanel();
```

Compondo layouts usando painéis

```
  public AplicacaoGrafica(){
    super("Aplicacao grafica simples");

    // Cria os componentes
    texto1  = new JLabel("Nome:");    campo1  = new JTextField(15);
    texto2  = new JLabel("Fone:");    campo2  = new JTextField(15);
    butOK   = new JButton("OK");
    textoR  = new JLabel("Resp:");    campoR  = new JTextField(20);

    // Define o layout do container básico
    setLayout(new GridLayout(2,1));
    // Define o layout dos Panels
    p1.setLayout(new GridLayout(2,2));
    p2.setLayout(new FlowLayout(FlowLayout.CENTER));

    // Adiciona os componentes aos panels
    p1.add(texto1); p1.add(campo1);
    p1.add(texto2); p1.add(campo2);
    p2.add(butOK);
    p2.add(textoR); p2.add(campoR);

    // Adiciona os panels ao container básico
    getContentPane().add(p1); getContentPane().add(p2);
  }
}
```

Desenhando sobre um componente

- Bibliotecas gráficas: desenha retas, círculos, polígonos etc.
- Método “paint” desenha conteúdo de um Component.
- Graphics é o contexto gráfico do dispositivo onde será desenhado o código de “paint”.
- Para capturar eventos de mouse basta criar um MouseListener e registra-lo junto ao componente desejado.

Desenhando sobre um componente

```
class AreaDeDesenho extends JComponent {
  BufferedImage img;
  Random r = new Random();

  public AreaDeDesenho() {
    img = new  BufferedImage(200, 200, BufferedImage.TYPE_INT_RGB);
    Graphics2D ig2 = img.createGraphics();
    ig2.setBackground(Color.WHITE);
    ig2.clearRect(0, 0, img.getWidth(), img.getHeight());

    this.setPreferredSize(new Dimension(200, 200));
    this.setBackground(Color.WHITE);
    this.setOpaque(true);

    this.addMouseListener(new MouseAdapter() {
      public void mouseClicked(MouseEvent e) {
          Graphics g = img.getGraphics();
          g.setColor(new Color(r.nextInt(256),  r.nextInt(256), r.nextInt(256)));
          g.fillOval(e.getX(), e.getY(), 10, 10);
          g.dispose();
          repaint();
      }
    });
  }
  public void paintComponent(Graphics g) {
        g.drawImage(img, 0, 0, null);
  }
}
```

Desenhando sobre um componente

```
public class AplicacaoCanvas extends JFrame {
   private JButton close = new JButton("Close");
   private AreaDeDesenho ades = new AreaDeDesenho();

   public AplicacaoCanvas() {
      super("Caixa de desenho");

      this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      this.setLayout(new BorderLayout());
      this.getContentPane().add(close, BorderLayout.SOUTH);
      this.getContentPane().add(ades, BorderLayout.CENTER);

      close.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent e) {
            System.exit(0);
         }
      });

      pack();
   }

   public static void main(String args[]) {
      new AplicacaoCanvas().setVisible(true);
   }
}
```

Exercícios

- Altere o programa do exemplo de maneira que o mesmo ligue cada duas bolinhas por uma reta.