GRÁFICOS EM JAVA

Miguel Jonathan – DCC-IM-UFRJ - *rev. junho 2010*

Pode-se desenhar em qualquer "componente", ou seja, em sub-classes das classes abstratas javax.swing.JComponent ou java.awt.Component. As hierarquias que interessam mais são as seguintes:

java.lang.Object
└ java.awt.Component
└ java.awt.Container
└ javax.swing.JComponent
└ **javax.swing.JPanel**

java.lang.Object
└ java.awt.Component
└ **java.awt.Canvas**

java.lang.Object
└ **java.awt.Graphics**
└ **java.awt.Graphics2D**

O melhor é desenhar em uma subclasse de JPanel ou Canvas. É possível desenhar direto em uma subclasse de JFrame, mas não é conveniente, porque o espaço da barra de título precisa ser levado em conta no cálculo das coordenadas y.

Desenhar é simples: dentro da classe do componente deve ser criado um método com assinatura:

```
public void paint(Graphics g)
```

O parâmetro g é uma instância de Graphics ou uma de suas subclasses, que será passada diretamente pelo sistema para o método paint(). Atualmente é passada uma instância de Graphics2D.

Todo o desenho é feito dentro do método paint(), através dos métodos de Graphics chamados a partir do parâmetro g. As coordenadas serão sempre considerando o ponto (0,0) como o canto superior esquerdo do componente, com x avançando para a direita, e y para baixo. Mais adiante serão vistos alguns desses métodos.

ATENÇÃO: O método paint() nunca deve ser chamado diretamente.
Quando o componente é criado, Java chama automaticamente o método void paint(Graphics g) e lhe passa uma instância de Graphics. Depois, caso seja necessário apagar e re-desenhar novamente sobre o componente, deve-se chamar o método repaint() do componente, sem qualquer argumento. Esse método é quem fará com que o método paint() seja chamado assim que for possível. (Note no programa exemplo que o método paint() não é chamado diretamente).

Alguns métodos de instância da classe Graphics são mostrados abaixo.
Consulte a API da classe para ter uma visão mais completa.
Os métodos devem ser chamados a partir da referência a um Graphics, que será o parâmetro do método paint() da classe do componente onde queremos desenhar:

a) Para traçar linhas retas:

```
drawLine(int x1, int y1 int,x2, int y2);
```

Traça uma linha do ponto (x1,y1) ao ponto (x2, y2) do componente.

b) Para desenhar caracteres e strings:

```
drawString (String s, int x, int y);
```

Desenha a string s a partir do ponto (x,y), usando a fonte default usada pela instância de Graphics:

c) Para mudar a fonte das letras, é possível usar o método:

```
setFont(Font f);
```

Uma fonte pode ser criada com o construtor:

```
Font(String nomeDaFonte, int estilo, int tamanho);
```

Veja no programa exemplo uma fonte criada de nome "Serif", estilo Negrito (*bold*), e tamanho 14.

A classe Graphics permite desenhar outras formas, trocar cores, etc. Consulte a API.

A seguir serão mostrados alguns exemplos de uso de Graphics:

1. Desenha 3 Linhas:

O primeiro programa exemplo desenha 3 linhas e uma String sobre um componente que é de uma sub-classe de JPanel. Ele faz o seguinte:

a) Define uma subclasse de JFrame para ser a janela, e uma subclasse de JPanel para ser o componente onde serão feitos os desenhos;
b) A classe do componente contém o método para obter os valores a renderizar, e também o método paint(), onde estão as chamadas aos métodos gráficos do seu parâmetro g. (obs: renderizar é gerar um desenho a partir de um modelo de dados. Por exemplo, gerar linhas a partir dos valores de seus tamanhos).
   O método paint() desenha 3 linhas de tamanhos 100, 200 e 300 pixels, separadas entre si por uma distância de 20 pixels. Leia os comentários para mais informações sobre o que o programa faz.
c) O construtor da janela constrói a instância do componente e o insere na janela com add(<componente>);
d) O método main() cria uma instância da janela, define suas dimensões, e a torná visível.

```
/*Desenha 3 linhas de comprimentos diferentes: 100, 200 e 300 pixels,
* separadas por uma distância de 20 pixels.
* A primeira linha fica 20 pixels abaixo do limite superior da janela.
* As linhas começam sempre na margem esquerda da tela (x=0).
* Após isso a frase "Foram impressas 3 linhas de comprimentos 100, 200 e 300 pixels".
* será escrita, 200 pixels abaixo,50 pixels à direita da margem.
* A fonte das letras será Serif, Negrito, tamanho 14.
* A janela será aberta com as dimensões largura = 500, altura = 400.
* A origem da janela (canto superior esquerdo) está no canto superior esquerdo da tela.
*/

import java.awt.*;  // para Graphics e Font
import javax.swing.*;  // para JPanel e JFrame

class Painel extends JPanel{
   public void paint(Graphics g){
      int compr;  // comprimento da linha
      int alt=0;  // altura da linha
      for(compr=100; compr<=300; compr+=100){
          alt+=20;
          g.drawLine(0,alt,compr,alt);
      }
      Font font = new Font("Serif",Font.BOLD,14);  // cria a fonte para escrever a frase
      g.setFont(font);  // estabelece a fonte que será usada a partir daqui.
      alt+=200;  //  define a altura 200 pixels abaixo da ultima linha desenhada.
      g.drawString("Foram impressas 3 linhas de comprimentos 100, 200 e 300 pixels", 50, alt);
 }
}

class Janela extends JFrame{
   public Janela(String titulo){
       super(titulo);
       setDefaultCloseOperation(EXIT_ON_CLOSE);
       add(new Painel());
   }
}

public class DesenhaTresLinhas{
    public static void main(String[] args){
        Janela f = new Janela("Teste de Linhas");
        f.setSize(500,400);
        f.setVisible(true);
    }
}
```

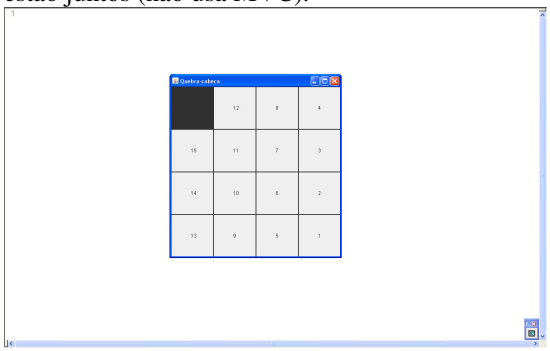
2. Desenho de um Gato:

O segundo exemplo mostra como desenhar formas geométricas, como retângulos e círculos e como preencher com cores, formando figuras construídas livremente:

```
// Exemplo de uso da classe Graphics
// Adaptado do original de Frans Coenen
// Dept. of Comp. Sci., University of Liverpool
import java.awt.*; // Font, Color
import javax.swing.*;

class Painel extends JPanel {

    public void paint(Graphics g) {

      g.setColor(Color.black);
      g.drawRect(50,50,60,60);      // Cabeça
      g.drawRect(80,225,140,5);     // Cauda
      g.setColor(Color.white);
      g.fillOval(20,110,120,120);   // corpo é um circulo branco
      g.setColor(Color.black);
      g.drawOval(20,110,120,120);   // circunferencia do corpo em preto
      g.fillOval(75,75,10,10);      // nariz
      g.setColor(Color.blue);
      g.fillOval(60,60,10,10);      // olhos
      g.fillOval(90,60,10,10);
      g.setColor(Color.black);
      g.drawLine(50,50,60,30);      // orelhas
      g.drawLine(60,30,70,50);
      g.drawLine(110,50,100,30);
      g.drawLine(100,30,90,50);
      g.setColor(Color.red);
      g.drawArc(60,80,40,20,180,180);  // Boca
      g.setColor(Color.black);
      Font serif = new Font("Serif",Font.BOLD,18);
      g.setFont(serif);
      g.drawString("Gato Java",200,50);
   }
}

class Gui extends JFrame{

    public Gui(String text) {
        super(text);
        setBackground(Color.yellow);  // fundo amarelo
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        add(new Painel());
    }
}

class DesenhaGato {

    public static void main(String[] args) {
        Gui gui = new Gui("Gato");
        gui.setSize(300,300);
        gui.setVisible(true);
   }
}
```

3. Jogo Quebra-Cuca

O terceiro exemplo é mais complexo. Mostra um jogo interativo, e precisa usar repaint() para repintar o painel a cada jogada. O jogo usa uma matriz para guardar o estado do jogo (o modelo). Nesse exemplo, o modelo, o controle e a vista estão juntos (não usa MVC).



Além da parte gráfica, o exemplo usa um ouvinte para os eventos de clique do mouse. O ouvinte é um exemplo de uma instância de uma *classe anônima interna* que é uma subclasse de Mouse Adapter. A classe Mouse Adapter é uma classe que implementa MouseListener, com todos os métodos dessa interface com corpos vazios. No exemplo, uma instância

de uma subclasse anônima de MouseAdapter é criada juntamente com a definição do único método redefinido que é mouseReleased().

Os comentários inseridos nos métodos ajudam a compreender o seu funcionamento:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;

class QuebraCuca extends Component{
  int lado;          // tamanho do lado de cada quadrado, em pixels
  int matriz[][];    // a matriz do quebra-cabeca;
  int linVazia,colVazia;   // coordenadas da casa vazia
  Dimension tam;     //
  MouseListener m;

  QuebraCuca(int tamanho){
   tam = new Dimension(tamanho, tamanho);
   lado = tamanho/4; // sao 4 quadrados em cada linha
   matriz = new int[4][4];
  /****
  *   O código abaixo usa uma classe local anônima para gerar o ouvinte de eventos de mouse.
  *   O ouvinte será referenciado pela variável m.
  *   Essa classe é sub-classe de MouseAdapter, e só redefine o método MouseReleased.
  *   Os demais métodos da interface MouseListener continuarão vazios.
  ****/
   m = new MouseAdapter(){
     public void mouseReleased(MouseEvent e){
        int colClic = e.getX()/lado;
        int linClic = e.getY()/lado;
        if (
          colClic < 4 &&
          linClic<4 &&
          colVazia==colClic &&
          (Math.abs(linVazia-linClic)==1) ||
          linVazia == linClic &&
          (Math.abs(colVazia-colClic)==1)
         ){
          matriz[linVazia][colVazia]= matriz[linClic][colClic];
          matriz[linVazia=linClic][colVazia=colClic]=0;
          repaint();
        }
      }
     };
   for (int lin =0;lin<4;lin++)
    for (int col=0; col<4; col++)
      matriz[3-col][3-lin] = 4*lin + col + 1;
   linVazia=colVazia=0;
   matriz[0][0]=0;
   addMouseListener(m);
  } // fim do construtor

  void desenhaQuadr(Graphics g, int lin, int col) {
  // somente o quadrado "vazio" nao será desenhado (e ficará preto)
  // note que a grade não precisa ser desenhada. Apenas nao é preenchida com nada.
    if(matriz[lin][col] !=0){
      g.clearRect(col*lado+1, lin*lado+1, lado-2, lado-2);
      g.drawString(new Integer(matriz[lin][col]).toString(),
                col*lado+ lado/2-4,lin*lado+lado/2+4);
     }
  }

  public void paint(Graphics g) {
    g.fillRect(0,0,lado*4, lado*4); // pinta tudo de preto
    for(int lin=0; lin<4;lin++)
      for(int col=0; col<4; col++)
        desenhaQuadr(g,lin,col);
  }

  public Dimension getPreferredSize() {
     return tam;
  }
  public static void main(String[] args) {
   JFrame f = new JFrame ("Quebra-cabeca");
   f.setSize(400,420);
   f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   f.add(new QuebraCuca(400));
   f.setVisible(true);
  }
```

```
}
```