

# Gráficos em Java (Tópico Especial)

Roland Teodorowitsch

Fundamentos de Programação - Escola Politécnica - PUCRS

24 de agosto de 2022

# Introdução

# Objetivos

- Criar programas em Java que utilizam formas geométricas básicas
- Ter um programa com uma estrutura simples a partir da qual se possa desenhar algumas formas geométricas
- NÃO se pretende aprofundar a discussão sobre as classes usadas para criar e controlar janelas em uma *Graphic User Interface* (GUI)

# Em main

- Inicia-se criando um objeto chamado `frame` da classe `JFrame`
- Um `JFrame` corresponde a uma moldura dentro da qual se podem colocar ou desenhar outros componentes (no exemplo a seguir, a moldura ou janela será de 400 por 400 *pixels*)
- Para este *frame*, define-se então o tamanho (chamada ao método `setSize`) e a operação de fechamento padrão (chamada ao método `setDefaultCloseOperation`)
- Em seguida cria-se um componente (`JComponent`), definindo para este componente um método para desenhar a janela (basicamente este método, que se chama `paintComponent`, chama o método `draw`, que será responsável por desenhar a janela)
- Adiciona-se o componente ao *frame* (chamada de método `add`)
- E, por fim, torna-se o frame *visível* (chamada de método `setVisible`)

# No método `draw`

- É este método que efetivamente desenha as figuras geométricas na janela (e deve ser declarado como `static` para poder ser chamado a partir de `main`)
- O método `draw` tem como parâmetro um objeto da classe `Graphics`
- A classe `Graphics` possui uma série de métodos com os quais se pode desenhar diferentes figuras geométricas (retângulos, elipses, segmentos de reta, etc.)
- Pode-se considerar que os objetos da classe `Graphics` funcionam de forma semelhante a `System.out`, porém desenhando figuras em um *frame* e não textos em um terminal
- No exemplo a seguir, o método `setColor` define a cor padrão de impressão como sendo azul, e `fillRect` é usado para desenhar duas fileiras com quadrados preenchidos de forma alternada

## Exemplo

# Exemplo

- O exemplo a seguir desenha duas fileiras de retângulos alternados em uma janela



# TwoRowsOfSquares.java (HORSTMANN, 2013, p. 180-181)

```
import java.awt.Color;
import java.awt.Graphics;
import javax.swing.JFrame;
import javax.swing.JComponent;

/** This program draws two rows of squares. */
public class TwoRowsOfSquares {

    public static void draw(Graphics g) {
        final int width = 20;
        g.setColor(Color.BLUE);
        // Top row. Note that the top left corner of the drawing has coordinates (0, 0)
        int x = 0;
        int y = 0;
        for (int i = 0; i < 10; i++) {
            g.fillRect(x, y, width, width);
            x = x + 2 * width;
        }
        // Second row, offset from the first one
        x = width;
        y = width;
        for (int i = 0; i < 10; i++) {
            g.fillRect(x, y, width, width);
            x = x + 2 * width;
        }
    }
}
```






# TwoRowsOfSquares.java (HORSTMANN, 2013, p. 180-181)




```
public static void main(String[] args) {  
    // Do not look at the code in the main method  
    // Your code will go into the draw method above  
    JFrame frame = new JFrame();  
    final int FRAME_WIDTH = 400;  
    final int FRAME_HEIGHT = 400;  
    frame.setSize(FRAME_WIDTH, FRAME_HEIGHT);  
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
    JComponent component = new JComponent() {  
        public void paintComponent(Graphics graph) {  
            draw(graph);  
        }  
    };  
    frame.add(component);  
    frame.setVisible(true);  
}
```

## Alguns métodos de Graphics

# Alguns métodos de Graphics (1)

Método	Resultado	Explicação
<code>g.drawRect(x, y, width, height)</code>		$(x, y)$ é o canto superior esquerdo.
<code>g.drawOval(x, y, width, height)</code>		$(x, y)$ é o canto superior esquerdo do retângulo que limita a elipse. Para desenhar um círculo usa-se o mesmo valor para <code>width</code> e <code>height</code> .
<code>g.fillRect(x, y, width, height)</code>		O retângulo é desenhado preenchido.

# Alguns métodos de Graphics (2)

Método	Resultado	Explicação
<code>g.fillOval(x, y, width, height)</code>		A elipse é desenhada preenchida.
<code>g.drawLine(x1, y1, x2, y2)</code>		$(x1, y1)$ e $(x2, y2)$ são os pontos inicial e final de um segmento de reta.
<code>g.drawString("Message", x, y)</code>		$(x, y)$ é o ponto base ( <i>basepoint</i> ).

## Alguns métodos de Graphics (3)

Método	Resultado	Explicação
<code>g.setColor(color)</code>	A partir deste ponto, os métodos para desenhar ou desenhar preenchido usarão a cor selecionada.	Use <code>Color.RED</code> , <code>Color.GREEN</code> , <code>Color.BLUE</code> e assim por diante.

# Exercícios

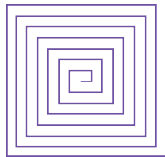
# Exercícios

- 1 Escreva uma aplicação gráfica em Java para desenhar a seguinte face:



Fonte: Horstmann (2013, p. 197)

- 2 Escreva uma aplicação gráfica em Java para desenhar uma espiral retangular como a mostrada na figura a seguir:



Fonte: Horstmann (2013, p. 197)

## Referências



# Referências

HORSTMANN, C. **Java for Everyone – Late Objectt.** 2. ed. Hoboken: Wiley, 2013. xxxiv, 589 p.