


Laboratório de Engenharia de Software

# INF1636 – PROGRAMAÇÃO ORIENTADA A OBJETOS

Departamento de Informática – PUC-Rio

Ivan Mathias Filho  
[ivan@inf.puc-rio.br](mailto:ivan@inf.puc-rio.br)

## Programa – Capítulo 10




Laboratório de Engenharia de Software

- Interfaces Gráficas e Swing
- Componentes e Containers
- Aplicações Gráficas

© LES/PUC-Rio

**Programa – Capítulo 10**




Laboratório de Engenharia de Software

- **Interfaces Gráficas e Swing**
- Componentes e Containers
- Aplicações Gráficas

© LES/PUC-Rio


**Modelos de interface**




Laboratório de Engenharia de Software

- JDK 1.0
  - AWT - *Abstract Window Toolkit*;
  - Tinha por objetivo a independência de plataforma;
  - Delegava aos toolkits das plataformas nativas a criação e o comportamento dos componentes de interface gráfica;
  - Modelo de eventos pobre e ineficiente;
  - Incompatibilidades de apresentação entre as diversas plataformas ("Write Once, Debug Everywhere!!!").
- JDK 1.1
  - Novo e mais eficiente modelo de eventos;
  - Introdução do Swing/JFC - *Java Foundation Classes*;
  - Problemas de compatibilidade não resolvidos.

© LES/PUC-Rio

JFC - Java Foundation Classes		
Laboratório de Engenharia de Software	<ul style="list-style-type: none"><li>• A JFC é um conjunto de pacotes usados para criação de interfaces gráficas com o usuário (GUI)</li><li>• Características da JFC:<ul style="list-style-type: none"><li>– Componentes de interface gráfica do Swing;</li><li>– Suporte a diversos <i>look &amp; feel</i>;</li><li>– Suporte a usuários com deficiências;</li><li>– Suporte a <i>drag &amp; drop</i>;</li><li>– Suporte a aplicações gráficas sofisticadas – Java 2D.</li></ul></li></ul>	
	© LES/PUC-Rio	

O Que é o Swing?		
Laboratório de Engenharia de Software	<ul style="list-style-type: none"><li>• O Swing é basicamente uma coleção de componentes gráficos adaptáveis;</li><li>• Utiliza componentes <i>lightweight</i>:<ul style="list-style-type: none"><li>– Não dependem do código nativo da plataforma alvo;</li><li>– Possuem comportamento similar nas diversas plataformas suportadas;</li><li>– Possuem <i>look &amp; feel</i> configurável;</li><li>– Conjunto reduzido de <i>top level containers</i>.</li></ul></li></ul>	
	© LES/PUC-Rio	

**LES**

- Laboratório de Engenharia de Software

© LES/PUC-Rio

**LES**



©1999 Allen Holtz

© LES/PUC-Rio

Programa – Capítulo 10




Laboratório de Engenharia de Software

- Interfaces Gráficas e Swing
- **Componentes e Containers**
- Aplicações Gráficas

© LES/PUC-Rio


Componentes e *Containers* (1)




Laboratório de Engenharia de Software

- Os elementos de interface com o usuário são classificados em componentes e *containers*;
- **Componente**
  - Define um componente de interface, como um botão, uma caixa de texto ou uma *combo box*;
  - Métodos como `paint()` e `repaint()`;
- **Container**
  - Define um componente que pode conter outros componentes;
  - Define métodos, como `add()`, para adicionar componentes que serão gerenciados pelo container;
  - Possui um gerenciador de layout.


© LES/PUC-Rio

Componentes e Containers (2)		
Laboratório de Engenharia de Software	<ul style="list-style-type: none"><li>• <b>Janela</b><ul style="list-style-type: none"><li>– É o <i>container</i> de mais alto nível (<i>top level container</i>);</li><li>– Existe para prover espaço para apresentação dos componentes Swing.</li></ul></li><li>• <b>Painel</b><ul style="list-style-type: none"><li>– É um container intermediário;</li><li>– Existe para controlar o posicionamento dos componentes.</li></ul></li><li>• Componentes atômicos, como botões e caixas de texto, realizam a interação com o usuário propriamente dita.</li></ul>	
	© LES/PUC-Rio	

Janela - JFrame		
Laboratório de Engenharia de Software	<ul style="list-style-type: none"><li>• Representa uma janela Swing;</li><li>• Seu construtor pode receber uma string com o título da janela;</li><li>• Possui um painel invisível, chamado <b>ContentPane</b>;</li><li>• Os componentes da janela são inseridos neste painel;</li><li>• Pode conter uma barra de menu.</li></ul>	
	© LES/PUC-Rio	

Laboratório de Engenharia de Software

## JFrame – Principais Construtores




- `JFrame(String title)`
- `JFrame()`

© LES/PUC-Rio

Laboratório de Engenharia de Software

## JFrame – Métodos importantes (1)



- `void setBounds(int x,int y,int width,int height)`
- `void setSize(int width,int height)`
- `void setLocation(int x,int y)`
- `void setVisible(boolean b)`
- `void setTitle(String title)`
- `Container getContentPane()`
- `void setJMenuBar(JMenuBar menubar)`
- `JMenuBar getJMenuBar()`
- `void setResizable(boolean resizable)`

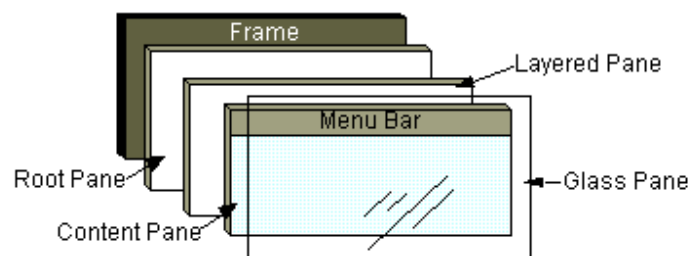
© LES/PUC-Rio

## JFrame – Métodos importantes (2)



- `void setDefaultCloseOperation(int op)`
  - Valores válidos para `op`:
    - `EXIT_ON_CLOSE`
    - `HIDE_ON_CLOSE`
    - `DISPOSE_ON_CLOSE`
    - `DO_NOTHING_ON_CLOSE`


## Estrutura de um JFrame





Laboratório de Engenharia de Software

## Camadas do JFrame




- **RootPane**
  - Gerencia as demais camadas.
- **LayeredPane**
  - Contém a *menu bar* e o *ContentPane*;
  - Pode conter várias subcamadas.
- **ContentPane**
  - Contém os componentes visíveis.
- **GlassPane**
  - invisível por *default*;
  - interceptação de eventos/pintura sobre uma região.

© LES/PUC-Rio

Laboratório de Engenharia de Software

## JFrame – Exemplo (1)



```
import javax.swing.*;

public class PrimFrame extends JFrame {
    public final int LARG_DEFAULT=400;
    public final int ALT_DEFAULT=300;

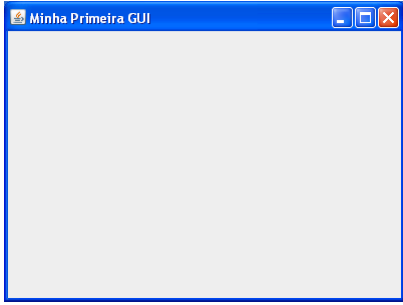
    public PrimFrame() {
        setSize(LARG_DEFAULT,ALT_DEFAULT);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }
}

public class EX1001 {
    public static void main(String[] args) {
        PrimFrame f=new PrimFrame();
        f.setTitle("Minha Primeira GUI");
        f.setVisible(true);
    }
}
```

© LES/PUC-Rio

## JFrame – Exemplo (2)

Laboratório de Engenharia de Software



© LES/PUC-Rio

## JFrame – Implementação Alternativa

Laboratório de Engenharia de Software

```
import javax.swing.*;

public class PrimFrame extends JFrame {
    public final int LARG_DEFAULT=400;
    public final int ALT_DEFAULT=300;

    public PrimFrame() {
        setSize(LARG_DEFAULT,ALT_DEFAULT);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }

    public static void main(String[] args) {
        PrimFrame f=new PrimFrame();

        f.setTitle("Minha Primeira GUI");
        f.setVisible(true);
    }
}
```

© LES/PUC-Rio

## JFrame – Posicionamento



- Por *default*, um frame é posicionado a partir do *pixel* localizado no canto superior esquerdo (0,0);
- Podemos posicionar um frame a partir de outras coordenadas por meio dos métodos `setBounds()` e `setLocation()`;
- É possível posicionar uma janela levando-se em conta a resolução do monitor do computador;
- Para tal, deve-se usar as classes `Toolkit` e `Dimension`, definidas no pacote `java.awt`.

© LES/PUC-Rio

## JFrame – Exemplo de Posicionamento



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

public class PrimFrame extends JFrame {
    public final int LARG_DEFAULT=400;
    public final int ALT_DEFAULT=300;

    public PrimFrame() {
        Toolkit tk=Toolkit.getDefaultToolkit();
        Dimension screenSize=tk.getScreenSize();
        int sl=screenSize.width;
        int sa=screenSize.height;
        int x=sl/2-LARG_DEFAULT/2;
        int y=sa/2-ALT_DEFAULT/2;

        setBounds(x,y,LARG_DEFAULT,ALT_DEFAULT);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }
}
```

© LES/PUC-Rio

## JPanel



- Representa um painel simples;
- É o container intermediário mais simples;
- Pode ser inserido em uma janela ou em outro painel;
- Para inseri-lo em uma janela (JFrame), deve-se fazê-lo por meio do método `getContentPane()`;
- Após obter o **Content Pane**, o método `add()` deve ser usado para inserir o painel na janela.

© LES/PUC-Rio

## JPanel – Exemplo (1)



```
import javax.swing.*;

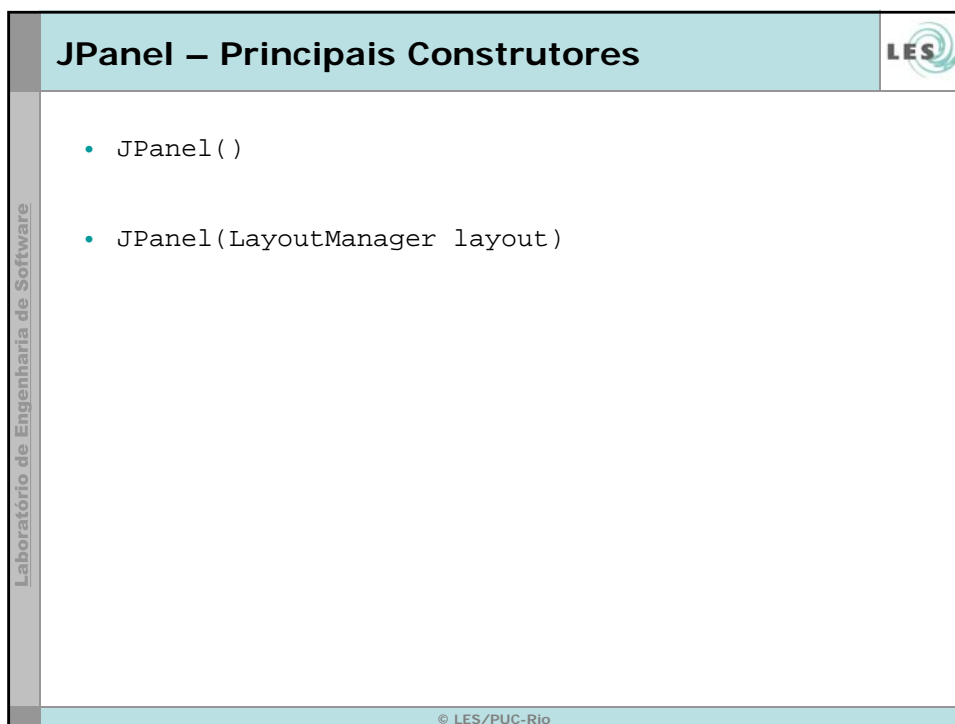
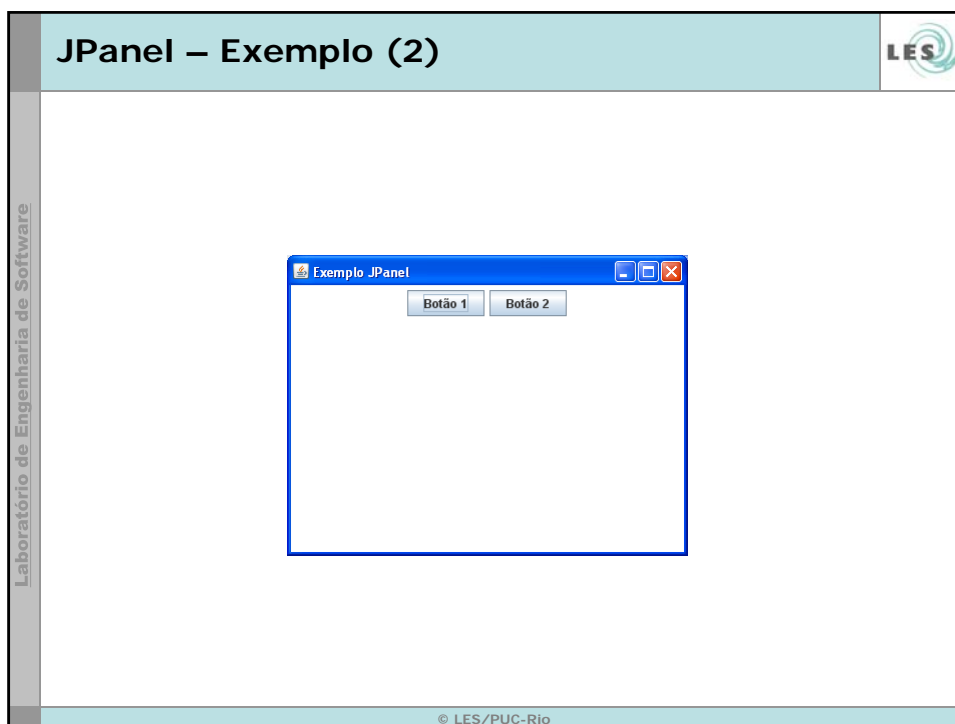
public class EX07Frame extends JFrame {
    JButton b1 = new JButton("Botão 1");
    JButton b2 = new JButton("Botão 2");
    JPanel p = new JPanel();


    public EX07Frame(String s) {
        super(s);
        p.add(b1);
        p.add(b2);
        p.setBackground(Color.WHITE);
        getContentPane().add(p);
        setSize(400,300);
    }

    public static void main(String[] args) {
        EX07Frame f=new EX07Frame("Exemplo JPanel");

        f.setVisible(true);
    }
}
```

© LES/PUC-Rio



Laboratório de Engenharia de Software	<b>JPanel – Métodos importantes</b>	
	<ul style="list-style-type: none"><li>• <code>void setSize(int width,int height)</code></li><li>• <code>void setLayout(LayoutManager layout)</code></li><li>• <code>Component add(Component comp)</code></li><li>• <code>void add(Component c,Object constraints)</code></li><li>• <code>void setEnabled(boolean b)</code></li><li>• <code>Void setBackground(Color c)</code></li><li>• <code>void paintComponent(Graphics g)</code></li><li>• <code>void repaint()</code></li></ul>	

© LES/PUC-Rio

Laboratório de Engenharia de Software	<b>Programa – Capítulo 10</b>	
	<ul style="list-style-type: none"><li>• Interfaces Gráficas e Swing</li><li>• Componentes e Containers</li><li>• <b>Aplicações Gráficas</b></li></ul>	

© LES/PUC-Rio

## A Classe Graphics (1)



Laboratório de Engenharia de Software

- Um painel é um componente visual que possui uma superfície sobre a qual se pode desenhar;
- Para tal deve-se usar objetos da classe `Graphics`;
- Eles possuem métodos para desenhar figuras, texto e imagens.
- Além disso, eles guardam informações como o tipo de fonte usado, a cor de fundo e a cor do desenho;
- Todos os desenhos em Java são feitos por meio de objetos da classe `Graphics`;

© LES/PUC-Rio

## A Classe Graphics (2)



Laboratório de Engenharia de Software

- As medidas dos objetos desenhados são definidas em pixels;
- A coordenada (0,0) se refere ao canto superior esquerdo do componente sobre o qual deseja-se desenhar.

© LES/PUC-Rio

## O Método `paintComponent()`



- Todas as vezes que uma janela precisar ser redesenhada o tratador de eventos irá gerar uma notificação;
- Isso faz com que o método `paintComponent()` de cada um dos componentes seja executado;
- Não se deve chamar esse método explicitamente, pois ele é chamado automaticamente;
- Quando for preciso forçar uma operação de repintagem, deve-se chamar o método `repaint()`;
- O método `repaint()` irá chamar o método `paintComponent()` para cada um dos componentes.

© LES/PUC-Rio

## Componente Gráfico – Exemplo




- O exemplo a seguir irá exibir um texto sobre a superfície de um painel;
- Para tal, o método `paintComponent()` do painel será sobrescrito;
- A exibição de texto é considerada um tipo de desenho;
- O método `drawString()` é usado para exibir texto sobre a superfície de um painel.

© LES/PUC-Rio



## Graphics – Exemplo (1)



Laboratório de Engenharia de Software

```

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

public class EX04Panel extends JPanel {
    public static final int TXT_X=120;
    public static final int TXT_Y=140;


    public void paintComponent(Graphics g) {
        super.paintComponent(g);

        g.drawString("Primeiro Programa Gráfico",TXT_X,TXT_Y);
    }
}

```

© LES/PUC-Rio

## Graphics – Exemplo (2)



Laboratório de Engenharia de Software

```

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

public class EX04Frame extends JFrame {
    public final int LARG_DEFAULT=400;
    public final int ALT_DEFAULT=300;

    public EX04Frame() {
        Toolkit tk=Toolkit.getDefaultToolkit();
        Dimension screenSize=tk.getScreenSize();
        int sl=screenSize.width;
        int sa=screenSize.height;
        int x=sl/2-LARG_DEFAULT/2;
        int y=sa/2-ALT_DEFAULT/2;


        setBounds(x,y,LARG_DEFAULT,ALT_DEFAULT);
        setDefaultCloseOperation(EXIT_ON_CLOSE);

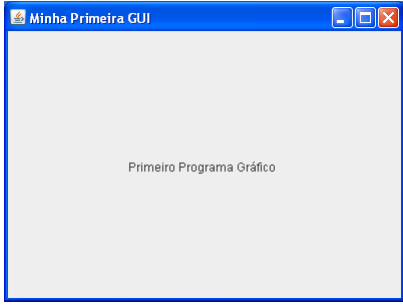
        getContentPane().add(new EX04Panel());
    }
}

```

© LES/PUC-Rio


## Graphics – Exemplo (3)

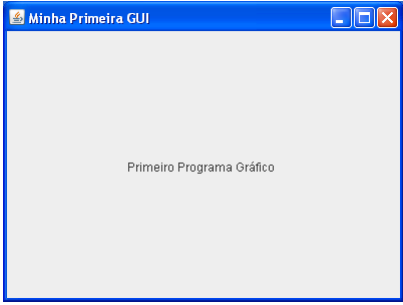




© LES/PUC-Rio

## A API Java 2D (1)





- A biblioteca Java 2D foi introduzida na versão 1.2 da linguagem Java;
- Ela organiza as formas geométricas por meio de objetos;
- Para utilizar a Java 2D é necessário um objeto da classe Graphics2D;
- A partir do Java 1.2 objetos da classe Graphics2D são passados no lugar de objetos da classe Graphics.

```

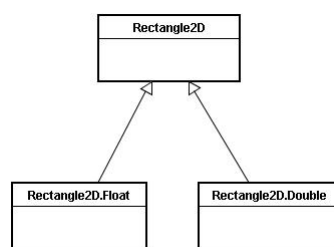
public void paintComponent(Graphics g) {
    super.paintComponent(g);
    Graphics2D g2d=(Graphics2D) g;
    . . .
}
    
```

© LES/PUC-Rio

## A API Java 2D (2)

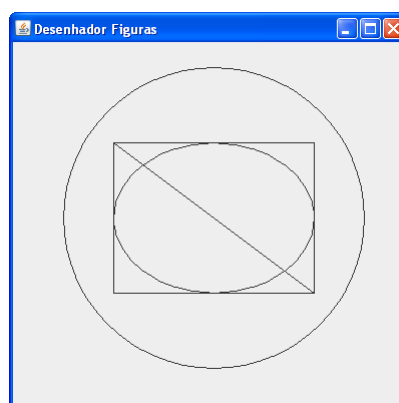


- A Java 2D utiliza números em ponto flutuante para definir as coordenadas gráficas das formas geométricas;
- Para evitar a ocorrência de erros na conversão de tipos, duas versões de cada classe (float e double) são fornecidas:




© LES/PUC-Rio

## Java 2D – Exemplo (1)



© LES/PUC-Rio

## Java 2D – Exemplo (2)



Laboratório de Engenharia de Software


```
import javax.swing.JPanel;
import java.awt.*;
import java.awt.geom.*;

public class ExPanel extends JPanel {
    public static final int TXT_X=120;
    public static final int TXT_Y=140;

    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics2D g2d=(Graphics2D) g;
```

© LES/PUC-Rio

## Java 2D – Exemplo (3)



Laboratório de Engenharia de Software

```
// Desenha retângulo
double leftX=100.0;
double topY=100.0;
double larg=200.0;
double alt=150.0;

Rectangle2D rt=new Rectangle2D.Double(leftX,topY,larg,alt);

g2d.draw(rt);

// Desenha a elipse interna ao retângulo
Ellipse2D e=new Ellipse2D.Double();
e setFrame(rt);
g2d.draw(e);
```

© LES/PUC-Rio

## Java 2D – Exemplo (4)



```
// Desenha uma diagonal do retângulo
Point2D p1=new Point2D.Double(leftX,topY);
Point2D p2=new Point2D.Double(leftX+larg,topY+alt);
g2d.draw(new Line2D.Double(p1,p2));

// Desenha círculo com o mesmo centro
double cX=rt.getCenterX();
double cY=rt.getCenterY();
double raio=150.0;

Ellipse2D circ=new Ellipse2D.Double();
circ.setFrameFromCenter(cX,cY,cX+raio,cY+raio);
g2d.draw(circ);
}
```

## Uso de Cores

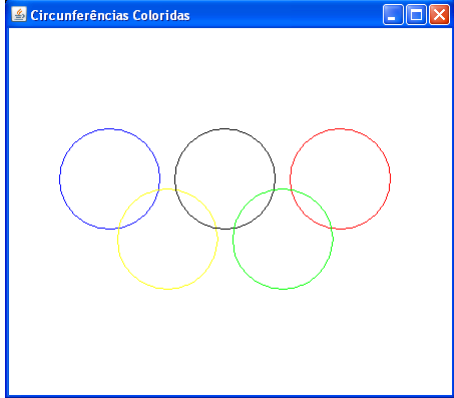


- O método `setPaint()` da classe `Graphics2D` permite definir a cor que será usada no próximo desenho;
- Caso se deseje alterar a cor em um desenho subsequente deve-se executar novamente o método `setPaint()`;
- A classe `java.awt.Color` é usada para definir a cor de um desenho;
- Ela contém várias cores pré-definidas, acessíveis por meio de variáveis estáticas.

## Uso de Cores – Exemplo (1)

Laboratório de Engenharia de Software

LES



© LES/PUC-Rio

## Uso de Cores – Exemplo (2)

Laboratório de Engenharia de Software

LES


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

public class ExPanel extends JPanel {
    public static final int TXT_X=120;
    public static final int TXT_Y=140;

    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics2D g2d=(Graphics2D) g;
        Ellipse2D circ;
```

© LES/PUC-Rio

**Uso de Cores – Exemplo (3)**



Laboratório de Engenharia de Software


```
// Desenha o círculo AZUL
circ=new Ellipse2D.Double(50,100,100,100);
g2d.setPaint(Color.BLUE);
g2d.draw(circ);

// Desenha o círculo AMARELO
circ=new Ellipse2D.Double(107.5,160,100,100);
g2d.setPaint(Color.YELLOW);
g2d.draw(circ);

// Desenha o círculo PRETO
circ=new Ellipse2D.Double(165,100,100,100);
g2d.setPaint(Color.BLACK);
g2d.draw(circ);
```

© LES/PUC-Rio

**Uso de Cores – Exemplo (4)**



Laboratório de Engenharia de Software

```
// Desenha o círculo VERMELHO
circ=new Ellipse2D.Double(280,100,100,100);
g2d.setPaint(Color.RED);
g2d.draw(circ);

// Desenha o círculo VERDE
circ=new Ellipse2D.Double(222.5,160,100,100);
g2d.setPaint(Color.GREEN);
g2d.draw(circ);

}
```

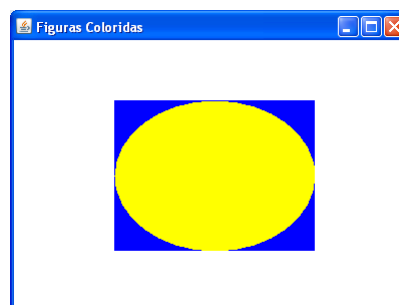
© LES/PUC-Rio

## Figuras Preenchidas



- É possível pintar o interior de formas geométricas com cores específicas;
- Para tal, deve-se usar o método `fill()` em vez do método `draw()`;
- A definição da cor de preenchimento deve ser feita da mesma maneira utilizada no exemplo anterior.

## Figuras Preenchidas – Exemplo (1)





## Figuras Preenchidas – Exemplo (2)



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

public class ExPanel extends JPanel {
    public static final int TXT_X=120;
    public static final int TXT_Y=140;

    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics2D g2d=(Graphics2D) g;
        Ellipse2D circ;
```

## Figuras Preenchidas – Exemplo (3)



```
// Desenha retângulo
double leftX=100.0;
double topY=60.0;
double larg=200.0;
double alt=150.0;

Rectangle2D rt=new
Rectangle2D.Double(leftX,topY,larg,alt);
g2d.setPaint(Color.BLUE);
g2d.fill(rt);

// Desenha a elipse interna ao retângulo
Ellipse2D e=new Ellipse2D.Double();
e.setFrame(rt);
g2d.setPaint(Color.YELLOW);
g2d.fill(e);
}
```

## Trabalhando com Imagens (1)



- É possível carregar uma imagem para posterior exibição em um objeto Graphics;
- A imagem precisa ser carregada de um arquivo local ou de uma URL na Internet;
- A carga da imagem é feita por meio do método `ImageIO.read()`;
- A exibição de uma imagem já carregada deve ser feita por meio do método `drawImage()` da classe Graphics;

## Trabalhando com Imagens (2)



- A carga de uma imagem pode levantar uma exceção do tipo `IOException`;
- Dessa forma, a carga deve ser feita dentro de um bloco `try...catch`.

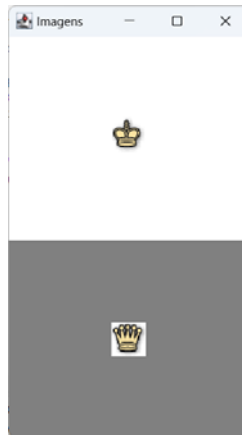
```
Image i;

try {
    i=ImageIO.read(new File("b_dama.gif"));
}
catch(IOException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
```

## Exemplo (1)



- Exibição, em dois painéis, das imagens de um rei e uma rainha usados no jogo de xadrez.



© LES/PUC-Rio

## Exemplo (2)



- A primeira classe é a que define o frame.

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

public class EX10Frame extends JFrame {
    JPanel p;

    public EX10Frame(String s, Image vet[]) {
        super(s);
        setLayout(null);

        p = new JPanel(vet[0]);
        p.setBackground(Color.WHITE);
        p.setBounds(0, 0, 400, 200);
        getContentPane().add(p);

        p = new JPanel(vet[1]);
        p.setBackground(Color.GRAY);
        p.setBounds(0, 200, 400, 200);
        getContentPane().add(p);
        setSize(250, 430);
    }
}
```

© LES/PUC-Rio

### Exemplo (3)



- A segunda classe é a que define o painel em que serão exibidas as imagens.

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

public class ExPanel extends JPanel {
    Image img;
    public ExPanel(Image i) {
        img=i;
    }

    public void paintComponent(Graphics g) {
        super.paintComponent(g);

        g.drawImage(img,100,80,null);
    }
}
```

© LES/PUC-Rio

### Exemplo (4)



- A terceira classe é a que contém o método main;
- Ela é responsável por ler as imagens que serão exibidas.

```
import java.awt.Image;
import java.io.*;
import javax.imageio.ImageIO;

public class EX1010 {
    public static void main(String[] args) {
        EX10Frame f;
        Image []vet=new Image[2];

        for(int i=0; i<2; i++)
            try {
                vet[i]=ImageIO.read(new File(i+".gif"));
            }
            catch(IOException e) {
                System.out.println(e.getMessage());
                System.exit(1);
            }
        f=new EX10Frame("Imagens",vet);
        f.setVisible(true);
    }
}
```

© LES/PUC-Rio