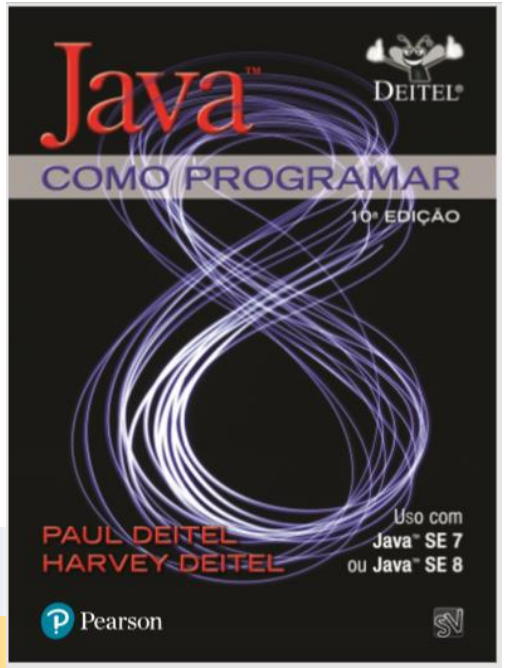


Guia de Laboratório

Lab 03

Prof. André Bernardi

andrebernardi@unifei.edu.br

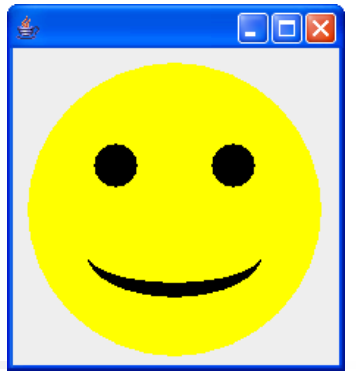
3º Laboratório ECOP15 – 20 e 21/09/2021

- **Utilizar como referência os exemplos:**
  - ***ArcsJPanel.java*** – desenhar arcos utilizando drawArc() e fillArc();
  - **DrawArcs.java** – adicionar arcos a um jFrame;
  - **PolygonsJPanel.java** – desenhar polígonos utilizando drawPolygon() e fillPolygon();
  - **DrawPolygons.java** – adicionar polígonos a um JFrame;

3º Laboratório ECOP15 – 20 e 21/09/2021

- **LinesRectsOvalsJPanel.java** – desenhar linhas utilizando drawLine();
- **LinesRectsOvals.java** – adicionar linhas a um jFrame;
- **ShowColors2Frame.java** – Como utilizar JColorChooser;
- **ShowColors2.java** – JFrame com JColorChooser;
- **Shapes2JPanel.java** – Criando formas com GeneralPath;
- **Shapes2** – JFrame com GeneralPath;

1ª Tarefa:



- Escreva um programa utilizando janela em Java para desenhar a figura ao lado. Crie um *Timer* para alternar o desenho dando a impressão que o **Smile** está piscando. Isto pode ser conseguido desenhando ou não o circulo preto que representa seu olho.

Exemplo - DrawSmiley

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

public class DrawSmiley extends JPanel
{
   boolean desenha = true;

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

      // draw the face
      g.setColor( Color.YELLOW );
      g.fillOval( 10, 10, 200, 200 );

      // draw the eyes
      g.setColor( Color.BLACK );
      g.fillOval( 55, 65, 30, 30 );
      if(desenha)
        g.fillOval( 135, 65, 30, 30 );

     // draw the mouth
      g.fillOval( 50, 110, 120, 60 );

   // "touch up" the mouth into a smile
      g.setColor( Color.YELLOW );
      g.fillRect( 50, 110, 120, 30 );
      g.fillOval( 50, 120, 120, 40 );
   } // end method paintComponent
```

Exemplo - DrawSmiley

```
    // construtor de DrawSmiley para inicializar o Timer
   public DrawSmiley()
   {
      Timer t = new Timer (500, new ActionListener()
         {
            public void actionPerformed(ActionEvent e)
            {
               desenha = !desenha;
               repaint();
            }
         } );
      t.start();
   }

} // end class DrawSmiley
```

Exemplo - DrawSmiley

```
// Aplicativo de teste que exibe um rosto sorridente.
import javax.swing.JFrame;

public class DrawSmileyTest
{
        public static void main(String[] args)
        {
                DrawSmiley panel = new DrawSmiley();
                JFrame janela = new JFrame();
                janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                janela.add(panel);
                janela.setSize(230, 250);
                janela.setVisible(true);
        }
 } // fim da classe DrawSmileyTest
```

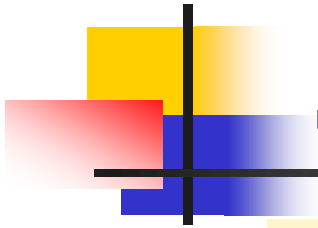
2ª Tarefa:

- Escreva um programa que utilize o método *drawPolyline* para desenhar uma espiral. Faça uma animação usando um Timer para representar a espiral girando.

Lembre-se da equação paramétrica da espiral para configurar os pontos dos vetores a serem usados na polyline.

```java
// Atividade 2
// desenha espiral girando na tela.
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

public class DrawEspiral extends JPanel
{
   int angulo = 0;
   int [] pontosX, pontosY;

   public DrawEspiral()
   {
      //inicializar pontos da espiral.
      pontosX = new int[ 3*360 ]; // três voltas
      pontosY = new int[ pontosX.length ]; // três voltas

      //calcular a espiral centrada em 0,0
      for (int i = 0; i< pontosX.length; i++)
      {
         pontosX[i] = (int)( 0.2*i * Math.cos( i*Math.PI/180) );
         pontosY[i] = (int)( 0.2*i * Math.sin( i*Math.PI/180) );
      }
```

```
      // cria o timer para fazer a espiral girar
      Timer t = new Timer (200, new ActionListener()
         {
            public void actionPerformed(ActionEvent e)
            {
               angulo+= 5;
               angulo %= 360;
               repaint();
            }
         } );
      t.start();
   }
   public void paintComponent( Graphics g )
   {
      super.paintComponent( g );

      Graphics2D g2d = (Graphics2D) g; // casting g para Graphics2D
      g2d.translate( getWidth()/2, getHeight()/2); // altera o eixo para o centro do painel.
      g2d.rotate( angulo * Math.PI/180); // rotaciona o eixo de "angulo" graus
      g2d.setPaint( Color.RED ); //mudar a cor da linha
      g2d.drawPolyline(pontosX, pontosY, pontosX.length); // desenha a espiral
   } // fim do método paintComponent
} // fim da classe DrawEspiral
```

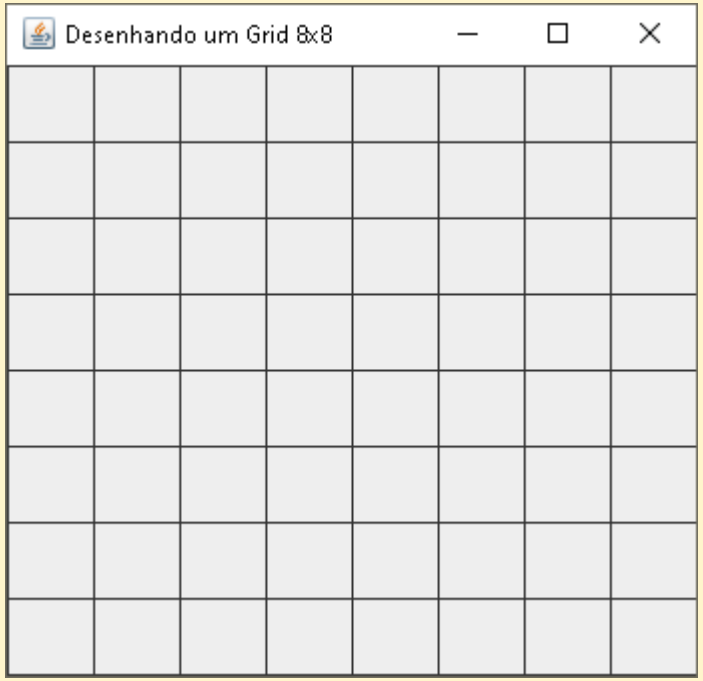


3ª Tarefa:

- Escreva um programa que desenhe no fundo da janela um grid de 8x8, utilizando o método drawLine.

Utilize os métodos getWidth() e getHeight() da classe JPanel para obter o tamanho da área a ser dividida.

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

public class DrawGrid extends JPanel
{
   public final int quantidade = 8;
   public void paintComponent( Graphics g )
   {
     super.paintComponent( g );
     int passoX = getWidth()/quantidade;
     int passoY = getHeight()/quantidade;

     for(int i = 0; i<= quantidade; i++)
     {
        g.drawLine( 0, i*passoY, getWidth(), i*passoY); // linhas horizontais
        g.drawLine( i*passoX, 0, i*passoX, getHeight() ); // linhas verticais
     }
   } // fim do método paintComponent
} // fim da classe DrawGrid
```

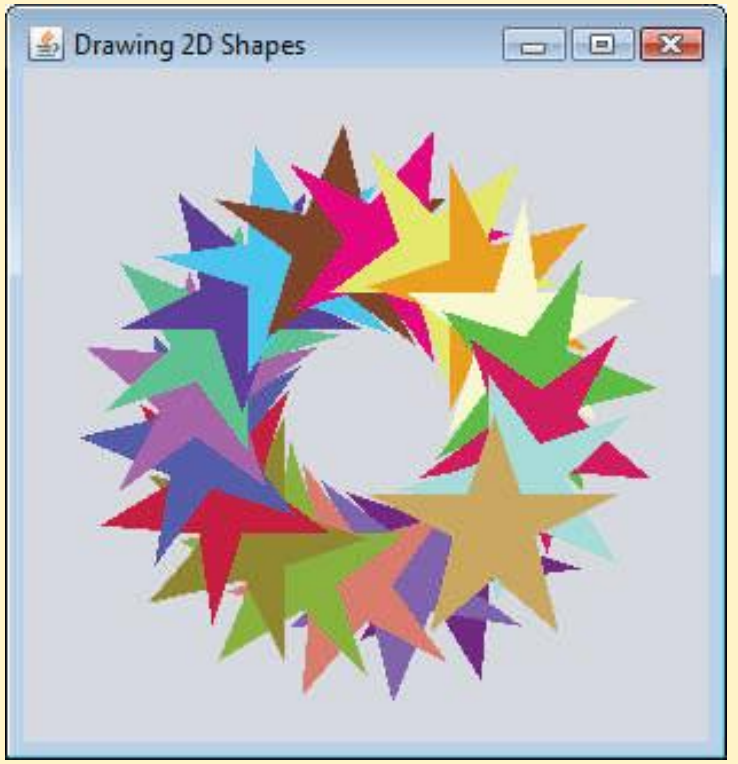


4ª Tarefa:

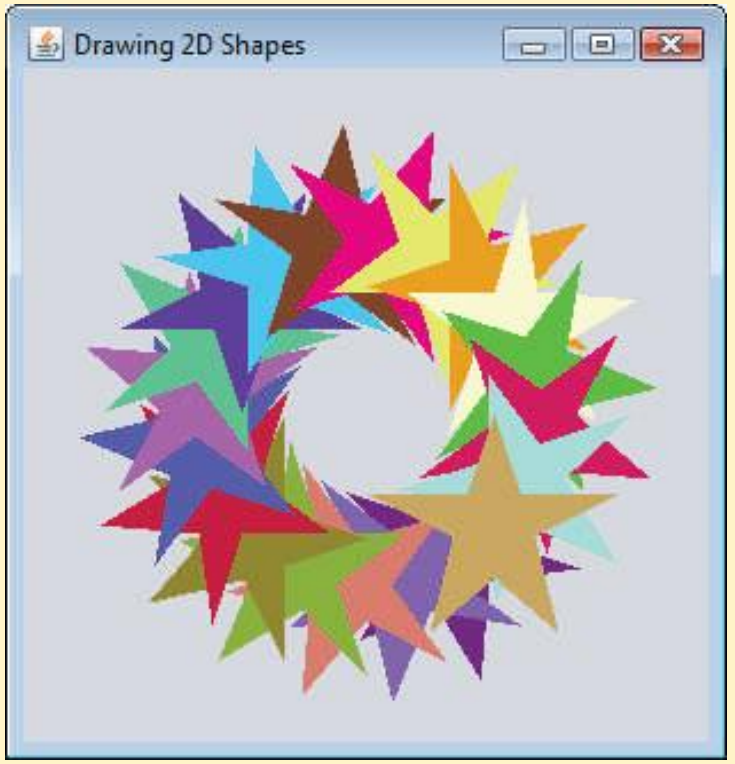
- Escreva um programa que mostre em posições aleatórias do fundo da janela triângulos. Utilize a classe **GeneralPath** e o método **fill** da classe **Graphics2D** para desenhar os triângulos.

Utilize o exemplo 13.31 como modelo.

```java
// Fig. 13.31: Shapes2JPanel.java
// Demonstrando um caminho geral.
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.geom.GeneralPath;
import java.util.Random;
import javax.swing.JPanel;

public class Shapes2JPanel extends JPanel
{
   // desenha caminhos gerais
   public void paintComponent( Graphics g )
   {
      super.paintComponent( g );          // chama o paintComponent da superclasse
      Random random = new Random();       // obtém o gerador de números aleatórios

      int xPoints[] = { 55, 67, 109, 73, 83, 55, 27, 37, 1, 43 };
      int yPoints[] = { 0, 36, 36, 54, 96, 72, 96, 54, 36, 36 };

      Graphics2D g2d = ( Graphics2D ) g;
      GeneralPath star = new GeneralPath(); // cria o objeto GeneralPath
```



```
      // configura a coordenada inicial do General Path
      star.moveTo( xPoints[ 0 ], yPoints[ 0 ] );

      // cria a estrela -- isso não desenha a estrela
      for ( int count = 1; count < xPoints.length; count++ )
         star.lineTo( xPoints[ count ], yPoints[ count ] );

      star.closePath(); // fecha a forma

      g2d.translate( 200, 200 ); // translada a origem para

      // gira em torno da origem e desenha estrelas em cores
      for ( int count = 1; count <= 20; count++ )
      {
         g2d.rotate( Math.PI / 10.0 ); // rotaciona o sistema de coordenadas

         // configura cores aleatórias
         g2d.setColor( new Color( random.nextInt( 256 ),
            random.nextInt( 256 ), random.nextInt( 256 ) ) );

         g2d.fill( star );              // desenha estrela preenchida
      } // for final
   } // fim do método paintComponent
} // fim da classe Shapes2JPanel
```

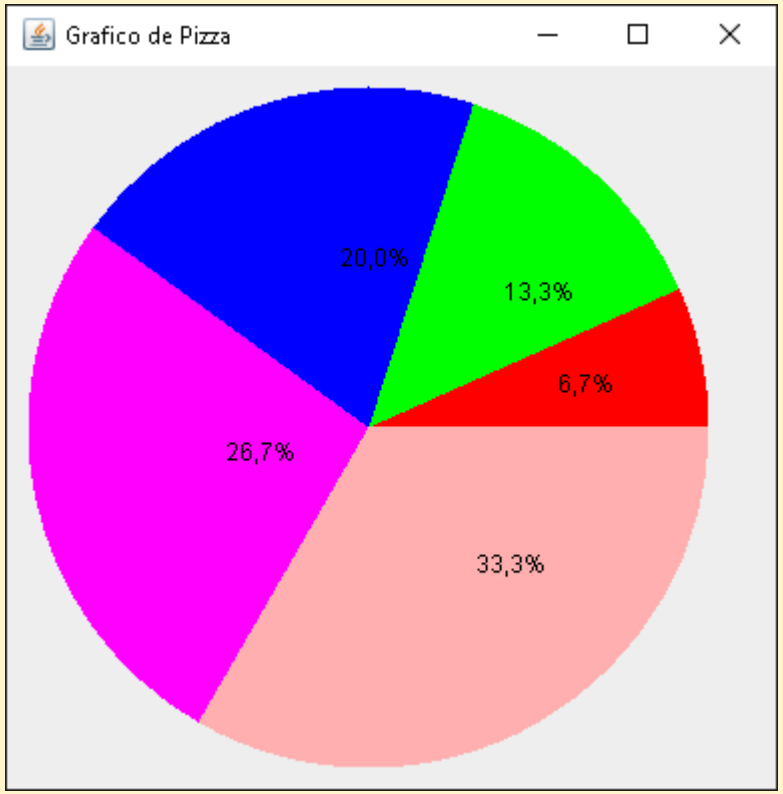


5ª Tarefa:

- Escreva um programa em Java utilizando janela para criar um gráfico de pizza dados cinco valores. Apresente um texto com a porcentagem de cada um dos cinco valores que irão compor o gráfico.

Exemplo de solução - Atividade 5

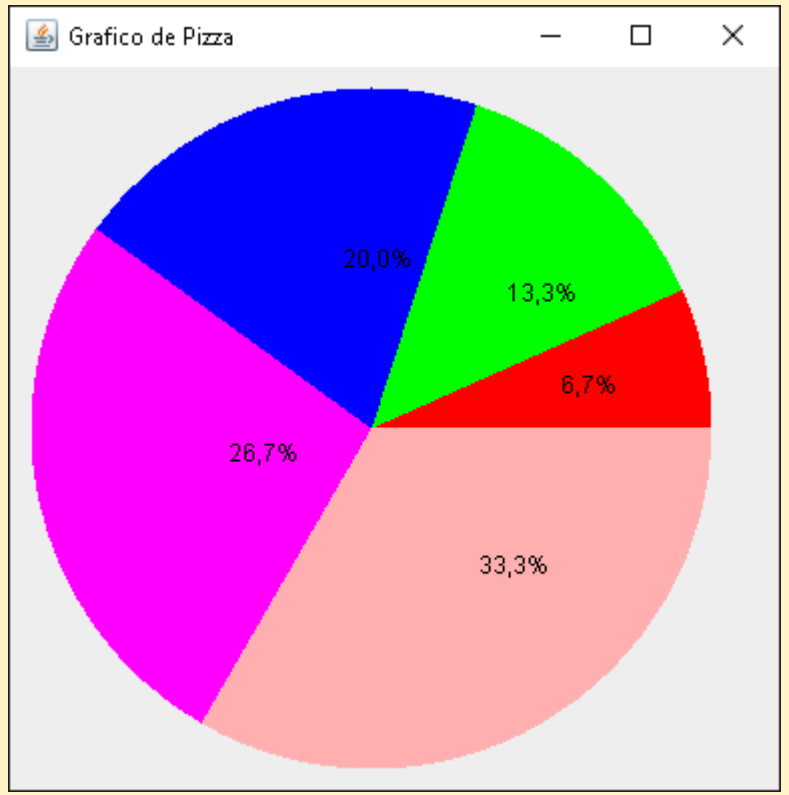
```
// Exemplo de solução da atividade 5
import java.awt.*;
import java.awt.event.*;

import javax.swing.*;

public class Pizza extends JPanel {
    private final int[] data;
    private int total;
    private final Color[] colors;
```



```
public Pizza(int[] data, Color[] colors) {
      this.data = data;
      this.colors = colors;
      total = 0;
      for (int i : data) {
          total += i;
      }
  }
  public void paintComponent(Graphics g) {
      super.paintComponent(g);
      Dimension size = getSize();
      Graphics2D g2d = (Graphics2D) g;
      int radius = Math.min(size.width, size.height) / 2 - 10;
      // desenhar as fatias
      float angle = 0;
      for (int i = 0; i < data.length; i++) {
          float perc = data[i] / (float)total; // calcula % da fatia
          g.setColor(colors[i]);
          g.fillArc(10, 10, radius*2, radius*2, (int)(angle*360), (int)(perc*360));
          angle += perc; // incrementa ângulo da próxima fatia
      }
```

```
        // desenhar as legenda
        angle = 0;
        for (int i = 0; i < data.length; i++) {
            float perc = data[i] / (float)total;

            int x = size.width/2;
            int y = size.height/2;
            x += (int)(radius/2 * Math.cos((angle + perc/2.0f) * Math.PI * 2.0));
            y -= (int)(radius/2 * Math.sin((angle + perc/2.0f) * Math.PI * 2.0));

            g.setColor(Color.BLACK);
            g2d.drawString(String.format("%.1f%%", perc*100.0), x, y);

            angle += perc;
        }
    }

@Override
    public Dimension getPreferredSize() {
        return new Dimension(200, 200);
    }
```

```
   public static void main (String args[])
   {
      JFrame janela = new JFrame("Grafico de Pizza");
      janela. setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      Pizza pizza = new Pizza(new int[] { 1, 2, 3, 4, 5 },
         new Color[] { Color.RED, Color.GREEN, Color.BLUE, Color.MAGENTA, Color.PINK });

      janela.add(pizza);
      janela.setSize(400,400);
      janela.setVisible(true);
   }
}
```



6ª Tarefa:

- Tome como base no exemplo 13.7 do livro Java Como Programar 10ª ed do Deitel, escreva um programa em Java que que possua dois botões, um no topo e um na base da janela. Esses botões serão utilizados para selecionar a cor inicial e final de um gradiente de cores a ser aplicado no painel colocado na região central da janela. Criar uma subclasse de JPanel que seja capaz de receber as cores que são configuradas pelo usuário através dos botões, e as utilize para desenhar um Rectangle2D com o tamanho do painel e preenchido com um gradiente de cores.

Exemplo de solução - Atividade 6

```
// Exemplo de solução da atividade 6. Baseado em figura 13.7
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class ShowColors2JFrame extends JFrame
{

   private final JButton changeColorFJButton;
   private final JButton changeColorIJButton;
   private Color corInicial = Color.YELLOW;
   private Color corFinal = Color.BLUE;
   private JGradientePanel colorJPanel;
```

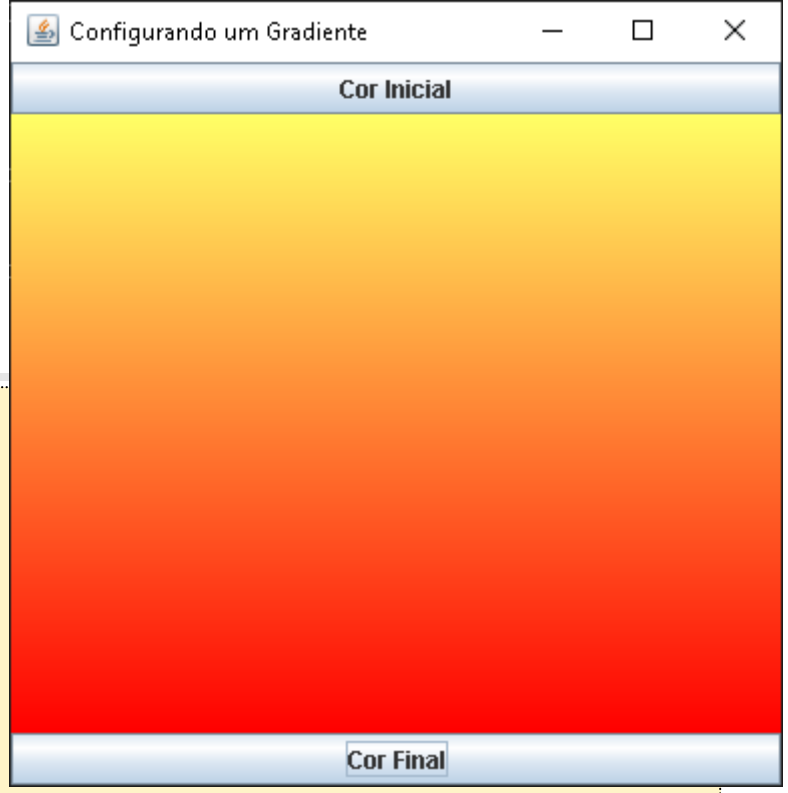
```
public ShowColors2JFrame() // configura a GUI
{
   super("Configurando um Gradiente");

   // cria JPanel para exibir as cores
   colorJPanel = new JGradientePanel();
   colorJPanel.setCorInicial(corInicial);
   colorJPanel.setCorFinal(corFinal);

   // configura changeColorFJButton e registra sua rotina de tratamento de evento
   changeColorFJButton = new JButton("Cor Final");
   changeColorFJButton.addActionListener( new ActionListener() // classe interna anônima
      {
         // exibe JColorChooser quando o usuário clica no botão
         @Override
         public void actionPerformed(ActionEvent event)
         {
            corFinal = JColorChooser.showDialog(
                ShowColors2JFrame.this, "Cor Inicial do Gradiente", color);
            // configura a cor padrão, se nenhuma cor for retornada
            if (corFinal == null)
                   corFinal = Color.YELLOW;
            // muda a cor de fundo do painel de conteúdo
            colorJPanel.setCorFinal(corFinal);
         }  // fim do método actionPerformed
      }     // fim da classe interna anônima
      );    // fim da chamada para addActionListener
```

```
// set up changeColorIJButton and register its event handler
changeColorIJButton = new JButton("Cor Inicial");
changeColorIJButton.addActionListener(
   new ActionListener() // anonymous inner class
   {
      // display JColorChooser when user clicks button
      @Override
      public void actionPerformed(ActionEvent event)
      {
         corInicial = JColorChooser.showDialog(
            ShowColors2JFrame.this, "Choose a color", corInicial);

         // set default color, if no color is returned
         if (corInicial == null)
            corInicial = Color.BLUE;

         // change content pane's background color
         colorJPanel.setCorInicial(corInicial);
      }
   } // end anonymous inner class
); // end call to addActionListener
```

```
      add(colorJPanel, BorderLayout.CENTER);
      add(changeColorFJButton, BorderLayout.SOUTH);
      add(changeColorIJButton, BorderLayout.NORTH);

      setSize(400, 400);
      setVisible(true);
   } // end ShowColor2JFrame constructor
} // end class ShowColors2JFrame
```

```
1 // Figura 13.8: ShowColors2.java
2 // Escolhendo cores com JColorChooser.
3 import javax.swing.JFrame;
4
5  public class ShowColors2
6  {
7  // executa o aplicativo
8     public static void main(String[] args)
9     {
10          ShowColors2JFrame application = new ShowColors2JFrame();
11          application.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
12    }
13 } // fim da classe ShowColors2
```

Exemplo - Gradiente




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

public class JGradientePanel extends JPanel
{
   Color c1, c2; // define as variáveis para cor inicial e final do gradiente.

   public JGradientePanel()
   {
      c1 = Color.YELLOW;
      c2 = Color.BLUE;
   }

   public void setCorInicial(Color corInicial){
      if (corInicial == null)
            corInicial = Color.YELLOW;
```

```
        c1 = corInicial;
        repaint();
    }
    public void setCorFinal(Color corFinal){
        if (corFinal == null)
             corFinal = Color.BLUE;

        c2 = corFinal;
        repaint();
    }
    @Override
    public void paintComponent(Graphics g)
    {
        super.paintComponent(g);

        Graphics2D g2d = (Graphics2D) g; // casting g para Graphics2D

        // Cria um gradiente vertical a partir do ponto inicial 0,0 com cor c1 e
        // termino em 0, altura do painel com cor c2. E seleciona no contexto gráfico
        g2d.setPaint(new GradientPaint(0, 0, c1, 0, getHeight(), c2, true));
        // desenha retângulo 2D preenchida com um gradiente azul-amarelo
        g2d.fill(new Rectangle2D.Double(0, 0, getWidth(), getHeight() ));
    }
} // end class JGradientePanel
```