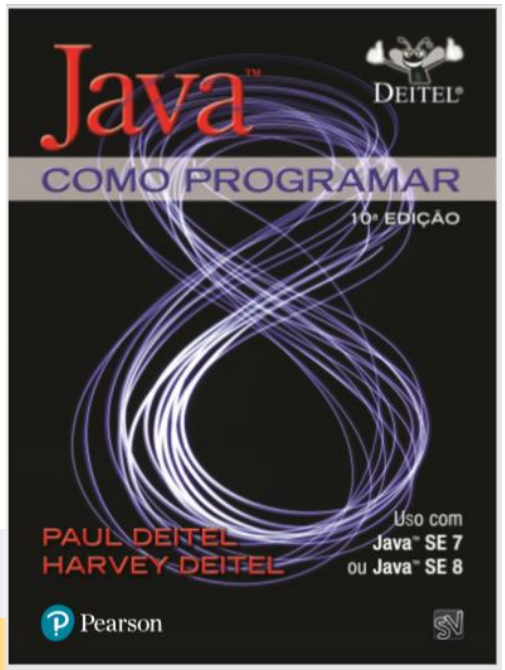


Guia de Laboratório

Lab 07

Prof. André Bernardi

andrebernardi@unifei.edu.br

Outubro - 2021

7º Laboratório ECOP15
25 e 26 de Outubro de 2021

- **Utilizar como referência os exemplos:**
  - ShowColors2JFrame e JGradientePanel
  - SliderDemo e SliderFrame

1ª Tarefa:

Alterar a tarefa 6 do lab3 para acrescentar menus com as seguintes opções:

- Criar um menu **Arquivo:**
  - Criar uma opção **Sobre**, que exibe as informações sobre o programa e o programador;
  - Adicionar um separador;
  - Criar uma opção **Sair**, que encerra o programa quando for selecionada.
- Criar um menu Configurar:
  - Adicionar dois **JMenuItens** para permitir a seleção das cores iniciais e finais do gradiente exibido na tela.

6ª Tarefa, Lab 03:

- 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;
```

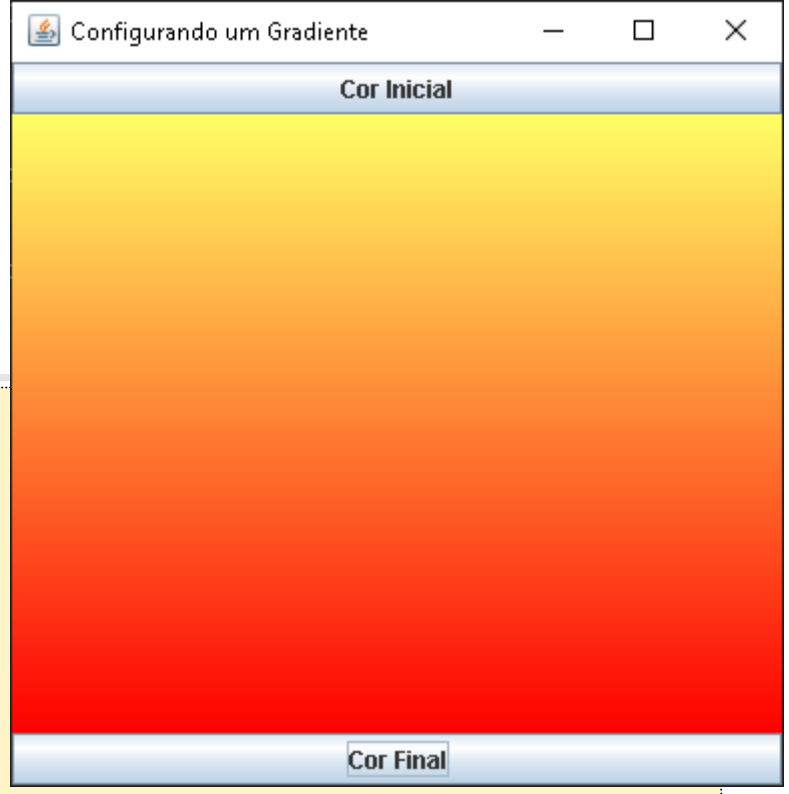
```
public ShowColors2JFrame() // configura a GUI
{
   super("Configurando um Gradiente");

   // 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
```

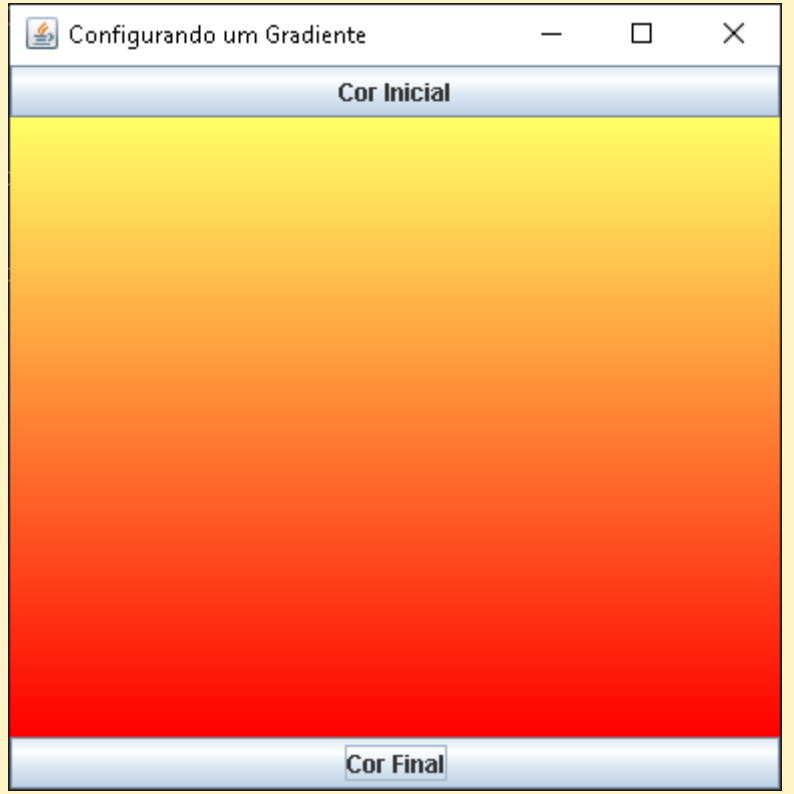
```
1 // Figura 13.8: ShowColors2.java
2 // Escolhendo cores com JColorChooser.
3 import javax.swing.JFrame;
4
5  public class ShowColors2
6  {
7  // executa o aplicativo
8     public static void main(String[] args)
9     {
10          ShowColors2JFrame application = new ShowColors2JFrame();
11     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

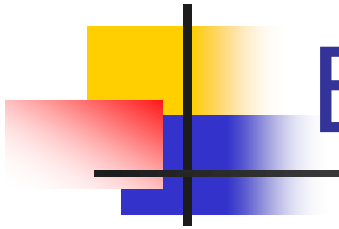
Exemplo - Gradiente




```
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
```





Exemplo de Solução Ex 1.

- A classe do painel que exibe o gradiente vai ser mantida inalterada.
- A classe que contém a janela que insere os dois botões e o painel do gradiente será alterada para inserir os menus solicitados.
- No código a seguir ficará em destaque os locais que foram alterados em relação ao programa original.

```
// Exemplo de solução da atividade 1. Baseado em figura 13.7 e ex6 do lab3
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class Lab7_01_JFrame extends JFrame
{
   private final JButton changeColorFJButton;
   private final JButton changeColorIJButton;
   private Color corInicial = Color.YELLOW;
   private Color corFinal = Color.BLUE;
   private JGradientePanel colorJPanel;

   // set up GUI
   public Lab7_01_JFrame()
   {
      super("Configurando um Gradiente");

      // create JPanel for display color
      colorJPanel = new JGradientePanel();
      colorJPanel.setCorInicial(corInicial);
      colorJPanel.setCorFinal(corFinal);
```

```
    // set up changeColorFJButton and register its event handler
    changeColorFJButton = new JButton("Cor Final");
    changeColorFJButton.addActionListener(   new ActionListener() // anonymous inner class
       { // display JColorChooser when user clicks button
          @Override
          public void actionPerformed(ActionEvent event)
          {
             selecionaCorFinal();
          }
       } // end anonymous inner class
    ); // end call to 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)
          {
             selecionaCorInicial();
          }
       } // end anonymous inner class
    ); // end call to addActionListener
```

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

    preparaMenu();

    setSize(400, 400);
    setVisible(true);

} // end Lab7_01_JFrame constructor

public void preparaMenu()
{
    JMenuBar barra = new JMenuBar();
    JMenu menuArquivo = new JMenu( "Arquivo" );
    menuArquivo.setMnemonic('A');
    barra.add(menuArquivo);

    JMenu menuConfigurar = new JMenu ("Configurar");
    menuConfigurar.setMnemonic('C');
    barra.add(menuConfigurar);

    setJMenuBar(barra);
```

```
JMenuItem itemSobre = new JMenuItem ( "Sobre ...");
menuArquivo.add(itemSobre);
itemSobre.addActionListener( new ActionListener() // anonymous inner class
    {   @Override
        public void actionPerformed(ActionEvent event)
        {
            JOptionPane.showMessageDialog(Lab7_01_JFrame.this,
                "Programa exemplo de solução\nquestão 1 do lab 07\n Prof. André",
                "Sobre", JOptionPane.PLAIN_MESSAGE);
        }
    }
);
menuArquivo.addSeparator();

JMenuItem itemSair = new JMenuItem ( "Sair" );
menuArquivo.add(itemSair);
itemSair.addActionListener(  new ActionListener() // anonymous inner class
    {   // terminate application when user clicks exitItem
        public void actionPerformed(ActionEvent event)
        {
            System.exit(0); // exit application
        }
    }      );
```

```
   JMenuItem itemCorInicial = new JMenuItem ("Cor Inicial ...");
   menuConfigurar.add(itemCorInicial);
   itemCorInicial.addActionListener( new ActionListener() // anonymous inner class
      {  // display JColorChooser when user clicks button
         public void actionPerformed(ActionEvent event)
         {
            selecionaCorInicial();
         }
      } // end anonymous inner class
   ); // end call to addActionListener

   JMenuItem itemCorFinal = new JMenuItem ("Cor Final ...");
   menuConfigurar.add(itemCorFinal);
   itemCorFinal.addActionListener(
      new ActionListener() // anonymous inner class
      {   // display JColorChooser when user clicks button
         @Override
         public void actionPerformed(ActionEvent event)
         {
            selecionaCorFinal();
         }
      }      ); // end call to addActionListener
}
```

```
  public void selecionaCorInicial()
  {
       corInicial = JColorChooser.showDialog(Lab7_01_JFrame.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);
   }

   public void selecionaCorFinal()
   {
        corFinal = JColorChooser.showDialog(Lab7_01_JFrame.this, "Choose a color", corFinal);

         // set default color, if no colorj is returned
         if (corFinal == null) corFinal = Color.YELLOW;

         // change content pane's background color
         colorJPanel.setCorFinal(corFinal);
   }
} // end class Lab7_01_JFrame
```

2ª Tarefa:

(Componente personalizado JSlider+JTextField)

Implementar uma subclasse de JPanel que sirva como um componente personalizado que exibe o valor selecionado em um JSlider em um JTextField. Sincronizar os dois componentes de modo que quando o usuário alterar o valor no JTextField o JSlider seja reposicionado para refletir o valor digitado. Um JLabel deve ser utilizado para identificar o JTextField. Os métodos JSlider *setValue* e *getValue* devem ser utilizados.

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

public class SliderPanel extends JPanel
{
   private JTextField texto;
   private JSlider slider;
   private int minValue = 0;
   private int maxValue = 255;

   public SliderPanel ()
   {
      texto = new JTextField(5);

      // set up JSlider to control diameter value
      slider =
         new JSlider(SwingConstants.HORIZONTAL, minValue, maxValue, 10);
      slider.setMajorTickSpacing(50); // create tick every 10
      slider.setPaintTicks(true); // paint ticks on slider
```

```
    slider.addChangeListener(          // register JSlider event listener
       new ChangeListener()           // anonymous inner class
       {  // handle change in slider value
          @Override
          public void stateChanged(ChangeEvent e)
          {
             texto.setText(""+slider.getValue());
             repaint();
          }
       }
    );
    texto.setText(""+slider.getValue());
    texto.addActionListener( new ActionListener() {// register JTextField event listener
       public void actionPerformed(ActionEvent e){
          int valor  = Integer.parseInt(texto.getText());
          slider.setValue(valor);
       }
    } );

    add (slider);
    add(new JLabel("valor selecionado"));
    add(texto);
}   // fim do construtor de SliderPanel
```

```
@Override
public void paintComponent(Graphics g)
{
   super.paintComponent(g);
}

// validate and set minValue, then repaint
public void setMinValue(int newValue)
{
   // if diameter invalid, default to 10
   minValue = (newValue >= 0 ? newValue : 10);
  if (minValue > maxValue) maxValue = minValue +1;
   repaint(); // repaint panel
}

// validate and set minValue, then repaint
public void setMaxValue(int newValue)
{
   maxValue = (newValue > minValue ? newValue : minValue+1);
   repaint(); // repaint panel
}
```

```
// used by layout manager to determine preferred size
   public Dimension getPreferredSize()
   {
      return new Dimension(400, 150);
   }

   // used by layout manager to determine minimum size
   public Dimension getMinimumSize()
   {
      return getPreferredSize();
   }

   public int getValue ()
   {
      return slider.getValue();
   }
} // end class SliderPanel
```

3ª Tarefa:

(Criando um seletor de cores) Declare uma subclasse do JPanel chamada MyColorChooser que fornece três objetos JSlider e três objetos JTextField. Cada JSlider representa os valores de 0 a 255 para as partes de azul, verde e vermelha de uma cor. Utilize esses valores como os argumentos para o construtor Color a fim de criar um novo objeto Color. Exiba o valor atual de cada JSlider no correspondente JTextField. Quando o usuário altera o valor do JSlider, o JTextField deve ser alterado correspondentemente. Utilize seu novo componente GUI como parte de um aplicativo que exibe o valor Color atual desenhando um retângulo preenchido. Utilize o componente criado na segunda questão para manter sincronizado o valor selecionado no JSlider e o preenchido no JTextField.

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

public class MyColorChooser extends JPanel
{
   private SliderPanel sliderR;
   private SliderPanel sliderG;
   private SliderPanel sliderB;

   public MyColorChooser ()
   {
      setLayout(new GridLayout(3,1) );

      sliderR = new SliderPanel();
      sliderG = new SliderPanel();
      sliderB= new SliderPanel();

      add (sliderR);
      add (sliderG);
      add (sliderB);
      sliderR.setOpaque(false);
      sliderG.setOpaque(false);
      sliderB.setOpaque(false);
   }

   @Override
   public void paintComponent(Graphics g)
   {
      super.paintComponent(g);
      g.setColor(getColor());
     g.fillRect(0,0, 200,200);

     //setBackground(getColor());
     this.setBackground(getColor());
   }

// used by layout manager to determine preferred size
   public Dimension getPreferredSize()
   {
      return new Dimension(400, 150);
   }
```

```
// used by layout manager to determine minimum size
   public Dimension getMinimumSize()
   {
      return getPreferredSize();
   }

   public Color getColor ()
   {
      Color cor = new Color(sliderR.getValue(), sliderG.getValue(), sliderB.getValue() );
      return cor;
   }

   //main de teste
   public static void main(String[] args)
   {
      JFrame janela =  new JFrame ("teste MyColorChooser");
      janela.add( new MyColorChooser());
      janela.setSize(400,450);
      janela.setVisible(true);
      janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   }
} // end class MyColorChooser
```

4ª Tarefa:

(Criando um seletor de cores: modificação) Alterar o programa anterior para que seja possível exibir a cor selecionada em um retângulo desenhado no fundo do próprio componente. Utilize o método paintComponent dessa subclasse de JPanel para desenhar o retângulo preenchido com a cor selecionada. A classe deve conter um método set que recebe como parâmetro os valores das componentes vermelho, verde e azul, a ser exibido no JSlider e JTextField correspondentes. Quando o método set for invocado, o painel de desenho deve automaticamente atualizar todas informações nele contida.