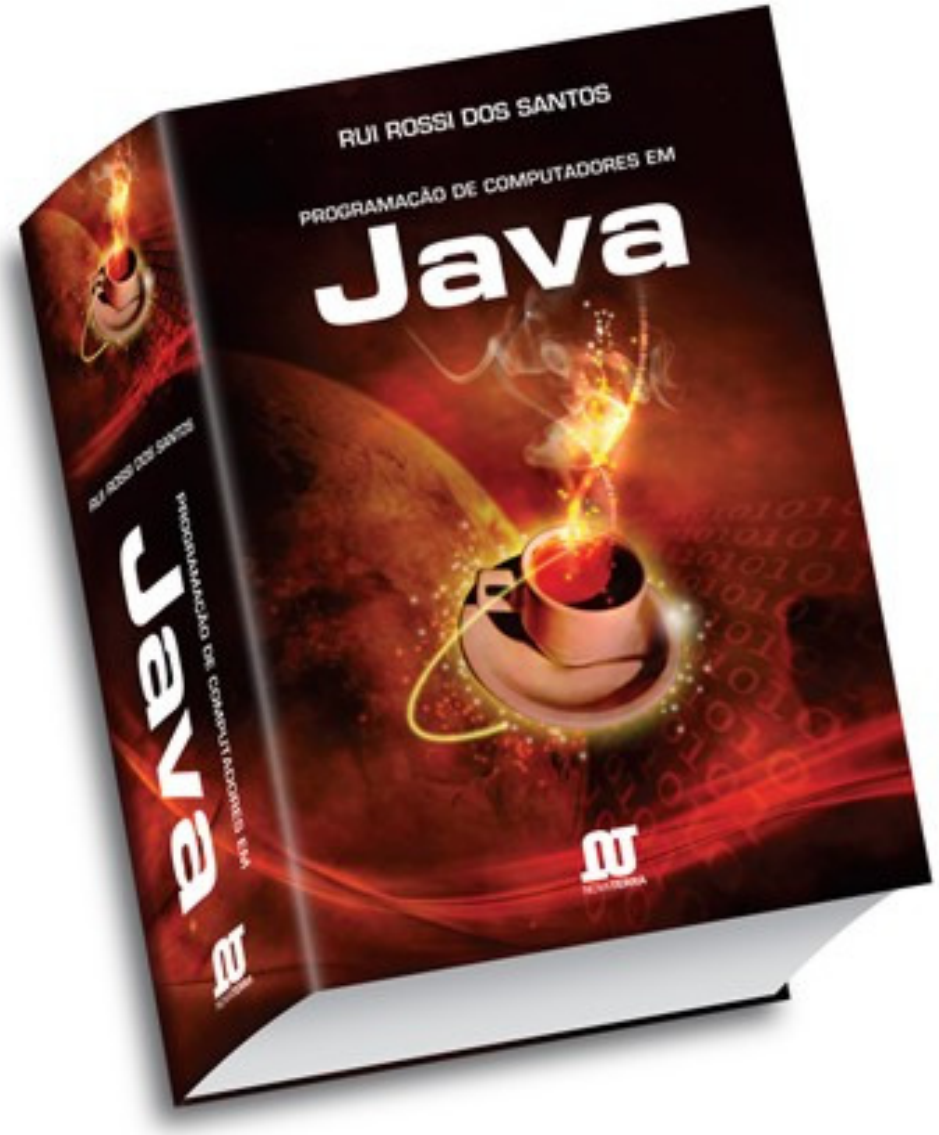


Capítulo 31

Outros Recursos para Interfaces Gráficos

Objetivos do Capítulo

- Introduzir recursos úteis à construção de grande variedade de interfaces gráficas e que não foram abordados antes.
- Apresentar seis componentes que podem ser agregados a uma janela: grades, barras de ferramentas, conjuntos de painéis acessíveis por abas, controles deslizantes, campos formatados e painéis de seleção de cor.
- Analisar o funcionamento das janelas internas e descrever os passos necessários para a definição de uma imagem de fundo para a área de trabalho de uma janela.
- Indicar como a aparência e o comportamento plugáveis dos componentes do Swing são manipulados.

Grades

- javax.swing.JTable
  - Atributos para a definição do modo de redimensionamento:
    - AUTO_RESIZE_OFF
    - AUTO_RESIZE_NEXT_COLUMN
    - AUTO_RESIZE_SUBSEQUENT_COLUMNS
    - AUTO_RESIZE_LAST_COLUMN
    - AUTO_RESIZE_ALL_COLUMNS
  - Construtores:
    - JTable( )
    - JTable(Object[][] rowData, Object[] columnNames)
    - JTable(TableModel dm)
    - JTable(Vector rowData, Vector columnNames)

Grades

- javax.swing.JTable
  - Métodos:
    - Class<?> getColumnClass(int column)
    - int getColumnCount( )
    - TableModel getModel( )
    - int getRowCount( )
    - ListSelectionModel getSelectionModel( )
    - Object getValueAt(int row, int column)
    - setModel(TableModel dataModel)
    - setValueAt(Object aValue, int row, int column)
    - setAutoresizeMode(int mode)
    - setSelectionMode(int selectionMode)

Grades

- interface javax.swing.ListSelectionModel
  - Atributos para a definição do modo de seleção da grade:
    - MULTIPLE_INTERVAL_SELECTION
    - SINGLE_INTERVAL_SELECTION
    - SINGLE_SELECTION
  - Métodos:
    - clearSelection( )
    - int getMaxSelectionIndex( )
    - int getMinSelectionIndex( )
    - boolean isSelectionEmpty( )

Grades

- javax.swing.table.AbstractTableModel implements javax.swing.TableModel
  - Métodos cuja implementação é obrigatória nas subclasses
    - int getRowCount()
    - int getColumnCount()
    - Object getValueAt(int row, int column)
  - Outros métodos:
    - Class<?> getColumnClass(int columnIndex)
    - String getColumnName(int column)
    - boolean isCellEditable(int rowIndex, int columnIndex)
    - setValueAt(Object aValue, int rowIndex, int columnIndex)

Grades

- **Código 31.1 – Grade.java**

Grades

- **Código 31.2 – ModeloGrade.java**

Barras de Ferramentas

- javax.swing.JToolBar
  - Atributos para definição da orientação:
    - HORIZONTAL
    - VERTICAL
  - Construtores:
    - JToolBar( )
    - JToolBar(String name, int orientation)
  - Métodos:
    - Component add(Component comp)
    - addSeparator( )
    - remove(Component comp)
    - remove(int index)
    - setFloatable(boolean b)
- javax.swing.SwingUtilities
  - Métodos:
    - static void updateComponentTreeUI(Component c)

Barras de Ferramentas

- **Código 31.3 – Barra.java**

Janelas Internas

- javax.swing.JInternalFrame
  - Construtores:
    - JInternalFrame( )
    - JInternalFrame(String title)
    - JInternalFrame(String title, boolean resizable, boolean closable, boolean maximizable, boolean iconifiable)
  - Métodos:
    - getContentPane( )
    - setTitle( )
    - setSize( )
    - setClosable( )
    - setIconifiable( )
    - setResizable( )
- javax.swing.JDesktopPane extends JLayeredPane
  - Métodos:
    - Component add(Component comp)

Janelas Internas

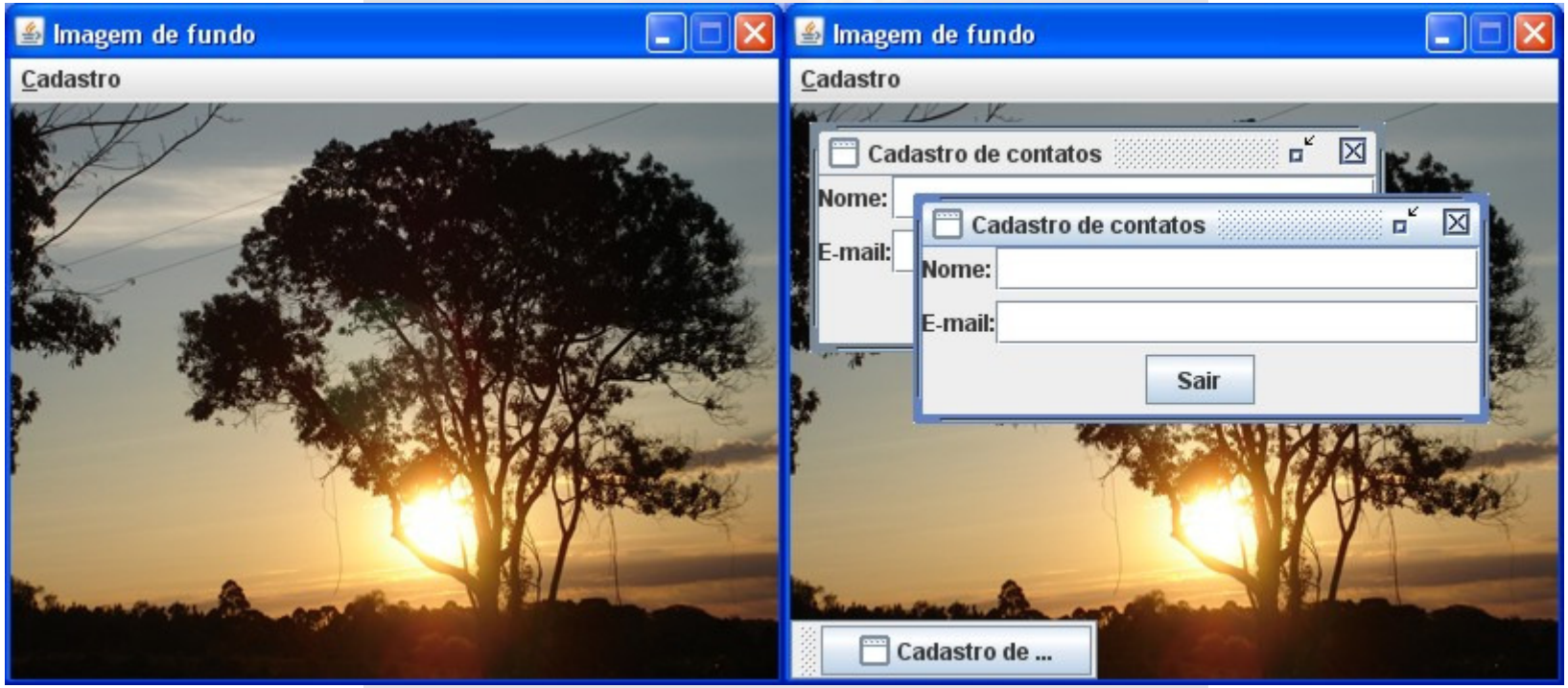
- Código 31.4 – IFContato.java
- Código 31.5 – JFMenu.java

Imagem de Fundo

- **Passos essenciais:**
  - **Criar uma subclasse de JDesktopPane ou de JPanel**
  - **Acrescentar um atributo para reter a imagem (ImageIcon)**
  - **Implementar um construtor que inicialize o atributo**
  - **Sobrescrever o método paintComponent( ) para desenhar a imagem**
  - **Sobrescrever o método getPreferredSize( ) se desejar adequar o tamanho do contêiner ao tamanho da imagem**

Imagem de Fundo

- Código 31.6 – DesktopImagem.java
- Código 31.7 – JFMenuFundo.java



Aparência e Comportamento Plugáveis

- javax.swing.LookAndFeel

- javax.swing.UIManager
  - Métodos estáticos:
    - UIManager.LookAndFeelInfo[] getInstalledLookAndFeels()
    - LookAndFeel getLookAndFeel( )
    - setLookAndFeel(LookAndFeel newLookAndFeel)
    - setLookAndFeel(String className)

- LookAndFeelInfo
  - Métodos:
    - String getClassName( )
    - String getName( )

Aparência e Comportamento Plugáveis

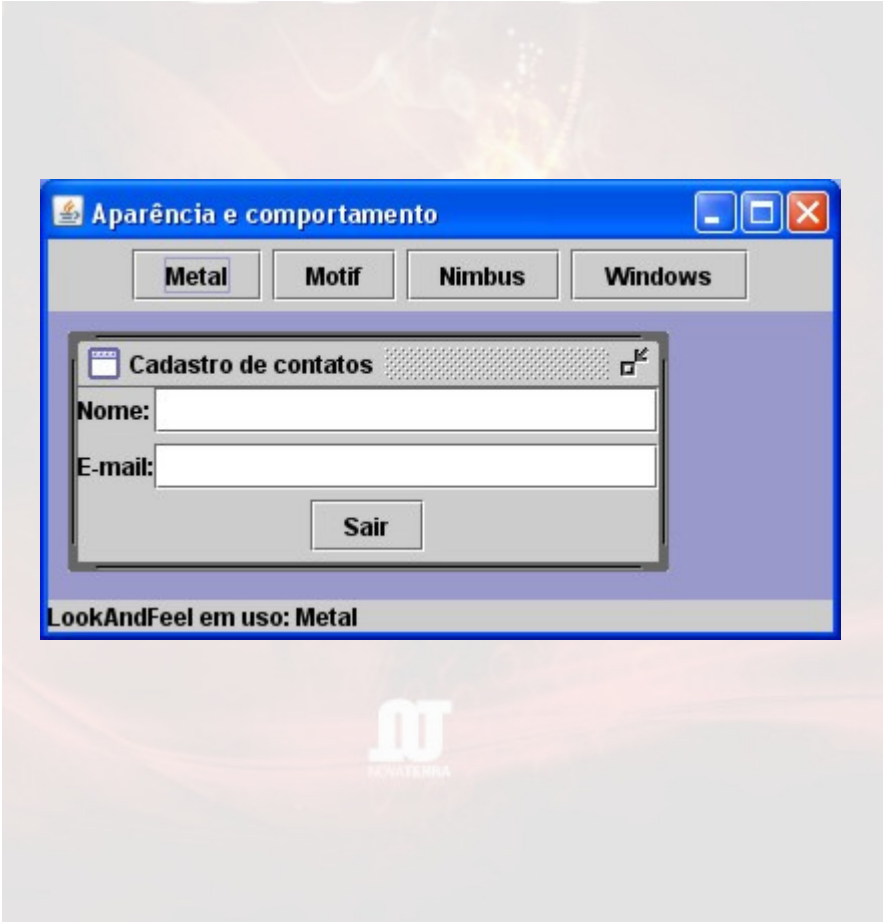
- **Código 31.8 – ListaLAFs.java**

Aparência e Comportamento Plugáveis

- **Código 31.9 – ExemploLAF.java**

Aparência e Comportamento Plugáveis

- **Código 31.10 – TrocaTema.java**

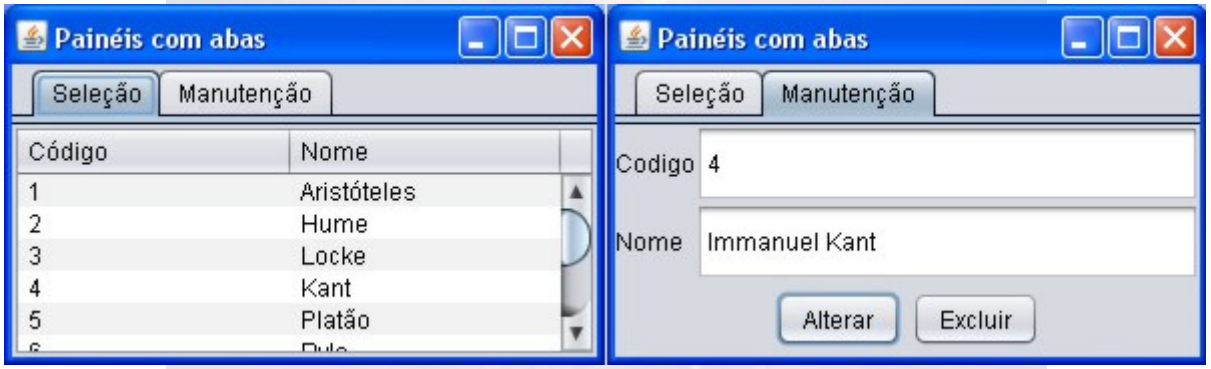


Fichários

- **javax.swing.JTabbedPane extends JComponent**
  - **Métodos:**
    - **Component add(String title, Component component)**
    - **Component add(Component component, int index)**
    - **Component getSelectedComponent( )**
    - **int getSelectedIndex( )**
    - **remove(Component component)**
    - **remove(int index)**
    - **setSelectedComponent(Component c)**
    - **setSelectedIndex(int index)**

Fichários

❑ Código 31.11 – ExemploAbas.java

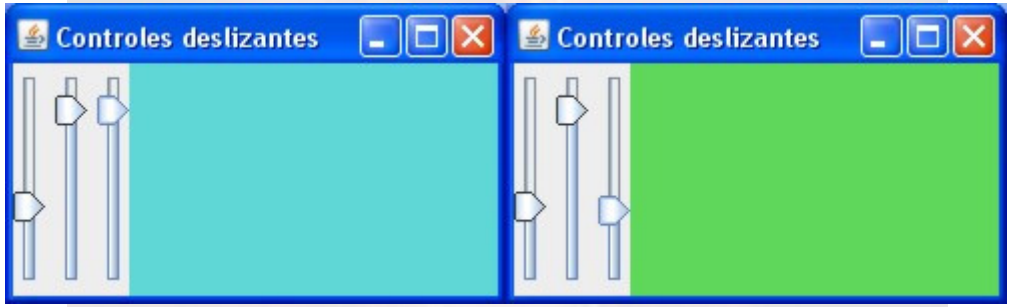


Controles Deslizantes

- javax.swing.JSlider extends JComponent
  - Atributos para definição da orientação:
    - HORIZONTAL
    - VERTICAL
  - Construtores:
    - JSlider( )
    - JSlider(int orientation)
    - JSlider(int min, int max)
    - JSlider(int min, int max, int value)
    - JSlider(int orientation, int min, int max, int value)
  - Métodos:
    - int getMaximum( )
    - int getMinimum( )
    - int getOrientation( )
    - int getValue( )
    - setMaximum(int maximum)
    - setMinimum(int minimum)
    - setOrientation(int orientation)
    - setValue(int n)

Controles Deslizantes

- **Código 31.12 – ExemploSlider.java**

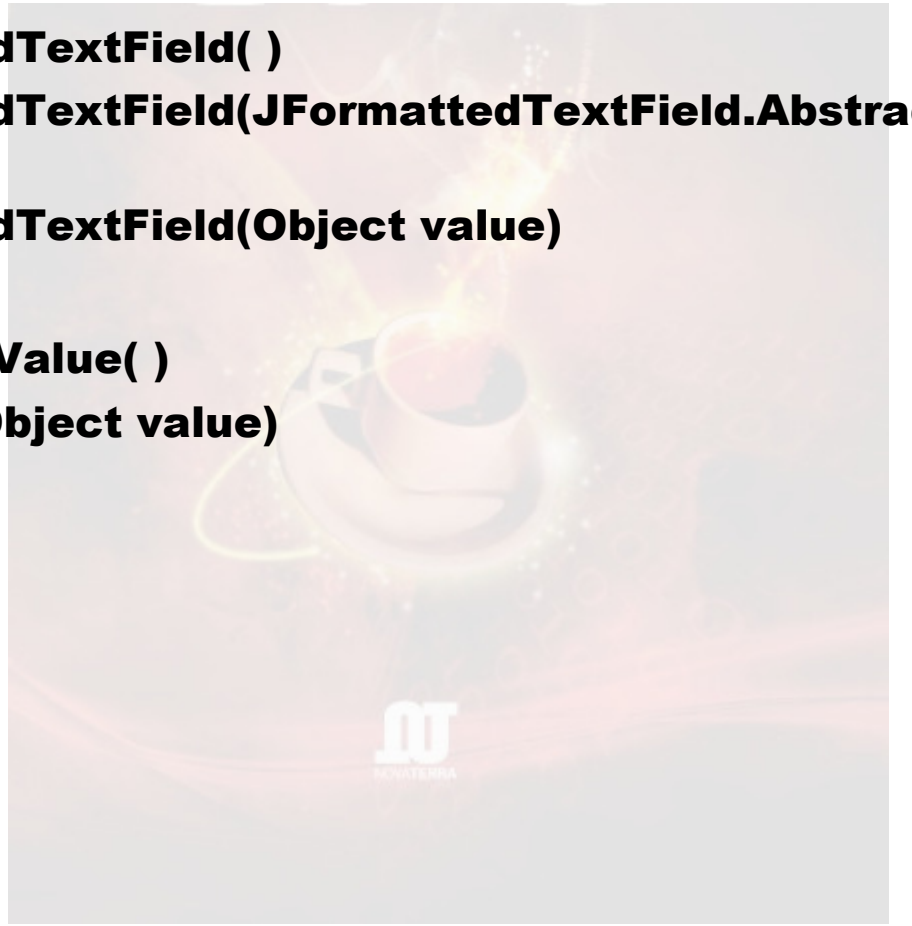


Campos de Texto Formatados

- **javax.swing.text.MaskFormatter**
  - **Construtor:**
    - **MaskFormatter(String mask)**
  - **Caracteres válidos para especificação da máscara:**
    - ***: permite letras, números e caracteres especiais.**
    - **A: permite letras e números.**
    - **?: permite somente letras.**
    - **U: permite somente letras e converte letras minúsculas em maiúsculas.**
    - **L: permite somente letras e converte letras maiúsculas em minúsculas.**
    - **#: permite somente números.**
    - **H: permite qualquer caractere que forma um número hexadecimal.**
    - **‘: permite imprimir qualquer um dos caracteres especiais anteriores.**

Campos de Texto Formatados

- **javax.swing.JFormattedTextField**
  - **Construtores:**
    - **JFormattedTextField( )**
    - **JFormattedTextField(JFormattedTextField.AbstractFormatter formatter)**
    - **JFormattedTextField(Object value)**
  - **Métodos:**
    - **Object getValue( )**
    - **setValue(Object value)**



Campos de Texto Formatados

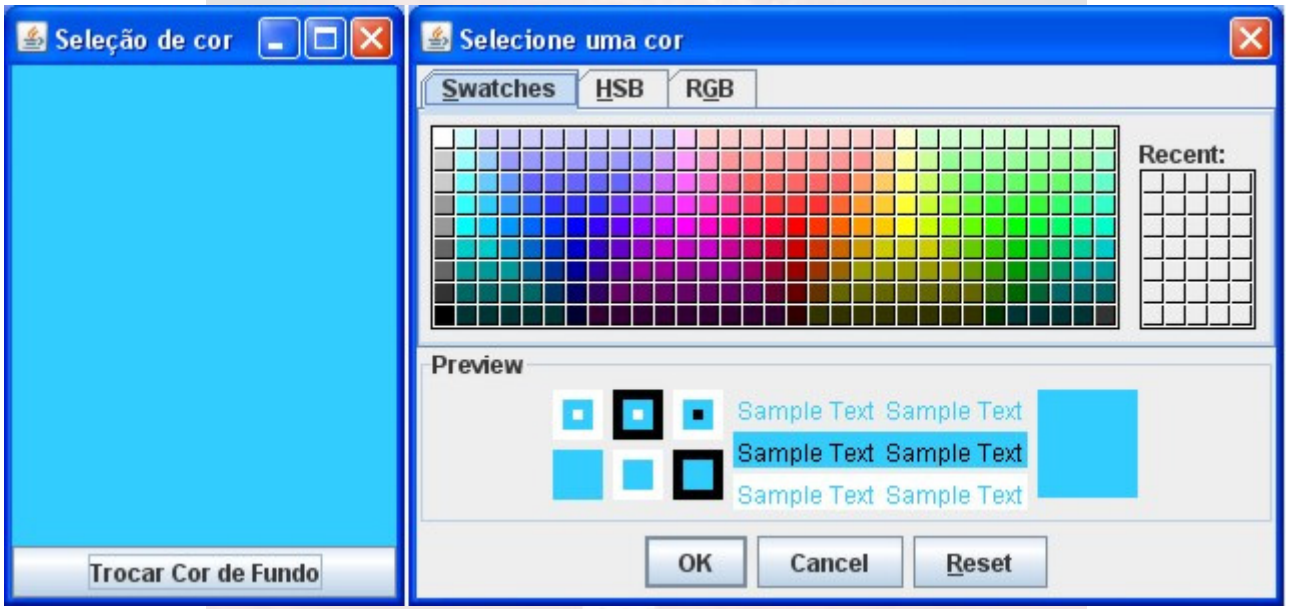
- **Código 31.13 – ExemploCamposFormatados.java**

Seleção de Cor

- javax.swing.JColorChooser
  - Método estático:
    - Color showDialog(Component component, String title, Color initialColor)
  - Outros métodos:
    - AbstractColorChooserPanel[] getChooserPanels()
    - Color getColor()
    - ColorSelectionModel getSelectionModel()
    - setColor(Color color)
    - AbstractColorChooserPanel removeChooserPanel(AbstractColorChooserPanel panel)

Seleção de Cor

- **Código 31.14 – DialogoCor.java**



Seleção de Cor

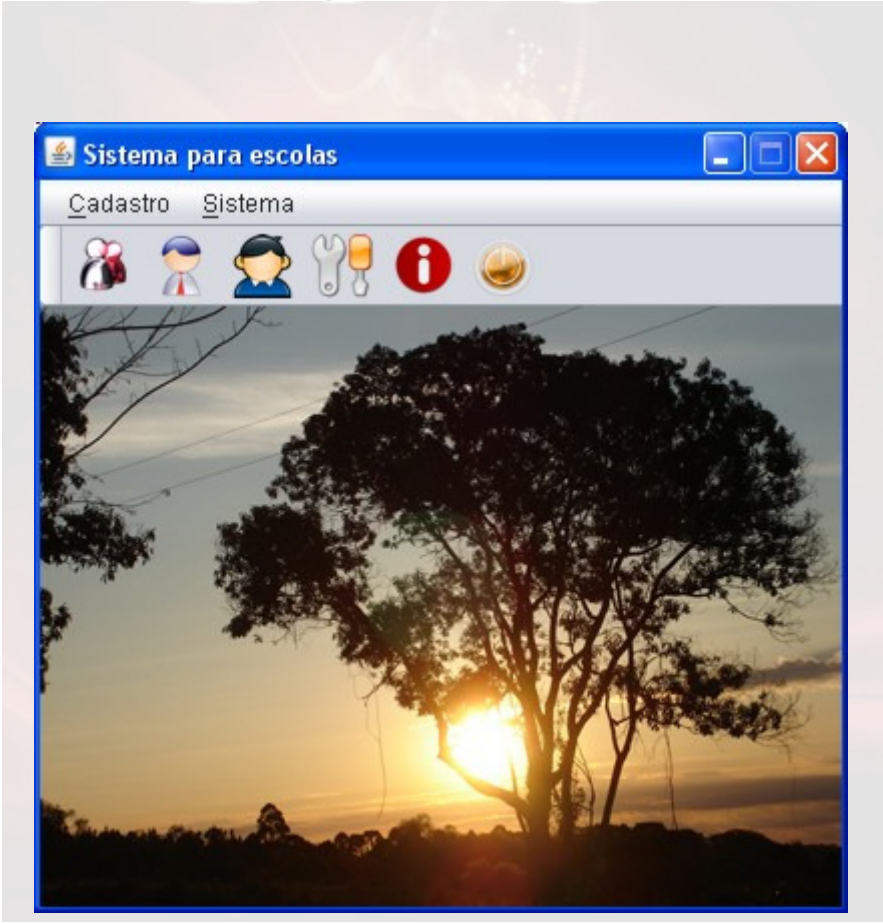
- **Código 31.15 – PainelCor.java**

Exercícios

- **Todos os exercícios propostos a seguir estão interligados e formam uma seqüência de passos para a construção de um único aplicativo.**
  - **O aplicativo que você deverá construir simula algumas operações que são comuns em sistemas utilizados em escolas que ofertam cursos de idiomas e de informática.**
  - **O primeiro exercício indica como deve ser criada a janela principal deste aplicativo e como criar a classe executável do mesmo.**
  - **O segundo exercício indica como criar as janelas dos cadastros.**
  - **O terceiro exercício indica como implementar as demais funcionalidades.**

Exercício 1

- **Crie uma nova janela, chamada JFPrincipal, de acordo com o modelo apresentado na figura abaixo.**

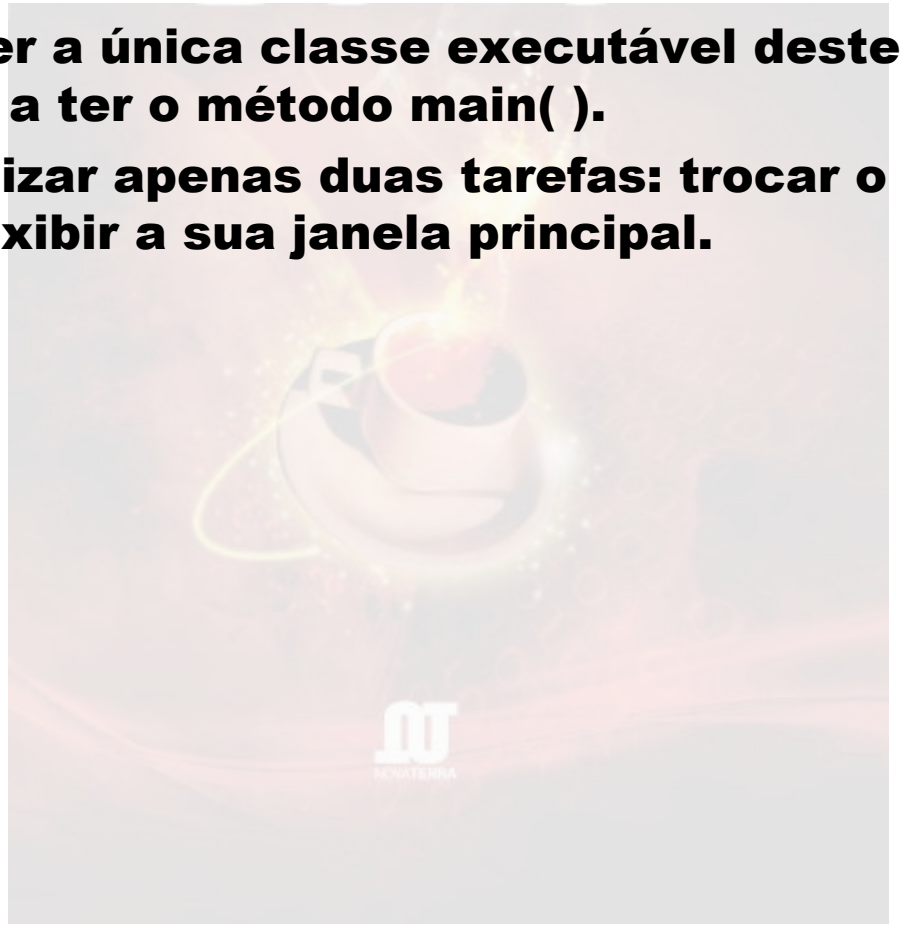


Exercício 1

- Crie uma nova janela, chamada JFPrincipal, de acordo com o modelo apresentado na figura abaixo.
- Esta é a janela principal do aplicativo e deve ter uma barra de menus e uma barra de ferramentas.
  - A barra de menus deve ter dois menus: “Cadastro” e “Sistema”.
  - O primeiro destes menus deve ter três itens: “Usuário”, “Professor” e “Aluno”.
  - O segundo menu também deve ter três itens: “Configurações”, “Sobre ...” e “Sair”.
  - A barra de ferramentas deve ter botões que permitam o acesso rápido às mesmas operações que são acessíveis através dos itens de menu.
  - A área de trabalho desta janela deve apresentar uma imagem de fundo.

Exercício 1

- **Crie outra classe, chamada Principal, que assuma o papel de classe principal do sistema.**
  - **Ela deverá ser a única classe executável deste aplicativo, ou seja, a única a ter o método main( ).**
  - **Ela deve realizar apenas duas tarefas: trocar o look and feel para o Nimbus e exibir a sua janela principal.**



Exercício 2

- O cadastro de usuários deverá ser composto pelos seguintes dados: um código numérico, o nome completo do usuário, seu login e sua senha.
  - Crie uma classe, chamada Usuario, para encapsular estes dados e utilize os métodos de escrita para validá-los.
- A janela de cadastro de Professores deverá ter campos para receber os seguintes dados: um código numérico, o nome completo do professor, seu salário, seu telefone e seu e-mail.
  - Além destes, acrescente campos para receber outros dados que você julgar que são pertinentes a este tipo de cadastro.
  - Crie uma classe, chamada, Professor, para encapsular estes dados e utilize os métodos de escrita para validá-los.
- A janela de cadastro de Alunos deverá ter campos para receber os seguintes dados: um código numérico, o nome completo do aluno, sua data de nascimento, seu telefone e seu e-mail.
  - Além destes, acrescente campos para receber outros dados que você julgar que são pertinentes a este tipo de cadastro.
  - Crie uma classe, chamada, Aluno, para encapsular estes dados e utilize os métodos de escrita para validá-los.

Exercício 2

- Crie uma janela interna para cada um dos três cadastros supracitados. A figura abaixo ilustra como os componentes destas janelas deverão ser organizados.

Exercício 2

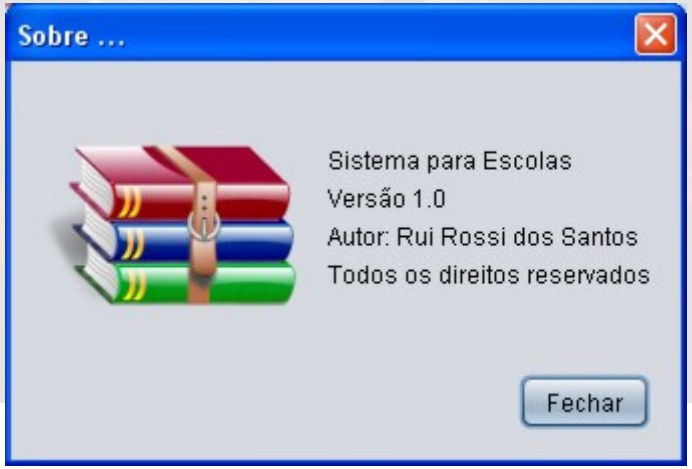
- Este modelo representa a janela de cadastro de usuários, mas você também pode empregá-lo para a construção das janelas de cadastro de professores e de alunos.
  - Note que estas janelas possuem duas fichas.
  - A primeira ficha apresenta os registros existentes em uma grade.
  - A segunda ficha permite incluir novos registros, alterar registros existentes e excluí-los.
  - Utilize campos formatados sempre que isso for conveniente.
- Cada usuário cadastrado deve ser representado como uma instância da classe Usuario, cada aluno cadastrado deve ser representado como uma instância da classe Aluno e cada professor cadastrado deve ser representado como uma instância da classe Professor.
  - Mesmo quando as janelas de cadastro forem fechadas, estas instâncias devem ser mantidas em memória.
  - Crie uma classe separada para manter estas coleções de objetos e pode chamá-la de Banco.
  - Declare três atributos estáticos do tipo java.util.List nesta classe para representar estas coleções e as instancie na classe Principal.

Exercício 3

- Sempre que o item de menu rotulado como “Configurações” for pressionado ou quando o botão correspondente da barra de ferramentas for acionado, o aplicativo deve apresentar um diálogo que permita ao usuário alterar as configurações de sua interface gráfica.
  - A figura abaixo ilustra como deve ser a aparência deste diálogo.
- Este diálogo deve permitir a alteração do look and feel do sistema e definir se a barra de ferramentas deve estar visível ou oculta.
  - Sempre que for aberto, a primeira caixa de combinação deve carregar dinamicamente todos os look and feels disponíveis e o look and feel em uso deve ser selecionado automaticamente.
  - A segunda caixa de combinação também já deve estar indicando a configuração atual relativa à barra de ferramentas.

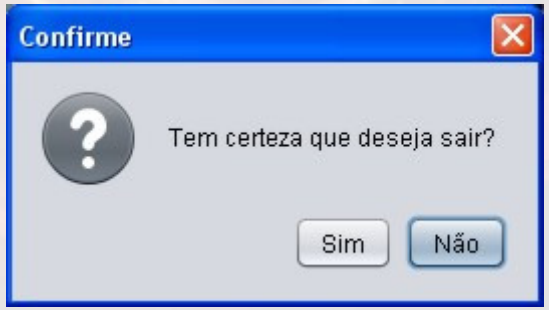
Exercício 3

- Sempre que o item de menu rotulado como “Sobre ...” for pressionado ou quando o botão correspondente da barra de ferramentas for acionado, deve ser apresentado um diálogo contendo informações acerca do sistema.
  - A figura abaixo ilustra como deve ser a aparência deste diálogo.
- Este diálogo pode ser gerado através do método showOptionDialog( ) da classe javax.swing.JOptionPane.
  - Se preferir, pode criar uma nova classe derivada da classe javax.swing.JDialog para representá-lo.



Exercício 3

- Quando o usuário acionar o item de menu rotulado como "Sair" ou quando acionar o botão correspondente da barra de tarefas, o aplicativo deve exibir um diálogo de confirmação.
  - A figura abaixo ilustra como deve ser a aparência deste diálogo.
  - Se o botão "Não" deste diálogo for pressionado, a janela principal do sistema deve ser mantida aberta.
  - Se o botão "Sim" for pressionado, a janela principal do sistema deve ser fechada e toda a memória utilizada pelo sistema deve ser liberada.



Contato

Com o autor:

Rui Rossi dos Santos
E-mail: livros@ruirossi.pro.br
Web Site: http://www.ruirossi.pro.br

Com a editora:

Editora NovaTerra
Telefone: (21) 2218-5314
Web Site: http://www.editoranovaterra.com.br