I GroupLayout

I.1 Introdução

O Java 6 SE inclui um novo gerenciador de layout poderoso chamado **`GroupLayout`**, que é o gerenciador de layout padrão no IDE do NetBeans (`www.netbeans.org`). Neste apêndice, resumimos o `GroupLayout` e então demonstramos como utilizar o **Matisse GUI designer** do IDE do NetBeans para criar uma GUI utilizando `GroupLayout` a fim de posicionar os componentes. O NetBeans gera o código `GroupLayout` automaticamente. Embora você possa escrever o código `GroupLayout` à mão, na maioria dos casos você utilizará uma ferramenta de design de GUI como aquela fornecida pelo NetBeans para tirar vantagem do poder do `GroupLayout`. Para obter mais detalhes sobre `GroupLayout`, consulte a lista de recursos Web no fim deste apêndice.

I.2 Fundamentos do GroupLayout

Os Capítulos 14 e 25 apresentaram vários gerenciadores de layout que fornecem capacidades básicas de layout de GUI. Também discutimos como combinar gerenciadores de layout e múltiplos contêineres para criar layouts mais complexos. A maioria dos gerenciadores de layout não oferece controle exato sobre o posicionamento dos componentes. No Capítulo 25, discutimos o `GridBagLayout`, que fornece controle mais preciso em relação à posição e ao tamanho dos seus componentes GUI. O `GridBagLayout` permite especificar a posição horizontal e vertical de cada componente, o número de linhas e colunas que cada componente ocupa na grade, e como os componentes aumentam e diminuem à medida que o tamanho do contêiner muda. Isso tudo é especificado ao mesmo tempo com um objeto `GridBagConstraints`. A classe `GroupLayout` é o próximo passo no gerenciamento de layouts. `GroupLayout` é mais flexível porque você pode especificar os layouts horizontais e verticais dos seus componentes de uma maneira independente.

Arranjos sequenciais e paralelos

Os componentes são organizados sequencialmente ou em paralelo. Os três `JButtons` na Figura I.1 são organizados com **orientação horizontal sequencial** — eles aparecem da esquerda para a direita na sequência. Verticalmente, os componentes são organizados em paralelo, portanto, de certo modo, eles "ocupam o mesmo espaço vertical". Componentes também podem ser organizados sequencialmente na direção vertical e em paralelo na direção horizontal, como você verá na Seção I.3. Para evitar a sobreposição de componentes, os componentes com a orientação vertical paralela normalmente são organizados na orientação horizontal sequencial (e vice-versa).

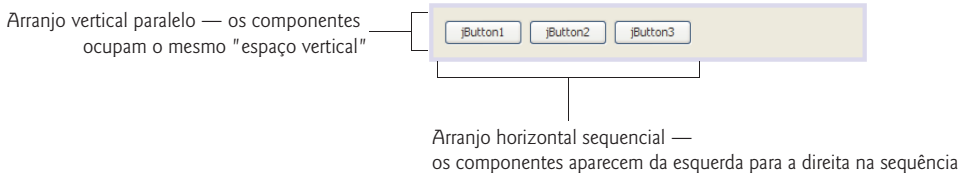

Figura I.1 | JButtons organizados sequencialmente para sua orientação horizontal e em paralelo para sua orientação vertical.

Grupos e alinhamento

Para criar interfaces com o usuário mais complexas, o `GroupLayout` permite criar **grupos** que contêm elementos sequenciais ou paralelos. Dentro de um grupo há componentes GUI, outros grupos e intervalos. O posicionamento de um grupo dentro de outro grupo é semelhante à criação de uma GUI utilizando contêineres aninhados, como um `JPanel` que contém outros `JPanel`s, que por sua vez contêm componentes GUI.

Ao criar um grupo, você pode especificar o **alinhamento** dos elementos do grupo. A classe `GroupLayout` contém quatro constantes para essa finalidade — `LEADING`, `TRAILING`, `CENTER` e `BASELINE`. A constante `BASELINE` aplica apenas orientações verticais. Na orientação horizontal, as constantes `LEADING`, `TRAILING` e `CENTER` representam alinhado à esquerda, alinhado à direita e centralizado, respectivamente. Na orientação vertical, `LEADING`, `TRAILING` e `CENTER` alinham os componentes nos centros na parte superior, inferior ou vertical, respectivamente. Alinhar componentes com `BASELINE` indica que eles devem ser alinhados utilizando a linha de base da fonte para o texto dos componentes. Para informações adicionais sobre linhas de base de fonte, consulte a Seção 15.4.

Espaçamento

O `GroupLayout` por padrão utiliza as diretrizes do design de GUI recomendadas da plataforma subjacente para o espaçamento entre componentes. O método **`addGap`** das classes `GroupLayout` aninhadas **`GroupLayout.Group`**, **`GroupLayout.SequentialGroup`** e **`GroupLayout.ParallelGroup`** permite controlar o espaçamento entre componentes.

Tamanho dos componentes

Por padrão, `GroupLayout` utiliza os métodos `getMinimumSize`, `getMaximumSize` e `getPreferredSize` de cada componente para ajudar a determinar o tamanho do componente. Você pode sobrescrever as configurações padrão.

I.3 Criando um `ColorChooser`

Agora apresentamos um aplicativo `ColorChooser` para demonstrar o gerenciador de layout `GroupLayout`. O aplicativo consiste em três objetos `JSlider`, cada um representando os valores de `0` a `255` para especificar os valores de vermelho, verde e azul de uma cor. Os valores selecionados para cada `JSlider` serão utilizados para exibir um retângulo preenchido com a cor especificada. Construímos o aplicativo utilizando o NetBeans. Para uma introdução mais detalhada ao desenvolvimento de aplicativos GUI no IDE do NetBeans, consulte `www.netbeans.org/kb/trails/matisse.html`.

Crie um novo projeto

Comece abrindo um novo projeto NetBeans. Escolha **File> New Project...**. No diálogo **New Project**, escolha **Java** na lista **Categories** e **Java Application** na lista **Projects** e então clique em **Next >**. Especifique `ColorChooser` como o nome do projeto e desmarque a caixa de seleção **Create Main Class**. Você também pode especificar a localização do seu projeto no campo **Project Location**. Clique em **Finish** para criar o projeto.

Adicione uma nova subclasse de `JFrame` ao projeto

Na guia **Projects** do IDE logo abaixo do menu e da barra de ferramentas **File** (Figura I.2), expanda o nó **Source Packages**. Clique com o botão direito do mouse no nó **<default package>** que aparece e escolha **New > JFrame Form**. No diálogo **New JFrame Form**, especifique `ColorChooser` como o nome de classe e clique em **Finish**. Essa subclasse de `JFrame` exibirá os componentes GUI do aplicativo. A janela do NetBeans agora deve ser semelhante à Figura I.3 com a classe `ColorChooser` mostrada na visualização **Design**. Os botões **Source** e **Design** na parte superior da janela `ColorChooser.java` permitem alternar entre editar o código-fonte e desenhar a GUI.

O modo **Design** só mostra a área do cliente do `ColorChooser` (isto é, a área que aparecerá dentro das margens da janela). Para construir uma GUI visualmente, arraste os componentes GUI da janela **Palette** para a área do cliente. Você pode configurar as propriedades de cada componente selecionando-o e então modificando os valores de propriedade que aparecem na janela **Properties** (Figura I.3). Ao selecionar um componente, a janela **Properties** exibe três botões — **Properties**, **Bindings**, **Events**, **Code** (ver o Figura I.4) — que permitem configurar vários aspectos do componente.

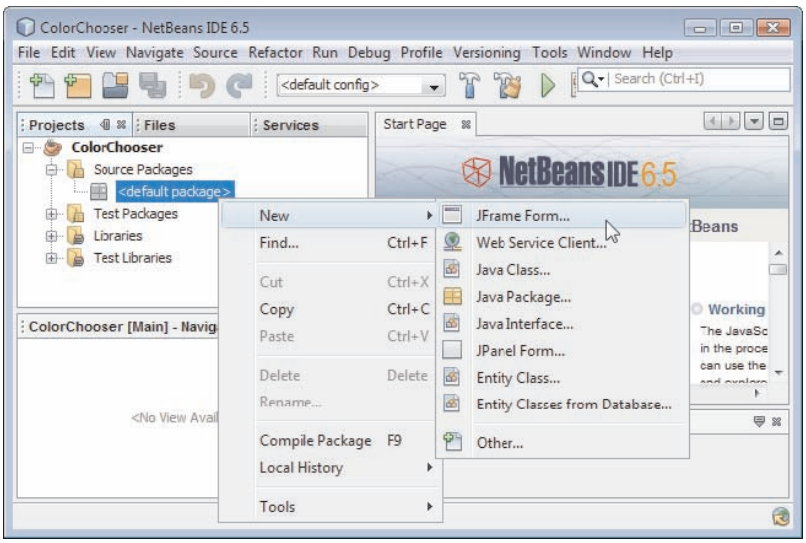

Figura I.2 | Adicionando um novo **JFrame Form** ao projeto `ColorChooser`.

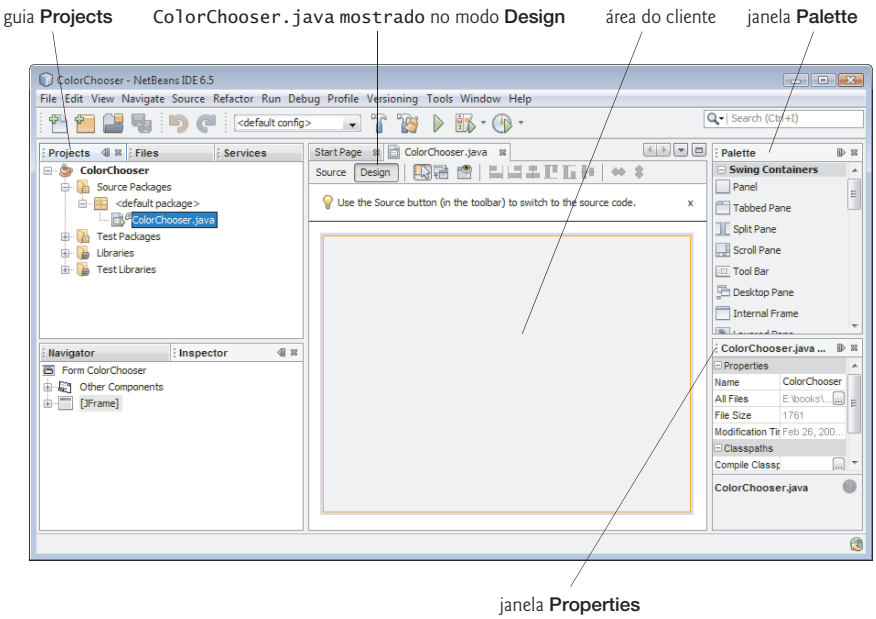

Figura I.3 | Classe `ColorChooser` mostrada na visualização **Design** do NetBeans.

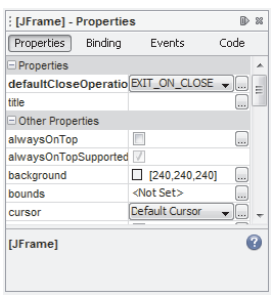

Figura I.4 | Janela **Properties** com botões que permitem configurar vários aspectos do componente.

Construa a GUI

Arraste três **Sliders** (objetos da classe `JSlider`) da **Palette** até o `JFrame` (talvez seja necessário rolar pela **Palette**). À medida que arrasta componentes perto das bordas da área do cliente ou perto de outros componentes, o NetBeans exibe **linhas-guia** (Figura I.5) que mostram as distâncias e alinhamentos recomendados entre o componente que você está arrastando, as bordas da área cliente e outros componentes.

À medida que segue os passos para construir a GUI, utilize as linhas-guia para organizar os componentes em três linhas e três colunas como na Figura I.6. Utilize a janela **Properties** para renomear os `JSliders` como `redJSlider`, `greenJSlider` e `blueJSlider`. Selecione o primeiro `JSlider`, clique no botão **Code** na janela **Properties** e altere a propriedade **Variable Name** para `redSlider`. Repita esse processo para renomear os outros dois `JSliders`. Em seguida, clique no botão **Properties** na janela **Properties**, selecione cada `JSlider` e altere sua propriedade **maximum** para 255 a fim de que ela produza valores no intervalo 0–255, e altere sua propriedade **value** para 0 a fim de que a miniatura do `JSlider` esteja inicialmente à esquerda do `JSlider`.

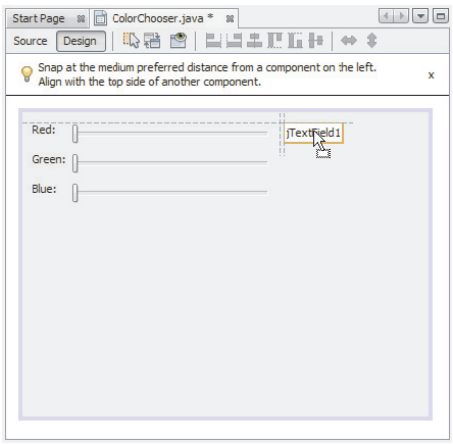

Figura I.5 | Posicionando o primeiro `JTextField`.

Figura I.6 | Layout de `JLabels`, `JSLiders` e `JTextFields`.

Arraste três **Labels** (objetos da classe `JLabel`) da **Palette** para o `JFrame` a fim de rotular cada `JSlider` com a cor que ele representa. Atribua aos `JLabels` os nomes `redJLabel`, `greenJLabel` e `blueJLabel`, respectivamente. Cada `JLabel` deve ser posicionado à esquerda do `JSlider` correspondente (Figura I.6). Altere a propriedade **text** de cada `JLabel` clicando duas vezes no `JLabel` e digitando o novo texto, ou selecionando o `JLabel` e alterando a propriedade **text** na janela **Properties**.

Adicione um **Text Field** (um objeto da classe `JTextField`) ao lado de cada um dos `JSliders` para exibir o valor do controle deslizante. Atribua aos `JTextFields` os nomes `redJTextField`, `greenJTextField` e `blueJTextField`, respectivamente. Altere cada propriedade **text** do `JTextField` para `0` a fim de utilizar as mesmas técnicas que você utilizou para os `JLabels`. Altere cada propriedade **columns** do `JTextField` para `4`.

Dê um clique duplo na margem da área de cliente para exibir o diálogo **Set Form Designer Size** e alterar o primeiro número (que representa a largura) para 410, então, clique em **OK**. Isso torna a área de cliente suficientemente larga para acomodar o **Panel** (um objeto da classe `JPanel`) que você adicionará a seguir. Por fim, adicione um **Panel** nomeado `colorJPanel` à direita desse grupo de componentes. Utilize as linhas-guia como mostrado na Figura I.7 para posicionar o `JPanel`. Altere a cor de fundo desse `JPANEL` para exibir a cor selecionada. Por fim, arraste a margem inferior da área de cliente em direção à parte superior da área **Design** até ver a linha de atração que mostra a altura recomendada da área do cliente (com base nos componentes na área de cliente) como mostrado na Figura I.8.

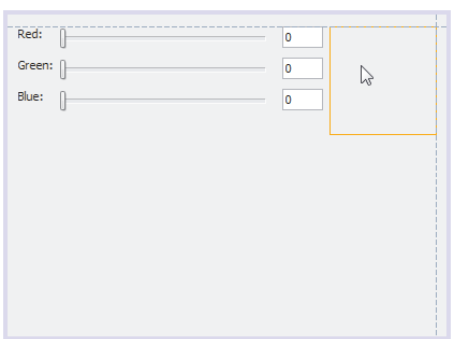

Figura I.7 | Posicionando o `JPanel`.

Figura I.8 | Configurando a altura da área do cliente.

Editando o código-fonte e adicionando handlers de evento

O IDE gerou automaticamente o código da GUI, incluindo os métodos para inicializar os componentes e alinhá-los utilizando o gerenciador de layout `GroupLayout`. Precisamos adicionar a funcionalidade desejada aos handlers de evento dos componentes. Para adicionar um handler de evento a um componente, clique com o botão direito do mouse nele e posicione o mouse sobre a opção **Events** no menu pop-up. Você pode então selecionar a categoria do evento que você deseja tratar e o evento específico dentro dessa categoria. Por exemplo, para adicionar os handlers de evento `JSlider` para esse exemplo, clique com o botão direito do mouse. Ao fazer isso, o NetBeans adiciona um `ChangeListener` ao `JSlider` e alterna da visualização **Design** para a visualização **Source**, na qual você pode inserir o código no handler de evento. Utilize o botão **Design** para retornar à visualização **Design** e repita os passos anteriores para adicionar os handlers de evento para os outros dois `JSliders`. Para completar os handlers de evento, primeiro adicione o método na Figura I.9. Em cada handler de evento `JSlider` configure o `JTextField` correspondente com o novo valor do `JSlider` e então chame o método `changeColor`. Por fim, no construtor depois da chamada a `initComponents`, adicione a linha

```
colorJPanel.setBackground( java.awt.Color.BLACK );
```

A Figura I.10 mostra a classe `ColorChooser` completada da maneira como é gerada no NetBeans. Excepcionalmente aqui, não reformatamos o código para corresponder com as convenções de codificação vistas por todo este livro.

```
1  // altera a cor de fundo do colorJPanel com base nos valores atuais
2  // dos JSliders
3  public void changeColor()
4  {
5     colorJPanel.setBackground( new java.awt.Color(
6        redJSlider.getValue(), greenJSlider.getValue(),
7        blueJSlider.getValue() ) );
8  } // fim do método changeColor
```

Figura I.9 | O método que altera a cor de fundo do `colorJPanel` com base nos valores dos três `JSliders`.

```
1  /*
2   * ColorChooser.java
3   *
4   * Criado em 26 de fevereiro de 2009, 11:05:35
5   */
6  /**
7   *
8   * @author paul
9   */
10 public class ColorChooser extends javax.swing.JFrame
11 {
12    /** Cria o novo formulário ColorChooser */
13    public ColorChooser()
14    {
15       initComponents();
16       colorJPanel.setBackground( java.awt.Color.BLACK );
17    }
18
19    // altera a cor de fundo do colorJPanel com base nos valores atuais
20    // dos JSliders
21    public void changeColor()
22    {
23       colorJPanel.setBackground( new java.awt.Color(
24          redJSlider.getValue(), greenJSlider.getValue(),
```

```
25              blueJSlider.getValue() ) );
26          } // fim do método changeColor
27
28          / ** Esse método é chamado a partir de dentro do construtor para
29           * inicializar o formulário.
30           * ATENÇÃO: NÃO modifique esse código. O conteúdo desse método é
31           * sempre regenerado pelo Form Editor.
32           */
33          @SuppressWarnings("unchecked")
34           // <editor-fold defaultstate="collapsed" desc="Generated Code">
35           private void initComponents() {
36
37              redJSlider = new javax.swing.JSlider();
38              greenJSlider = new javax.swing.JSlider();
39              blueJSlider = new javax.swing.JSlider();
40              redjLabel = new javax.swing.JLabel();
41              greenJLabel = new javax.swing.JLabel();
42              blueJLabel = new javax.swing.JLabel();
43              redJTextField = new javax.swing.JTextField();
44              greenJTextField = new javax.swing.JTextField();
45              blueJTextField = new javax.swing.JTextField();
46              colorJPanel = new javax.swing.JPanel();
47
48              setDefaultCloseOperation( javax.swing.WindowConstants.EXIT_ON_CLOSE);
49
50              redJSlider.setMaximum(255);
51              redJSlider.setValue(0);
52              redJSlider.addChangeListener(new javax.swing.event.ChangeListener() {
53                  public void stateChanged(javax.swing.event.ChangeEvent evt) {
54                      redJSliderStateChanged(evt);
55                  }
56              });
57
58              greenJSlider.setMaximum(255);
59              greenJSlider.setValue(0);
60              greenJSlider.addChangeListener(new javax.swing.event.ChangeListener() {
61                  public void stateChanged(javax.swing.event.ChangeEvent evt) {
62                      greenJSliderStateChanged(evt);
63                  }
64              });
65
66              blueJSlider.setMaximum(255);
67              blueJSlider.setValue(0);
68              blueJSlider.addChangeListener(new javax.swing.event.ChangeListener() {
69                  public void stateChanged(javax.swing.event.ChangeEvent evt) {
70                      blueJSliderStateChanged(evt);
71                  }
72              });
73
74              redjLabel.setText("Red:");
75
76              greenJLabel.setText("Green:");
77
78              blueJLabel.setText("Blue:");
79
80              redJTextField.setColumns(4);
81              redJTextField.setText("0");
82
83              greenJTextField.setColumns(4);
84              greenJTextField.setText("0");
85
86              blueJTextField.setColumns(4);
87              blueJTextField.setText("0");
88
89              javax.swing.GroupLayout colorJPanelLayout = new javax.swing.GroupLayout(colorJPanel);
90              colorJPanel.setLayout(colorJPanelLayout);
91              colorJPanelLayout.setHorizontalGroup(
92                  colorJPanelLayout.createParallelGroup( javax.swing.GroupLayout.Alignment.LEADING)
93                  .addGap(0, 100, Short.MAX_VALUE)
```

```
94            );
95            colorJPanelLayout.setVerticalGroup(
96                colorJPanelLayout.createParallelGroup( javax.swing.GroupLayout.Alignment.LEADING)
97                .addGap(0, 100, Short.MAX_VALUE)
98            );
99
100           javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
101           getContentPane().setLayout(layout);
102           layout.setHorizontalGroup(
103               layout.createParallelGroup( javax.swing.GroupLayout.Alignment.LEADING)
104               .addGroup(layout.createSequentialGroup()
105                   .addContainerGap()
106                   .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
107                       .addComponent(greenJLabel)
108                       .addComponent(blueJLabel)
109                       .addComponent(redjLabel))
110                   .addPreferredGap( javax.swing.LayoutStyle.ComponentPlacement.RELATED)
111                   .addGroup(layout.createParallelGroup( javax.swing.GroupLayout.Alignment.LEADING)
112                       .addGroup(layout.createSequentialGroup()
113                           .addComponent(blueJSlider, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
114                           .addPreferredGap( javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
115                           .addComponent(blueJTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
116                       .addGroup(layout.createSequentialGroup()
117                           .addComponent(greenJSlider, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
118                           .addPreferredGap( javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
119                           .addComponent(greenJTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
120                       .addGroup(layout.createSequentialGroup()
121                           .addComponent(redJSlider, javax.swing.GroupLayout.PREFERRED_SIZE, javax.
swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
122                           .addPreferredGap( javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
123                           .addComponent(redJTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)))
124                   .addPreferredGap( javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.
swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
125                   .addComponent(colorJPanel, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.
GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
126                   .addContainerGap())
127           );
128           layout.setVerticalGroup(
129               layout.createParallelGroup( javax.swing.GroupLayout.Alignment.LEADING)
130               .addGroup(layout.createSequentialGroup()
131                   .addContainerGap()
132                   .addGroup(layout.createParallelGroup( javax.swing.GroupLayout.Alignment.LEADING)
133                       .addComponent(colorJPanel, javax.swing.GroupLayout.PREFERRED_SIZE, javax.
swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
134                       .addGroup(layout.createSequentialGroup()
135                           .addGroup(layout.createParallelGroup( javax.swing.GroupLayout.Alignment.
LEADING)
136                               .addComponent(redJSlider, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
137                               .addComponent(redjLabel)
138                               .addComponent(redJTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
139                           .addPreferredGap( javax.swing.LayoutStyle.ComponentPlacement.RELATED)
140                           .addGroup(layout.createParallelGroup( javax.swing.GroupLayout.Alignment.
LEADING)
141                               .addComponent(greenJSlider, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
142                               .addComponent(greenJLabel)
143                               .addComponent(greenJTextField, javax.swing.GroupLayout.PREFERRED_
SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
144                           .addPreferredGap( javax.swing.LayoutStyle.ComponentPlacement.RELATED)
145                           .addGroup(layout.createParallelGroup( javax.swing.GroupLayout.Alignment.
LEADING)
146                               .addComponent(blueJTextField, javax.swing.GroupLayout.PREFERRED_
```

```
SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
147                                 .addComponent(blueJSlider, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
148                                 .addComponent(blueJLabel))))
149                     .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
150             );
151
152             pack();
153         }// </editor-fold>
154
155         private void redJSliderStateChanged(javax.swing.event.ChangeEvent  evt) {
156            redJTextField.setText( String.valueOf( redJSlider.getValue() ) );
157            changeColor();
158         }
159
160         private void greenJSliderStateChanged(javax.swing.event.ChangeEvent evt) {
161            greenJTextField.setText( String.valueOf( greenJSlider.getValue() ) );
162            changeColor();
163         }
164
165         private void blueJSliderStateChanged(javax.swing.event.ChangeEvent evt) {
166            blueJTextField.setText( String.valueOf( blueJSlider.getValue() ) );
167            changeColor();
168         }
169
170        /**
171         * argumentos de linha de comando @param
172         */
173        public static void main( String args[] )
174        {
175           java.awt.EventQueue.invokeLater( new Runnable()
176           {
177              public void run()
178              {
179                 new ColorChooser().setVisible( true );
180              }
181
182           } );
183        }
184
185         // Declaração de variáveis - não modifique
186         private javax.swing.JLabel blueJLabel;
187         private javax.swing.JSlider blueJSlider;
188         private javax.swing.JTextField blueJTextField;
189         private javax.swing.JPanel colorJPanel;
190         private javax.swing.JLabel greenJLabel;
191         private javax.swing.JSlider greenJSlider;
192         private javax.swing.JTextField greenJTextField;
193         private javax.swing.JSlider redJSlider;
194         private javax.swing.JTextField redJTextField;
195         private javax.swing.JLabel redjLabel;
196         // Fim da declaração de variáveis
197     }
```

Figura I.10 | A classe `ColorChooser` que utiliza `GroupLayout` para seu layout da GUI.

O método `initComponents` (linhas 35–153) foi inteiramente gerado pelo NetBeans com base nas suas interações com o designer da GUI. Esse método contém o código que cria e formata a GUI. As linhas 37–87 constroem e inicializam os componentes GUI. As linhas 89–153 especificam o layout desses componentes utilizando `GroupLayout`. As linhas 102–127 especificam o grupo horizontal e as linhas 128–150, o grupo vertical. Observe a complexidade desse código. Cada vez mais o desenvolvimento de softwares é feito com ferramentas que geram códigos complexos como esse, facilitando e agilizando esse processo para você.

Adicionamos manualmente a instrução que altera a cor de fundo do `colorJPanel` na linha 16 e o método `changeColor` nas linhas 21–26. Quando o usuário move o polegar em um dos `JSliders`, o handler de evento do `JSLIDER` especifica o texto no seu `JTextField` correspondente ao novo valor do `JSLIDER` (linhas 156, 161 e 166) e então chama o método `changeColor` (linhas 157, 162 e 167) para atualizar a cor de fundo do `colorJPanel`. O método `changeColor` obtém o valor atual de cada `JSlider` (linhas 24–25) e utiliza esses valores como os argumentos para o construtor `Color` a fim de criar um novo `Color`.

I.4 Recursos Web GroupLayout

`weblogs.java.net/blog/tpavek/archive/2006/02/getting_to_know_1.html`
A Parte 1 das postagens no blog `GroupLayout` de Tomas Pavek oferece uma visão geral da teoria por trás do `GroupLayout`.

`weblogs.java.net/blog/tpavek/archive/2006/03/getting_to_know.html`
A Parte 2 das postagens no blog `GroupLayout` de Tomas Pavek apresenta uma GUI completa implementada com o `GroupLayout`.

`java.sun.com/javase/6/docs/api/javax/swing/GroupLayout.html`
Documentação da API para a classe `GroupLayout`.

`wiki.java.net/bin/view/Javadesktop/GroupLayoutExample`
Fornece uma demonstração do Address Book de uma GUI construída manualmente com o `GroupLayout` com o código-fonte.

`weblogs.java.net/blog/claudio/archive/nb-layouts.html`
Tutorial sobre o `GroupLayout` baseado em Flash.

`www.developer.com/java/ent/article.php/3589961`
Tutorial: "Building Java GUIs with Matisse: A Gentle Introduction", por Dick Wall.

`myeclipseide.com/enterpriseworkbench/help/index.jsp?topic=/com.genuitec.eclipse.dehory.doc/doc/quickstart/index.html`
Tutorial: "Matisse4MyEclipse—Swing RCP Development Quickstart," de MyEclipse. Introduz uma versão do designer de GUI Matisse para o IDE do Eclipse.