```
// Fig. 13.5: ColorJPanel.java
// Changing drawing colors.

import java.awt.Graphics;
```

// A linha de código `import java.awt.Graphics;` é uma instrução de importação na linguagem de programação Java. Ela importa a classe `Graphics` do pacote `java.awt`. A classe `Graphics` fornece métodos para desenhar gráficos em componentes Java, como janelas e painéis//

```
import java.awt.Color;
import javax.swing.JPanel;

public class ColorJPanel extends JPanel
```

//Esta linha de código define uma classe chamada ColorJPanel que herda da classe JPanel. Isso significa que ColorJPanel é uma subclasse de JPanel e herda todas as suas propriedades e métodos//

//public: Esta palavra-chave especifica que a classe ColorJPanel pode ser acessada a partir de qualquer outra classe no mesmo pacote ou em qualquer subpacote//

//class: Esta palavra-chave é usada para definir uma classe em Java//

//ColorJPanel: Este é o nome da classe. É uma convenção em Java usar CamelCase para nomes de classes, onde a primeira letra de cada palavra é maiúscula//

//Estende (herda): esta palavra-chave indica que a classe ColorJPanel é uma subclasse da classe JPanel. Isso significa que ColorJPanel herda todas as propriedades e métodos do JPanel//

//JPanel: Este é o nome da superclasse, que neste caso é JPanel//

```
{
   // draw rectangles and Strings in different colors

   @Override
```

//A palavra-chave override em Java é usada para indicar que um método em uma subclasse está redefinindo a implementação de um método herdado de sua superclasse//
//subescreve o método//

```
   public void paintComponent(Graphics g)
```

// public**:** Este modificador de acesso torna o método acessível de qualquer outra classe no mesmo pacote ou subpacote//

//void**:** Indica que o método não retorna nenhum valor//

// paintComponent**:** O nome do método. Por convenção, usa-se CamelCase para nomes de métodos//

//(Graphics g)**:** O parâmetro do método. Ele é do tipo Graphics e representa o objeto usado para desenhar na tela. Através desse objeto, você pode acessar vários métodos para desenhar linhas, formas, texto e imagens//

```
{
    super.paintComponent(g);
```

// A linha de código super.paintComponent(g); em Java é uma chamada ao método paintComponent() da superclasse imediata da classe atual//

```
    this.setBackground(Color.WHITE);
```

// this: Esta palavra-chave refere-se à instância atual do objeto no qual o método está sendo chamado. Neste caso, refere-se ao componente Swing cuja cor de fundo está sendo definida//

//.setBackground(): Esta é uma chamada de método neste objeto. O método setBackground() é fornecido pela classe JComponent, que é a superclasse de muitos componentes Swing comuns, como JPanel, JButton e JLabel//

//(Color.WHITE): Este é o argumento passado para o método setBackground(). Especifica a cor de fundo desejada, que neste caso é Color.WHITE, representando a cor branca//

//Exemplo: Color.WHITE – o ponto(.) é um concatenador entre objeto//

```
    // set new drawing color using integers

    g.setColor(new Color(255, 0, 0));
```

// g: Representa o objeto Graphics, que é passado como parâmetro para o método paintComponent(). O objeto Graphics fornece métodos para desenhar vários elementos gráficos em um componente//

//.setColor(): Esta é uma chamada de método no objeto Graphics g. O método setColor() é usado para definir a cor do desenho atual para operações de desenho subsequentes//

//(new Color(255, 0, 0)): Este é o argumento passado para o método setColor(). Ele cria um novo objeto Color com os valores vermelhos, verdes e azuis (RGB) especificados. Neste caso, os valores são (255, 0, 0), que representam a cor vermelha.//

//r – red (vermelho), g – green (verde), b – blue (azul)//

```
      g.fillRect(15, 25, 100, 20);
```

```
//fillRecct – objeto de linhas retas//
```

```
      g.drawString("Current RGB: " + g.getColor(), 130, 40);
```

//sinal de +, concatenador de duas palavras//

```
      // set new drawing color using floats
      g.setColor(new Color(0.50f, 0.75f, 0.0f));
      g.fillRect(15, 50, 100, 20);
      g.drawString("Current RGB: " + g.getColor(), 130, 65);

      // set new drawing color using static Color objects
      g.setColor(Color.BLUE);
      g.fillRect(15, 75, 100, 20);
      g.drawString("Current RGB: " + g.getColor(), 130, 90);

      // display individual RGB values
      Color color = Color.MAGENTA;
      g.setColor(color);
      g.fillRect(15, 100, 100, 20);
      g.drawString("RGB values: " + color.getRed() + ", " +
         color.getGreen() + ", " + color.getBlue(), 130, 115);
   }
} // end class ColorJPanel
```