计算机科学丛书

原书第4版

Java How to Program

(Fourth Edition)

Java程序设计教程

(美) Harvey M. Deitel Paul J. Deitel 著 袁兆山 刘宗田 苗沛荣 等译

下册：提高篇

利用UML和设计模式
介绍面向对象的程序设计

机械工业出版社
China Machine Press

附光盘壹张

本书详细介绍Java面向对象程序设计的基本理论及实用知识，全书分为上下两册。下册详细讲述Java程序设计的高级主题，共有12章，主要介绍图形与Java 2D、图形用户界面、异常处理、多线程、文件和流、网络、多媒体、数据结构、Java工具包与位操作、集合、Java媒体框架和Java Sound等内容。

本书实用性强，以多个案例分析为中心，每章都提供了丰富的示例、练习和项目。

本书适合作为高等院校的Java语言课程教材，也可作为各种Java语言培训班的教材。

随书光盘包含全书上下册的实例代码，及其他辅助内容。第22章和附录中的部分内容也在光盘中。

本书版权登记号：图字：01-2001-4781

图书在版编目（CIP）数据

Java程序设计教程（原书第4版 下册·提高篇）/（美）戴特（Deitel, H. M.），（美）戴特（Deitel, P. J.）著；袁兆山等译. -北京：机械工业出版社，2004.9

（计算机科学丛书）

书名原文：Java How to Program (Fourth Edition)

ISBN 7-111-14701-4

Ⅰ.J… Ⅱ.①戴…②戴…③袁… Ⅲ.Java语言－程序设计－教材 Ⅳ.TP312

中国版本图书馆CIP数据核字（2004）第058341号

机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）

责任编辑：李 英

北京牛山世兴印刷厂印刷·新华书店北京发行所发行

2004年9月第1版第1次印刷

787mm × 1092mm 1/16 · 42.75印张

印数：0 001-4 000册

定价：69.00元（附光盘）

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线（010）68326294

目　录

第11章　图形和Java2D

目标

- 理解图形环境和图形对象。
- 理解并能够控制颜色。
- 理解并能够控制字体。
- 理解并能够使用Graphics方法绘制线条、矩形、圆角矩形、三维矩形、椭圆、弧和多边形。
- 能够用Java2D API中 Graphics2D类的方法绘制线条、矩形、圆角矩形、椭圆、弧和多边形。
- 能够指定Graphics2D的Paint和Stroke属性来显示图形。

百闻不如一见。

——中国名谚

从不同角度看，大自然可被视为圆柱体、球体、圆锥体。

——保罗·塞尚

没有经历的事千万别相信——甚至包括谚语，除非在生活中亲历过。

——约翰·济慈

一幅画向我传递了一本书的数页纸的内容。

——伊凡·谢尔戈维奇·屠格涅夫

提纲

11.1 简介

本章概述Java中关于绘制二维图形、控制颜色和字体的一些功能。Java最初的一个吸引人之处是它支持图形，使Java程序员能够增强applet和应用程序在可视化方面的功能。作为Java2D API的一部分，Java现在具有大量更为成熟的绘图功能。本章先介绍一些简单的Java绘图功能，然后介绍Java2D中一些新的和高级的功能，例如绘制图形时控制线条的样式，以及如何在图形中填充色彩和图案等。

图11-1显示了Java类层次结构的一部分，其中包括本章中涉及的一些基本的图形类和Java2D API中的类和接口。Color类包含了控制颜色的方法和常量；Font类包含了控制字体的方法和常量；FontMetrics类包含了获取字体信息的方法；Polygon类包含了创建多边形的方法；Graphics类包含了绘制字符串、线条、矩形及其他形状的方法。图11-1下半部分的一些类和接口来自Java2D API，BasicStroke类决定所绘线条的特征。GradientPaint和TexturePaint帮助决定填充图形的色彩或图案特征。而GeneralPath、Arc2D、Ellipse2D、Line2D、Rectangle2D和RoundRectangle2D定义了各种Java2D形状。

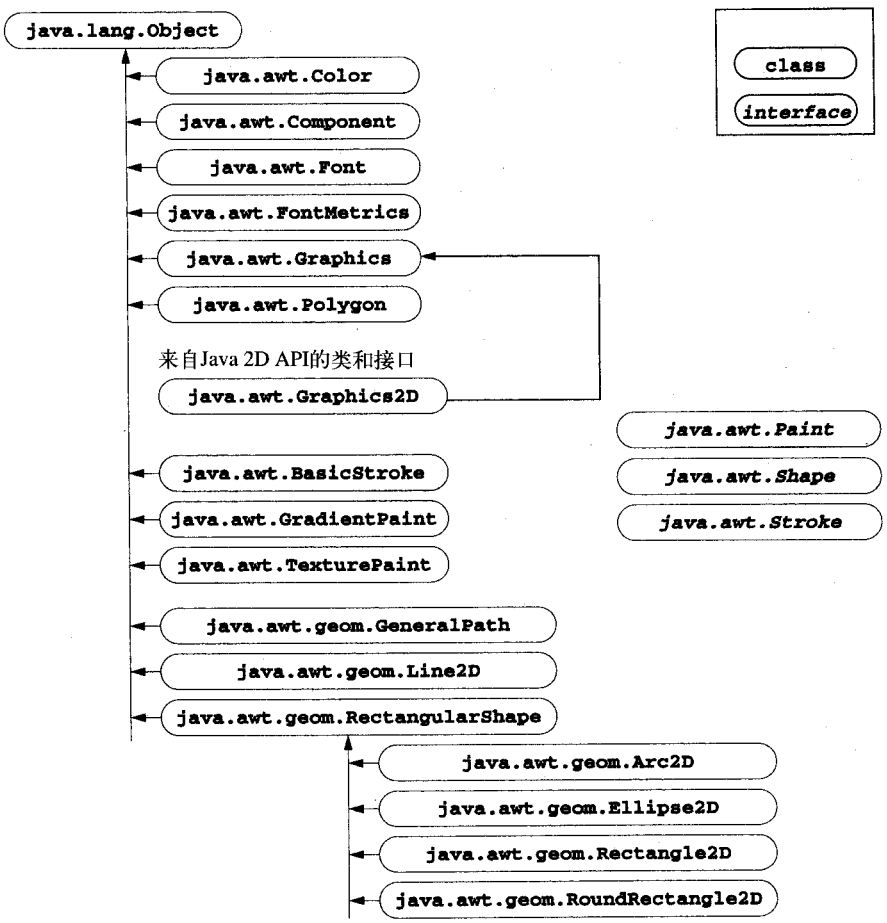

图11-1 Java类层次结构的一部分，其中包括本章中涉及的一些基本的图形功能和Java 2D API 中的类

用Java绘图，必须首先理解（图11-2）Java坐标系统。这是一种对屏幕上每个点进行定位的方案。默认时，GUI组件（例如：applet和窗口）左上角的坐标为（0，0）。坐标由一个x坐标（横坐标）和一个y坐标（垂直坐标）组成。x坐标是从左上角向右移动的水平距离，y坐标是从左上角向下移动的垂直距离。x轴描述了横坐标，y轴描述了垂直坐标。

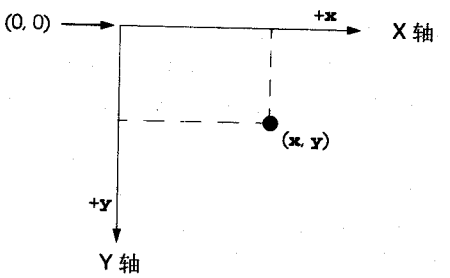

图11-2 以像素为单位的Java坐标系统


软件工程评述11.1 由于窗口的左上角坐标（0，0）实际上是指标题栏后面的最左上角，所以使用坐标绘制图形时应该调整它到窗口边框的里面。Container类（Java中所有窗口类的超类）中有一个专门针对该目的方法getInsets，它能返回一个Insets对象（java.awt包）。Insets对象中有4个public成员——top、bottom、left和right，它们代表了窗口中可绘制区域的四条边的像素数。

通过指定坐标可在屏幕上显示文本和形状。坐标单位为像素，它是显示器分辨率的最小单元。


可移植性提示11.1 不同的显示器有不同的分辨率（即像素的密度不同）。这可能导致相同的图形在不同的显示器上有不同的大小。

11.2 图形环境和图形对象

图形环境的使用使Java可以在屏幕上绘图。图形对象通过控制如何绘制图形来管理图形环境。图形对象包含绘图、字体控制、颜色控制等方法。本书中所有在屏幕上绘图的applet都使用了Graphics对象g（applet中paint方法的参数）来管理applet的图形环境。本章用应用程序来说明绘图，但在此出现的每种技术都可以在applet中使用。

Graphics类是个抽象类（因为Graphics对象不能被实例化），这是由Java的可移植性决定的。在支持Java的各个平台上，绘图的实现是不同的，还没有一个类能在所有系统上实现绘图功能。例如，运行微软Windows的PC绘制矩形的方法与UNIX工作站绘制矩形的方法是不同的，而它们与Macintosh绘制矩形图形的方法也不相同。在各种平台上使用Java时，系统创建Graphics的一个派生类，由它来实际实现所有的绘图性能。Graphics类将这种实现隐藏起来，使我们能够以独立于平台的方式编写使用图形的程序。

Component类是java.awt包中许多类的超类（第12章将详细讨论Component类）。Component类的paint方法以一个Graphics对象作为参数。在Component执行paint操作时，系统将Graphics对象传送给paint方法。paint方法的头部为：

```
public void paint (Graphics g)
```

Graphics对象g接收一个引用，该引用指向系统派生的Graphics类的一个对象。读者应该比较熟悉paint方法的头部，因为applet一直在使用它。实际上，Component类是JApplet类的一个间接基类——JApplet是本书中所有applet的超类，而JApplet类的很多功能又是从Component类继承的。Component类中定义的paint方法在默认时不做任何事情——编

程人员必须编程覆盖它。

程序员很少直接调用paint方法，因为绘制图形是一个事件驱动过程。在开始执行applet时，程序会自动地调用paint方法（在调用JApplet类的init和start方法之后）。为了再次调用paint，必须产生一个事件，例如掩盖applet或取消掩盖。类似地，当任何Component被显示时，Component的paint方法被调用。

如果程序员需要调用paint，系统便会调用Component类的repaint方法。repaint方法立即调用Component类的update方法以清除Component背景中以前的绘图，然后update方法直接调用paint。repaint方法经常被直接调用，以便强制进行paint操作。不应覆盖repaint方法，因为该方法执行一些依赖于系统的任务。update方法很少被直接调用，但有时被覆盖。正如在第18章中可以看到的，覆盖update方法对“平滑”动画（即减少“闪烁”）很有用。repaint和update的头部为:

```
public void repaint ()
public void update (Graphics g)
```

方法update用一个Graphics对象作为参数，当方法update被调用时，系统自动提供该对象。

本章重点讨论paint方法。下一章介绍图形的事件驱动性质，并详细讨论repaint和update方法。我们也讨论JComponent类，javax.swing包中大多数GUI组件的超类。JComponent的子类通常用它们的paintComponent方法绘图。

11.3 颜色控制

颜色美化了程序的外观并且有助于传达信息。例如，交通信号灯有3种不同颜色——红色表示停止；黄色表示注意；绿色表示通行。

Color类定义了控制Java程序中颜色的方法和常量。图11-3总结了预定义的颜色常量，图11-4则总结了颜色的方法和构造函数。注意，图11-4中有两种方法是专门用于颜色控制的Graphics方法。

颜色常量	颜　色	RGB值
public final static Color orange	橙色	255, 200, 0
public final static Color pink	粉红色	255, 175, 175
public final static Color cyan	青色	0, 255, 255
public final static Color magenta	紫红色	255, 0, 255
public final static Color yellow	黄色	255, 255, 0
public final static Color black	黑色	0, 0, 0
public final static Color white	白色	255, 255, 255
public final static Color gray	灰色	128, 128, 128
public final static Color lightGray	浅灰色	192, 192, 192
public final static Color darkGray	深灰色	64, 64, 64
public final static Color red	红色	255, 0, 0
public final static Color green	绿色	0, 255, 0
public final static Color blue	蓝色	0, 0, 255

图11-3 Color类的static常量和RGB值

每种颜色都是由红、绿和蓝三种颜色构成的。这三种组成部分称为RGB值。一个RGB值

有三个部分，这三个部分都是0～255范围内的整数，或0.0～1.0范围内的浮点数。第一部分定义了红色的含量；第二部分定义了绿色的含量；第三部分则定义了蓝色的含量。RGB值越大，相应颜色的含量就越大。Java允许程序员在256×256×256（大约是1670万）种颜色中进行选择，但不是所有的计算机都能显示所有这些颜色。如果情况的确如此，计算机就会显示出最接近的颜色。

方　法	说　明
`public Color (int r, int g, int b)`	在红、绿、蓝颜色含量的基础上创建一种颜色，且每种含量都用0～255之间的一个整数来表示
`public Color (float r, float g, float b)`	在红、绿、蓝颜色含量的基础上创建一种颜色，且每种含量都用0.0～1.0之间的一个浮点数来表示
`public int getRed ()  // Color类`	返回一个0～255之间表示红色含量的值
`public int getBlue ()  // Color类`	返回一个0～255之间表示蓝色含量的值
`public int getGreen ()  // Color类`	返回一个0～255之间表示绿色含量的值
`public Color getColor ()  //  Graphics类`	返回一个代表图形环境当前颜色的Color对象
`public void setColor (Color c)  //  Graphics类`	设置图形环境当前的绘图颜色

图11-4 Color的方法和与颜色相关的Graphics方法

常见编程错误11.1 将static Color类常量以大写字母开头是一种语法错误。

图11-4给出了两种Color构造函数，一种具有3个int参数，另一种具有3个float参数，每个参数分别指定了红色、绿色和蓝色的含量。记住，int值在0～255之间，float值在0.0～1.0之间。所构造的Color对象具有指定的红色、绿色和蓝色含量。Color类的getRed、getGreen和getBlue方法返回0～255之间的值，分别代表红色、绿色和蓝色的含量。Graphics类的getColor方法返回一个表示当前绘图颜色的Color对象。Graphics类方法setColor设置当前绘图颜色。

图11-5的程序通过使用几种不同的颜色绘制填充矩形和字符串，来说明图11-4中的一些方法。

当这个程序开始运行时，调用ShowColor类的paint方法（第23～第49行）绘制一个窗口。程序的第29行使用Graphics中的setColor方法设置当前颜色。setColor方法接受一个Color对象作为参数。表达式new Color(255,0,0)生成一个表示红色的Color对象（红色值为255，绿色、蓝色值都为0）。第30行使用Graphics中的fillRect方法，用当前颜色绘制一个填充矩形。fillRect和drawRect方法（见第3章）使用同样的参数。程序的第31行使用Graphics的drawString方法并用当前颜色绘制一个字符串。表达式g.getColor()返回Graphics对象的当前颜色。Color返回值调用Color类的toString方法，再连接到字符串“Current RGB:”后面。注意，Color对象的String表示包含了类名、包名（java.awt.Color）和红、绿、蓝的值。

```
1  // Fig. 11.5: ShowColors.java
2  // Demonstrating Colors.
3
4  // Java core packages
5  import java.awt.*;
6  import java.awt.event.*;
7
8  // Java extension packages
9  import javax.swing.*;
10
11 public class ShowColors extends JFrame {
12
13    // constructor sets window's title bar string and dimensions
14    public ShowColors()
15    {
16       super( "Using colors" );
17
18       setSize( 400, 130 );
19       setVisible( true );
20    }
21
22    // draw rectangles and Strings in different colors
23    public void paint( Graphics g )
24    {
25       // call superclass's paint method
26       super.paint( g );
27
28       // set new drawing color using integers
29       g.setColor( new Color( 255, 0, 0 ) );
30       g.fillRect( 25, 25, 100, 20 );
31       g.drawString( "Current RGB: " + g.getColor(), 130, 40 );
32
33       // set new drawing color using floats
34       g.setColor( new Color( 0.0f, 1.0f, 0.0f ) );
35       g.fillRect( 25, 50, 100, 20 );
36       g.drawString( "Current RGB: " + g.getColor(), 130, 65 );
37
38       // set new drawing color using static Color objects
39       g.setColor( Color.blue );
40       g.fillRect( 25, 75, 100, 20 );
41       g.drawString( "Current RGB: " + g.getColor(), 130, 90 );
42
43       // display individual RGB values
44       Color color = Color.magenta;
45       g.setColor( color );
46       g.fillRect( 25, 100, 100, 20 );
47       g.drawString( "RGB values: " + color.getRed() + ", " +
48          color.getGreen() + ", " + color.getBlue(), 130, 115 );
49    }
50
51    // execute application
52    public static void main( String args[] )
53    {
54       ShowColors application = new ShowColors();
55
56       application.setDefaultCloseOperation(
57          JFrame.EXIT_ON_CLOSE );
58    }
59
60 }  // end class ShowColors
```

图11-5 演示设置和获取Color对象

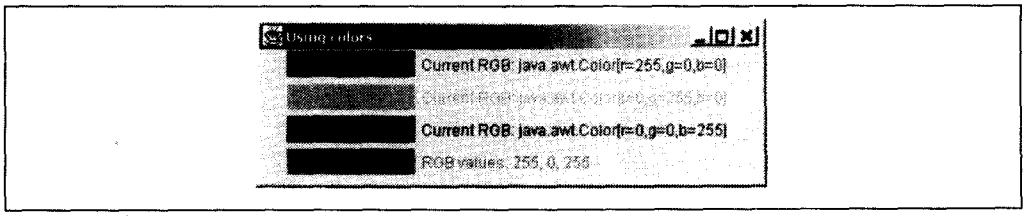

图11-5 （续）

程序的第34～第36行和第39～第41行再次进行了同样的工作。程序的第34行使用Color构造函数，并通过3个float参数来生成绿色（0.0f为红色值，1.0f为绿色值，0.0f为蓝色值）。注意常量的语法形式。浮点常数后增加字母f表示该常数作为float类型对待。通常，浮点常数作为double类型对待。

在程序的第39行设置的颜色为预定义的Color常量（Color.blue）。注意，无需使用new操作符生成常量。原因是Color常量是static的，程序运行时Color类一旦被载入内存，就定义这些常量。

在程序的第47行和第48行中，Color的getRed、getGreen和getBlue方法分别获得预定义的Color.magenta对象的红、绿、蓝值。

注意在main方法中的第56和第57行用JFrame类的setDefaultCloseOperation方法指定用户在关闭运行程序窗口时的默认动作。本例中，指定JFrame.EXIT_ON_CLOSE以期在用户点击关闭窗口时能终止程序。其他选项如DO_NOTHING_ON_CLOSE（忽略窗口关闭事件），HIDE_ON_CLOSE（隐藏窗口，以后可以重新显示），DISPOSE_ON_CLOSE（关闭窗口，以后也不再显示）。基于这点考虑，只在用户关闭窗口时程序还需执行其他附加功能时，使用WindowListener，否则，将在用户关闭窗口时使用setDefaultCloseOperation来指定终止程序。

软件工程评述11.2 要改变颜色，必须创建新的Color对象（或使用预定义的Color常量），因为在Color类中没有改变当前颜色的方法。

Java的一个新功能是预定义了一个选择颜色的GUI控件JColorChooser（javax.swing包）。图11-6中的程序可以通过按一个按钮来显示JColorChooser对话框。当选择一种颜色，并且按下对话框的OK按钮时，程序窗口中的背景颜色就会改变。

程序的第35～第36行（改变颜色的actionPerformed方法）使用JColorChooser类中的static方法showDialog来显示颜色选择对话框。该方法返回被选择的Color对象（如果按下Cancel按钮，或没有按下OK按钮就关闭了对话框，则返回null）。

showDialog方法有3个参数，一个引用指向其父组件component，String为对话框标题栏中显示的信息，还有一个参数为该对话框中所选择的初始颜色。父组件是指显示该对话框的窗口，当颜色选择对话框在屏幕上显示时，用户不能与其父组件交互。这种类型的对话框叫“模态对话框”，我们将在第13章说明它。注意ShowColors2.this的特殊语法。在使用内部类时，你能通过用外部类名和点操作符（.）限定this来访问外部类对象的this引用。

在用户选择了颜色后，程序的第39～第40行判断Color是否为null，如果是null，则设置Color为默认值Color.lightGray。第43行用setBackground方法来改变内容面板的背景色（在本程序中由container表示）。setBackground方法是众多Component方法中的一个，大多数GUI组件都可使用它。

```
1  // Fig. 11.6: ShowColors2.java
2  // Demonstrating JColorChooser.
3
4  // Java core packages
5  import java.awt.*;
6  import java.awt.event.*;
7
8  // Java extension packages
9  import javax.swing.*;
10
11 public class ShowColors2 extends JFrame {
12    private JButton changeColorButton;
13    private Color color = Color.lightGray;
14    private Container container;
15
16    // set up GUI
17    public ShowColors2()
18    {
19       super( "Using JColorChooser" );
20
21       container = getContentPane();
22       container.setLayout( new FlowLayout() );
23
24       // set up changeColorButton and register its event handler
25       changeColorButton = new JButton( "Change Color" );
26
27       changeColorButton.addActionListener(
28
29          // anonymous inner class
30          new ActionListener() {
31
32             // display JColorChooser when user clicks button
33             public void actionPerformed( ActionEvent event )
34             {
35                color = JColorChooser.showDialog(
36                   ShowColors2.this, "Choose a color", color );
37
38                // set default color, if no color is returned
39                if ( color == null )
40                   color = Color.lightGray;
41
42                // change content pane's background color
43                container.setBackground( color );
44             }
45
46          } // end anonymous inner class
47
48       ); // end call to addActionListener
49
50       container.add( changeColorButton );
51
52       setSize( 400, 130 );
53       setVisible( true );
54    }
55
56    // execute application
57    public static void main( String args[] )
58    {
```

图11-6 演示JColorChooser对话

```
59          ShowColors2 application = new ShowColors2();
60
61          application.setDefaultCloseOperation(
62             JFrame.EXIT_ON_CLOSE );
63       }
64
65    }  // end class ShowColors2
```

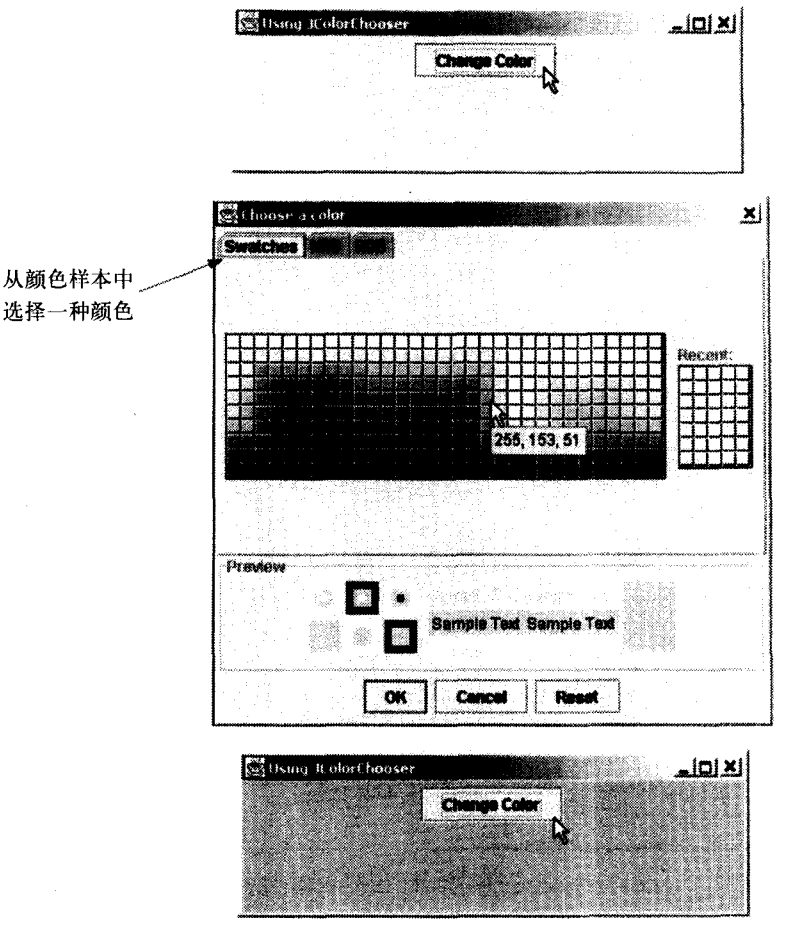

图11-6 （续）

图11-6所示的第2个屏幕说明了默认的JColorChooser对话框允许用户以不同的方式选择颜色。注意对话框的顶部实际上有3个标签：Swatches、HSB和RGB。它们代表选择颜色的3种不同方法。HSB标签页可以基于色彩、饱和度和亮度来选择颜色。RGB标签页让你可以使用滑尺分别选择红、绿、蓝三者的不同值来获得所需要的颜色。图11-7显示了HSB标签页和RGB标签页。

11.4 字体控制

本节介绍用于字体控制的方法和常量。大多数字体控制方法和字体控制常量属于Font类。图11-8总结了Font类和Graphics类的一些方法。

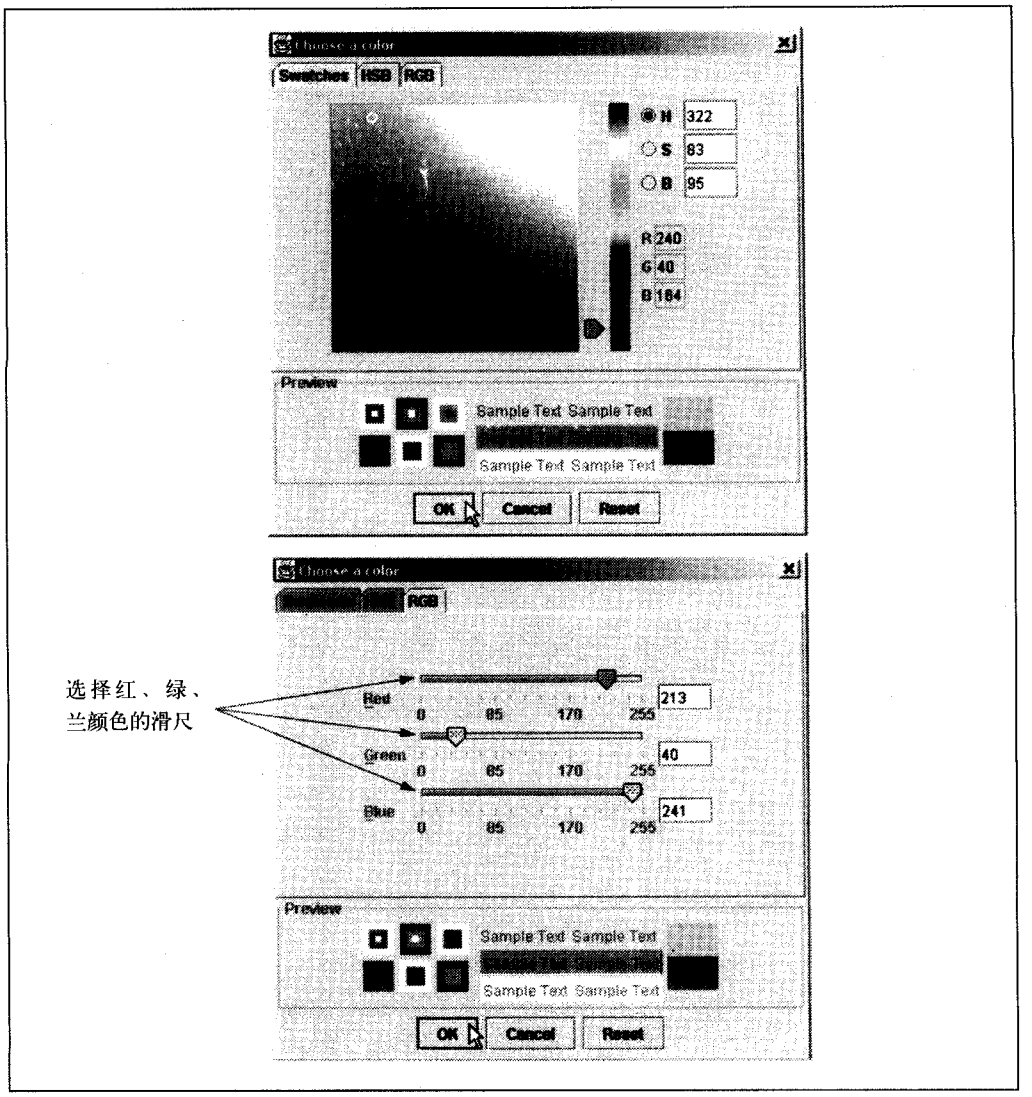

图11-7 JColorChooser对话框的HSB和RGB标签页

类Font的构造函数有3个参数：字体名称、字体风格和字体大小。字体名称可以是运行程序的系统所支持的任何一种字体，如标准Java字体Monospaced、SansSerif和Serif。字体风格可以是Font.PLAIN、Font.ITALIC或Font.BOLD（Font类的static常量）。字体风格还可以组合在一起使用（如：Font.ITALIC+Font.BOLD）。字体大小是以点来衡量的。一个点是1/72英寸。Graphics类的setFont方法用于设置当前字体——也就是将要显示文本的字体——setFont方法的参数为一个Font对象。

可移植性提示11.2 各个系统的字体类型的数目变化很大。J2SDK保证能够提供的字体包括Monospaced、SansSerif、Serif、Dialog和DialogInput。

常见编程错误11.2 指定一个系统不存在的字体是一种逻辑错误。Java将以该系统的默认字体来代替。

方法或常量	说　明
public final static int PLAIN　//Font类	一个代表普通字体风格的常量
public final static int BOLD　//Font类	一个代表黑体字体风格的常量
public final static int ITALIC　//Font类	一个代表斜体字体风格的常量
public Font (String name, int style, int size)	用指定的字体、风格和大小创建一个Font对象
public int getStyle()　// Font类	返回表示当前字体风格的整数值
public int getSize()　// Font类	返回表示当前字体大小的整数值
public String getName()　// Font类	用一个字符串返回当前字体的名称
public String getFamily()　// Font类	用一个字符串返回字体的家族名称
public boolean isPlain()　// Font类	测试一种字体是否是普通字体。如果是普通字体，则返回true
public boolean isBold()　// Font类	测试一种字体是否是黑体风格。如果字体是黑体，则返回true
public boolean isItalic()　// Font类	测试一种字体是否是斜体风格。如果字体是斜体，则返回true
public Font getFont()　// Graphics类	返回一个表示当前字体的Font对象引用
public void setFont (Font f)　// Graphics类	将当前字体设置成由Font对象引用f所指定的字体、风格和大小

图11-8 Font常量、Font方法和Graphics类与字体相关的方法

图11-9的程序显示了4种不同大小、不同字体的文本。在程序的第30行、第35行、第40行和第47行用Font构造函数初始化Font对象（每次通过调用Graphics的setFont方法来改变当前字体）。每次以字符串形式传递字体名称(Serif、Monospaced或SansSerif)，字体风格（Font.PLAIN、Font.ITALIC或Font.BOLD）和字体大小来调用Font构造函数。一旦调用了Graphics的setFont方法，在字体被改变前，所有在调用后显示的文本都将以新字体出现。注意，因为第45行把字体改成了红色，所以下面的字符串就用红色显示。

软件工程评述11.3 *要改变字体，就必须创建一个Font对象，因为Font类中没有改变当前字体特征的方法。*

人们通常需要获取有关当前字体的信息，例如字体名称、字体风格和字体大小。图11-8总结了用来获取字体信息的Font类方法。getStyle方法返回的是一个代表当前风格的整数值。返回的这个整数值可以是Font.PLAIN、Font.ITALIC、Font.BOLD或者是Font.PLAIN、Font.ITALIC和Font.BOLD的任意组合。

getSize方法返回以点为单位的字体大小。getName方法返回当前字体名称的字符串；getFamily方法返回当前字体所属字体家族的名称，字体家族的名称由系统平台具体规定。

可移植性提示11.3 Java使用标准化的字体名称，并且为了可移植性将这些名称映射到系统指定的字体名称上，这一过程对程序员而言是透明的。

```
1  // Fig. 11.9: Fonts.java
2  // Using fonts
3
4  // Java core packages
5  import java.awt.*;
6  import java.awt.event.*;
7
8  // Java extension packages
9  import javax.swing.*;
10
11 public class Fonts extends JFrame {
12
13    // set window's title bar and dimensions
14    public Fonts()
15    {
16       super( "Using fonts" );
17
18       setSize( 420, 125 );
19       setVisible( true );
20    }
21
22    // display Strings in different fonts and colors
23    public void paint( Graphics g )
24    {
25       // call superclass's paint method
26       super.paint( g );
27
28       // set current font to Serif (Times), bold, 12pt
29       // and draw a string
30       g.setFont( new Font( "Serif", Font.BOLD, 12 ) );
31       g.drawString( "Serif 12 point bold.", 20, 50 );
32
33       // set current font to Monospaced (Courier),
34       // italic, 24pt and draw a string
35       g.setFont( new Font( "Monospaced", Font.ITALIC, 24 ) );
36       g.drawString( "Monospaced 24 point italic.", 20, 70 );
37
38       // set current font to SansSerif (Helvetica),
39       // plain, 14pt and draw a string
40       g.setFont( new Font( "SansSerif", Font.PLAIN, 14 ) );
41       g.drawString( "SansSerif 14 point plain.", 20, 90 );
42
43       // set current font to Serif (times), bold/italic,
44       // 18pt and draw a string
45       g.setColor( Color.red );
46       g.setFont(
47          new Font( "Serif", Font.BOLD + Font.ITALIC, 18 ) );
48       g.drawString( g.getFont().getName() + " " +
49          g.getFont().getSize() +
50          " point bold italic.", 20, 110 );
51    }
52
53    // execute application
```

图11-9 使用Graphics方法setFont来改变Font