用 Java 设计文本编辑器 MiniEditor

内容提要：在本文构造实现了一个文本编辑器 MiniEditor，主要内容包括：MiniEditor 的功能需求分析；MiniEditor 的基本设计思路和类的划分；MiniEditor 的具体实现。

关键字：Java、文本编辑器 MiniEditor

引言：设计实现一个 Java 应用程序的过程如下：

（1）功能需求分析。

（2）设计和类划分。

（3）代码编写实现。

本文就按照这个步骤来实现文本编辑器 MiniEditor 的制作。

正文：

1 MiniEditor 功能需求分析

作为文本编辑器，至少应该具备以下几种功能：

（1）所见即所得的文本输入。

（2）方便的选中文本、复制文本、删除文本、插入文本的功能。

（3）方便的查找指定文字、替换文字的功能。

（4）打印功能。

（5）简单的排版功能，如设置字体、字号等。

另外，还要求 MiniEditor 实现一个称为“经典视图 Classic View”的功能，用以使用经典仿 Terminal 形式的界面。

2 MiniEditor 基本设计思路和类划分

基于第 1 节中提出对于 MiniEditor 功能需求的分析，对这个应用程序设计划分类如下：

（1）MiniEditor：这个类作为主类，实现主要功能，并实现图形用户界面。

（2）MenuColor：这个类用来实现文字颜色的编辑功能。

（3）MenuFont：这个类用来实现文字字体、字号的编辑功能。

（4）PrintableTextArea：这个类将实现 TextArea 的功能，并使得文字可以打印输出。

3 MiniEditor 的具体实现

3.1 MiniEditor 类的设计

MiniEditor 类实现整体功能，包括窗体的初始化、各种用户事件监听和响应（编辑、保存、打开等等）。

1. 父类和主要接口

设计 MiniEditor 整个窗体特性继承自 JFrame 类。

为了对用户命令做出响应（如保存文件），MiniEditor 类必须能够监听到用户的命令，因此设计 MiniEditor 类实现 ActionListener 接口。

为了对用户的键盘操作（即编辑输入事件）做出响应，MiniEditor 类必须能够监听到键盘敲击事件，因此设计 MiniEditor 类实现 KeyListener 接口。

MiniEditor 还将实现一个状态显示栏，用于显示当前编辑光标位置，为此设计 MiniEditor 类实现 CaretListener 接口。

此外为了提供一个可以撤消/重复的操作，为 MiniEditor 添加一个 UndoHandler，UndoHandler 类本身实现 UndoListener 接口。

2. 主要方法

下面以表格的形式列出 MiniEditor 类至少应该具有的方法和各自的功能描述（如表 1 所示）。

表 1　MiniEditor 类的主要方法

方法	功能描述
static void main(String args[])	MiniEditor 应用程序的入口，选取特定的 Look And Feel 并初始化窗体
void actionPerformed(ActionEvent e)	重载 ActionListener 接口中的方法，用于对用户命令进行响应，用户命令包括“保存”、“打开”、“关闭”、“打印”
void keyTyped(KeyEvent e)	重载 KeyListener 接口中的方法，用于对用户键盘按下操作进行响应，写入相应的字符到编辑器
void caretUpdate(CaretEvent e)	重载 CaretListener 接口中的方法，用于获取当前光标位置，为 showStatus()方法提供数据
void showStatus()	实时显示当前光标位置

3. 基本效果

图 1 为 MiniEditor 的基本效果图。

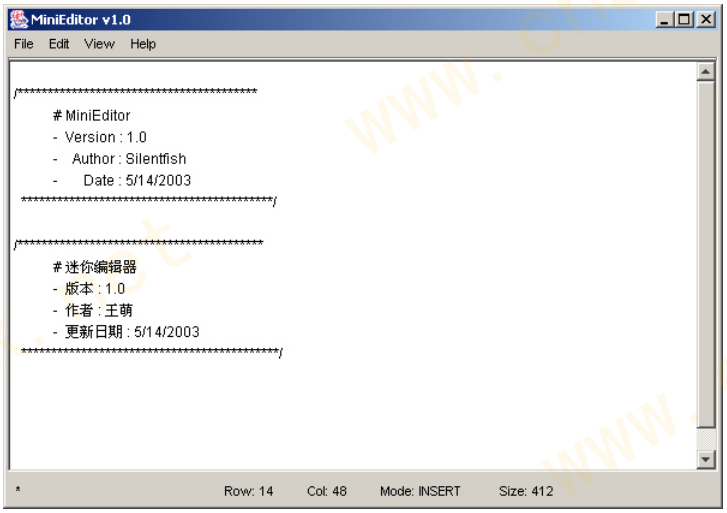

图 1　MiniEditor 的基本效果图

4. 代码分析

MiniEditor.java 代码如下：

```
// MiniEditor.java
/*
 * 文件名: MiniEditor.java
 * 说　明: MiniEditor 主类,实现主要功能
 */
// 导入 AWT 包
import java.awt.*;
import java.awt.event.*;
// 导入 IO 包
import java.io.*;
// 导入 SWING 包
import javax.swing.*;
import javax.swing.event.*;
import javax.swing.text.*;
import javax.swing.undo.*;

// 主类 MiniEditor

public class MiniEditor extends JFrame
```

```
                    implements ActionListener,
                               CaretListener,
                               DocumentListener,
                               KeyListener {

// 菜单项目声明

    JMenuItem menuFileNew, menuFileOpen, menuFileClose,
              menuFileSave, menuFileSaveAs,
              menuFilePrint, menuFilePrintSetup,
              menuFileExit,
              menuEditUndo, menuEditRedo,
              menuEditCut, menuEditCopy, menuEditPaste,
              menuEditDeleteSelection, menuEditDeleteLine, menuEditDeleteWord,
              menuEditFind, menuEditReplace, menuEditGoTo, menuEditSelectAll,
              menuViewFont, menuViewColor,
              menuViewDoubleSpace,
              menuHelpAbout;

    // 选项菜单声明
    JCheckBoxMenuItem menuViewClassic, menuViewStatus, menuViewWordWrap;
    // 可打印 TextArea 声明
    PrintableTextArea ta;
    TextField tfs, tfro, tfrn;
    // 状态栏标签声明
    Label fileStatus, statusRow, statusCol, statusMode, statusSize;
    // 弹出对话框按钮声明
    Button bs, brf, brr, brra;
    // 子 Frame 和 JDialog 实例声明
    JFrame fr;
    JDialog dl;
    // 其他组件声明
    String fns;

    Color defaultForeground, defaultBackground, defaultCaretColor;
    Font defaultFont = new Font（"Courier", Font.PLAIN,12）;
    Checkbox dsLoop, dsMatchCase, drMatchCase;
    JLabel dsMessage2;

    // UndoManager 声明
    protected UndoManager undo = new UndoManager（）;
    protected UndoableEditListener undoHandler = new UndoHandler（）;

    // 状态标记
    static int foundCount = 0;
    int FindStartPos = 0;
    boolean findingLoop = true;
    boolean beginTextListener, isNewFile;
    boolean INSERTMODE = true;
    boolean BACKSPACE = false;

    // 入口方法
    public static void main（String[] args）{
      try {
        // 选取不同 LookAndFeel
        //UIManager.setLookAndFeel（"com.sun.java.swing.plaf.motif.MotifLookAndFeel"）;
        //UIManager.setLookAndFeel（"javax.swing.plaf.metal.MetalLookAndFeel"）;
```

```
        UIManager.setLookAndFeel（"com.sun.java.swing.plaf.windows.WindowsLookAndFeel"）;
    } catch（Exception e） {
    }
    // 设定 Locale
    //Locale.setDefault（Locale.CHINA）;
    //Locale.setDefault（new Locale（"zh", "CN"））;
    MiniEditor me = new MiniEditor（）;
    me.windowLayout（）;
}

// 窗体布局方法
public void windowLayout（）{
    // 对窗体进行布局操作
    // 添加组件到窗体并进行合理布局
    // 初始化窗体
    JFrame fr = new JFrame（"MiniEditor v1.0"）;
    fr.getContentPane（）.setLayout（new BorderLayout（5, 5））;
    JPanel p = new JPanel（）;
    // 初始化状态栏
    fileStatus = new Label（"File Status:                                                    "）;
    statusRow = new Label（"Row:          "）;
    statusCol = new Label（"Col:          "）;
    statusMode = new Label（"Mode:                    "）;
    statusSize = new Label（"Size:                    "）;
    // 初始化文本编辑框
    ta = new PrintableTextArea（20, 50）;
    ta.setCursor（new Cursor（Cursor.HAND_CURSOR））;
    ta.setFont（defaultFont）;
    defaultForeground = ta.getForeground（）;
    defaultBackground = ta.getBackground（）;
    defaultCaretColor = ta.getCaretColor（）;
    ta.setWrapStyleWord（true）;
    ta.getDocument（）.addDocumentListener（this）;
    ta.addCaretListener（this）;
    ta.addKeyListener（this）;
    ta.getDocument（）.addUndoableEditListener（undoHandler）;
    JScrollPane scroller = new JScrollPane（）;
    JViewport port = scroller.getViewport（）;
    port.add（ta）;
    fr.getContentPane（）.add（"Center",scroller）;
    fr.getContentPane（）.add（"South",p）;
    p.setLayout（new FlowLayout（FlowLayout.LEFT, 5, 0））;
    p.add（fileStatus）;
    p.add（statusRow）;
    p.add（statusCol）;
    p.add（statusMode）;
    p.add（statusSize）;
    // 初始化菜单栏
    JMenuBar mb = new JMenuBar（）;
    // 初始化 File 菜单
    JMenu menuFile = new JMenu（"File", true）;
    menuFile.setMnemonic（'F'）;
    // 初始化 New 菜单项
```

```
        menuFileNew = new JMenuItem（"New", 'N'）;
        menuFileNew.setAccelerator （ KeyStroke.getKeyStroke （ KeyEvent.VK_N,
InputEvent.CTRL_MASK））;
        menuFileNew.addActionListener（this）;
        // 初始化 Open 菜单项
        menuFileOpen = new JMenuItem（"Open...", 'O'）;
        menuFileOpen.setAccelerator （ KeyStroke.getKeyStroke （ KeyEvent.VK_O,
InputEvent.CTRL_MASK））;
        menuFileOpen.addActionListener（this）;
        // 初始化 Close 菜单项
        menuFileClose = new JMenuItem（"Close", 'C'）;
        menuFileClose.setAccelerator （ KeyStroke.getKeyStroke （ KeyEvent.VK_F4,
InputEvent.CTRL_MASK））;
        menuFileClose.addActionListener（this）;
        // 初始化 Save 菜单项
        menuFileSave = new JMenuItem（"Save", 'S'）;
        menuFileSave.setAccelerator （ KeyStroke.getKeyStroke （ KeyEvent.VK_S,
InputEvent.CTRL_MASK））;
        menuFileSave.addActionListener（this）;
        // 初始化 Save as 菜单项
        menuFileSaveAs = new JMenuItem（"Save As...", 'A'）;
        menuFileSaveAs.setAccelerator（KeyStroke.getKeyStroke（KeyEvent.VK_F12, 0））;
        menuFileSaveAs.addActionListener（this）;
        // 初始化 Print 菜单项
        menuFilePrint = new JMenuItem（"Print", 'P'）;
        menuFilePrint.setAccelerator （ KeyStroke.getKeyStroke （ KeyEvent.VK_P,
InputEvent.CTRL_MASK））;
        menuFilePrint.addActionListener（this）;
        menuFilePrint.setEnabled（false）;
        // 初始化 Print Setup 菜单项
        menuFilePrintSetup = new JMenuItem（"Print Setup..."）;
        menuFilePrintSetup.addActionListener（this）;
        menuFilePrintSetup.setMnemonic（'u'）;
        menuFilePrintSetup.setEnabled（false）;
        // 初始化 Exit 菜单项
        menuFileExit = new JMenuItem（"Exit", 'x'）;
        menuFileExit.setAccelerator （ KeyStroke.getKeyStroke （ KeyEvent.VK_X,
InputEvent.CTRL_MASK））;
        menuFileExit.addActionListener（this）;
        // 初始化 Edit 菜单项
        JMenu menuEdit = new JMenu（"Edit", true）;
        // 初始化 Undo 菜单项
        menuEditUndo = new JMenuItem（"Undo"）;
        menuEditUndo.setAccelerator （ KeyStroke.getKeyStroke （ KeyEvent.VK_Z,
InputEvent.CTRL_MASK））;
        menuEditUndo.addActionListener（this）;
        menuEditUndo.setEnabled（false）;
        // 初始化 Redo 菜单项
        menuEditRedo = new JMenuItem（"Redo"）;
        menuEditRedo.addActionListener（this）;
        menuEditRedo.setEnabled（false）;
        // 初始化 Cut 菜单项
        menuEditCut = new JMenuItem（"Cut", 't'）;
```

```
        menuEditCut.setAccelerator（KeyStroke.getKeyStroke（KeyEvent.VK_X,
InputEvent.CTRL_MASK））；
        menuEditCut.addActionListener（this）；
        // 初始化 Copy 菜单项
        menuEditCopy = new JMenuItem（"Copy", 'C'）；
        menuEditCopy.setAccelerator（KeyStroke.getKeyStroke（KeyEvent.VK_C,
InputEvent.CTRL_MASK））；
        menuEditCopy.addActionListener（this）；
        // 初始化 Paste 菜单项
        menuEditPaste = new JMenuItem（"Paste", 'P'）；
        menuEditPaste.setAccelerator（KeyStroke.getKeyStroke（KeyEvent.VK_V,
InputEvent.CTRL_MASK））；
        menuEditPaste.addActionListener（this）；
        // 初始化 Delete 菜单项
        JMenu menuEditDelete = new JMenu（"Delete"）；
        menuEditDelete.setMnemonic（'D'）；
        // 初始化 Selection 菜单项
        menuEditDeleteSelection = new JMenuItem（"Selection", 'S'）；
        menuEditDeleteSelection.setAccelerator（KeyStroke.getKeyStroke（KeyEvent.VK_DELETE,
0））；
        menuEditDeleteSelection.addActionListener（this）；
        // 初始化 Line 菜单项
        menuEditDeleteLine = new JMenuItem（"Line", 'L'）；
        menuEditDeleteLine.setAccelerator（KeyStroke.getKeyStroke（KeyEvent.VK_Y,
InputEvent.CTRL_MASK））；
        menuEditDeleteLine.addActionListener（this）；
        // 初始化 Word 菜单项
        menuEditDeleteWord = new JMenuItem（"Word", 'W'）；
        menuEditDeleteWord.setAccelerator（KeyStroke.getKeyStroke（KeyEvent.VK_D,
InputEvent.CTRL_MASK））；
        menuEditDeleteWord.addActionListener（this）；
        // 初始化 Find 菜单项
        menuEditFind = new JMenuItem（"Find...", 'F'）；
        menuEditFind.setAccelerator（KeyStroke.getKeyStroke（KeyEvent.VK_F,
InputEvent.CTRL_MASK））；
        menuEditFind.addActionListener（this）；
        // 初始化 Replace 菜单项
        menuEditReplace = new JMenuItem（"Replace...", 'R'）；
        menuEditReplace.setAccelerator（KeyStroke.getKeyStroke（KeyEvent.VK_H,
InputEvent.CTRL_MASK））；
        menuEditReplace.addActionListener（this）；
        menuEditReplace.setEnabled（false）；
        // 初始化 Go To 菜单项
        menuEditGoTo = new JMenuItem（"Go To...", 'G'）；
        menuEditGoTo.setAccelerator（KeyStroke.getKeyStroke（KeyEvent.VK_G,
InputEvent.CTRL_MASK））；
        menuEditGoTo.addActionListener（this）；
        // 初始化 Select All 菜单项
        menuEditSelectAll = new JMenuItem（"Select All", 'A'）；
        menuEditSelectAll.setAccelerator（KeyStroke.getKeyStroke（KeyEvent.VK_A,
InputEvent.CTRL_MASK））；
        menuEditSelectAll.addActionListener（this）；
        // 初始化 View 菜单项
```

```
        JMenu menuView = new JMenu（"View", true）；
        // 初始化 Font 菜单项
        menuViewFont = new JMenuItem（"Font...", 'F'）；
        menuViewFont.setAccelerator （ KeyStroke.getKeyStroke （ KeyEvent.VK_F,
InputEvent.CTRL_MASK + InputEvent.ALT_MASK））；
        menuViewFont.addActionListener（this）；
        // 初始化 Color
        menuViewColor = new JMenuItem（"Color...", 'C'）；
        menuViewColor.setAccelerator （ KeyStroke.getKeyStroke （ KeyEvent.VK_C,
InputEvent.CTRL_MASK + InputEvent.ALT_MASK））；
        menuViewColor.addActionListener（this）；
        // 初始化 Classic 菜单项
        menuViewClassic = new JCheckBoxMenuItem（"Classic"）；
        menuViewClassic.setMnemonic（'l'）；
        menuViewClassic.setAccelerator （ KeyStroke.getKeyStroke （ KeyEvent.VK_L,
InputEvent.CTRL_MASK + InputEvent.ALT_MASK））；
        menuViewClassic.setState（false）；
        menuViewClassic.addActionListener（this）；
        // 初始化 Status 菜单项
        menuViewStatus = new JCheckBoxMenuItem（"Status"）；
        menuViewStatus.setMnemonic（'S'）；
        menuViewStatus.setAccelerator （ KeyStroke.getKeyStroke （ KeyEvent.VK_S,
InputEvent.CTRL_MASK + InputEvent.ALT_MASK））；
        menuViewStatus.setState（true）；
        menuViewStatus.addActionListener（this）；
        // 初始化 Word　Wrap 菜单项
        menuViewWordWrap = new JCheckBoxMenuItem（"Word　Wrap"）；
        menuViewWordWrap.setMnemonic（'W'）；
        menuViewWordWrap.setAccelerator （ KeyStroke.getKeyStroke （ KeyEvent.VK_W,
InputEvent.CTRL_MASK + InputEvent.ALT_MASK））；
        menuViewWordWrap.setState（false）；
        menuViewWordWrap.addActionListener（this）；
        // 初始化 Double　Space 菜单项
        menuViewDoubleSpace = new JMenuItem（"Double　Space", 'D'）；
        menuViewDoubleSpace.setAccelerator （ KeyStroke.getKeyStroke （ KeyEvent.VK_D,
InputEvent.CTRL_MASK + InputEvent.ALT_MASK））；
        menuViewDoubleSpace.addActionListener（this）；
        // 初始化 Help 菜单项
        JMenu menuHelp = new JMenu（"Help", true）；
        // 初始化 About 菜单项
        menuHelpAbout = new JMenuItem（"About", 'A'）；
        menuHelpAbout.setAccelerator（KeyStroke.getKeyStroke（KeyEvent.VK_F1, 0））；
        menuHelpAbout.addActionListener（this）；

        // 添加菜单
        fr.setJMenuBar（mb）；
        mb.add（menuFile）；
        menuFile.add（menuFileNew）；
        menuFile.add（menuFileOpen）；
        menuFile.add（menuFileClose）；
        menuFile.addSeparator（）；
        menuFile.add（menuFileSave）；
        menuFile.add（menuFileSaveAs）；
```

```
    menuFile.addSeparator（）；
    menuFile.add（menuFilePrint）；
    menuFile.add（menuFilePrintSetup）；
    menuFile.addSeparator（）；
    menuFile.add（menuFileExit）；
    mb.add（menuEdit）；
    menuEdit.add（menuEditUndo）；
    menuEdit.add（menuEditRedo）；
    menuEdit.addSeparator（）；
    menuEdit.add（menuEditCut）；
    menuEdit.add（menuEditCopy）；
    menuEdit.add（menuEditPaste）；
    menuEdit.add（menuEditDelete）；
    menuEditDelete.add（menuEditDeleteSelection）；
    menuEditDelete.add（menuEditDeleteLine）；
    menuEditDelete.add（menuEditDeleteWord）；
    menuEdit.addSeparator（）；
    menuEdit.add（menuEditFind）；
    menuEdit.add（menuEditReplace）；
    menuEdit.add（menuEditGoTo）；
    menuEdit.addSeparator（）；
    menuEdit.add（menuEditSelectAll）；
    mb.add（menuView）；
    menuView.add（menuViewFont）；
    menuView.add（menuViewColor）；
    menuView.addSeparator（）；
    menuView.add（menuViewClassic）；
    menuView.add（menuViewStatus）；
    menuView.add（menuViewWordWrap）；
    menuView.addSeparator（）；
    menuView.add（menuViewDoubleSpace）；
    mb.add（menuHelp）；
    menuHelp.add（menuHelpAbout）；
    fr.addWindowListener（new WindowAdapter（） {
                        public void windowClosing（WindowEvent e） {
                            System.exit（0）；
                        }
                    }）；
    fr.setSize（600,420）；
    fr.setVisible（true）；
}
// 事件监听方法
public void actionPerformed（ActionEvent ae） {
    // 文件菜单事件响应
    if（ae.getSource（） == menuFileNew） {
        // 新建文件
        ta.replaceRange（"", 0, ta.getText（）.length（））；
        fns = null;
        fileStatus.setText（"New File"）；
        undo.discardAllEdits（）；
        menuEditUndo.setEnabled（false）；
    }
    else if（ae.getSource（） == menuFileOpen） {
```

```
        // 打开文件
        String s = null;
        StringBuffer strPool = new StringBuffer（）;
        Frame openFileFrame = new Frame（"Open file"）;
        FileDialog fileDialog = new FileDialog（openFileFrame）;
        fileDialog.setMode（FileDialog.LOAD）;
        fileDialog.setFile（"*.txt;*.java"）;
        fileDialog.show（）;
        String file = fileDialog.getFile（）;
        String directory = fileDialog.getDirectory（）;
// 读取文件
         if（file != null） {
          fns = directory + file;
          ta.replaceRange（"", 0, ta.getText（）.length（））;
          BufferedReader br;
            try{
              br = new BufferedReader（new FileReader（fns））;
              s = br.readLine（）;
              while（s != null） {
                strPool.append（s + '\15' + "\12"）;
                s = br.readLine（）;
              }
              br.close（）;
              ta.setText（strPool.toString（））;
            } catch（IOException e） {
            }
          // 显示状态
          fileStatus.setText（"File opened."）;
          isNewFile = true;
          undo.discardAllEdits（）;
          menuEditUndo.setEnabled（false）;
        }
      }
      else if（ae.getSource（） == menuFileClose） {
        // 关闭文件
        ta.replaceRange（"", 0, ta.getText（）.length（））;
        fileStatus.setText（"File closed without save"）;
        undo.discardAllEdits（）;
        menuEditUndo.setEnabled（false）;
        fns = null;
      }
      else if（ae.getSource（） == menuFilePrint） {
        // 打印文件
        JOptionPane.showMessageDialog（null, "TODO... Only the FIRST page now.", "Print", 1）;
        ta.printIt（ta.getText（）, ta.getFont（））;
     }
      else if（ae.getSource（） == menuFileSave） {
        // 保存文件
        OutputStreamWriter osw;
        if（fns != null） {
          try{
            osw = new OutputStreamWriter（
                    new BufferedOutputStream（
                      new FileOutputStream（fns））;
```

```
            osw.write（ta.getText（）, 0, ta.getText（）.length（））；
            osw.close（）；
            fileStatus.setText（"file saved"）；
          } catch（IOException e） {
          }
          if（fileStatus.getText（）.endsWith（"*"）） {
            fileStatus.setText（fileStatus.getText（）.substring（0,
                                    fileStatus.getText（）.length（） - 1））；
          }
        }
        else {
          Frame saveFileFrame = new Frame（"Save file"）；
          FileDialog fileDialog = new FileDialog（saveFileFrame）；
          fileDialog.setMode（FileDialog.SAVE）；
          fileDialog.setFile（"*.txt;*.java"）；
          fileDialog.show（）；
          String file = fileDialog.getFile（）；
          String directory = fileDialog.getDirectory（）；
          if（file != null） {
            fns = directory + file;
            try{
              osw = new OutputStreamWriter（
                        new BufferedOutputStream（
                          new FileOutputStream（fns））；
              osw.write（ta.getText（）, 0, ta.getText（）.length（））；
              osw.close（）；
              fileStatus.setText（"File saved"）；
            } catch（IOException e） {
            }
          }
        }
      }
      else if（ae.getSource（） == menuFileSaveAs） {
        // 另存文件
        OutputStreamWriter osw;
        Frame saveFileFrame = new Frame（"Save file"）；
        FileDialog fileDialog = new FileDialog（saveFileFrame）；
        fileDialog.setMode（FileDialog.SAVE）；
        fileDialog.setFile（"*.txt;*.java"）；
        fileDialog.show（）；
        String file = fileDialog.getFile（）；
        String directory = fileDialog.getDirectory（）；
        if（file != null） {
          fns = directory + file;
          try{
            osw = new OutputStreamWriter（
                      new BufferedOutputStream（
                        new FileOutputStream（fns））；
            osw.write（ta.getText（）, 0, ta.getText（）.length（））；
            osw.close（）；
            fileStatus.setText（"File saved"）；
          } catch（IOException e） {
          }
        }
```

```
}
else if（ae.getSource（）  == menuFileExit）  {
  // 退出 MiniEditor
  System.exit（0）;
}
// 编辑菜单事件响应
else if（ae.getSource（）  == menuEditUndo）  {
  // Undo 操作
  if（undo.canUndo（））  {
    try {
      undo.undo（）;
    } catch（CannotUndoException ex）  {
        System.out.println（"Unable to undo: " + ex）;
        ex.printStackTrace（）;
    }
    if（!undo.canUndo（））
      menuEditUndo.setEnabled（false）;
  }
  else
    menuEditUndo.setEnabled（false）;
}
else if（ae.getSource（）  == menuEditRedo）  {
  // Redo 操作，未实现
    }
else if（ae.getSource（）  == menuEditCut）  {
  //Cut 操作
  ta.cut（）;
}
else if（ae.getSource（）  == menuEditCopy）  {
  // Copy 操作
  ta.copy（）;
}
else if（ae.getSource（）  == menuEditPaste）  {
  // Paste 操作
  ta.paste（）;
}
else if（ae.getSource（）  == menuEditDeleteSelection）  {
  // DeleteSelection 操作
  ta.replaceRange（"", ta.getSelectionStart（）, ta.getSelectionEnd（））;
}
else if（ae.getSource（）  == menuEditDeleteLine）  {
  // 删除行操作
  String str = ta.getText（）;
  int pos = ta.getCaretPosition（）;
  int lineStart = 0, lineEnd = 0;
  lineStart = str.substring（0, pos）.lastIndexOf（'\12'）;
  lineEnd = str.indexOf（'\15', pos）;
  lineStart =  （lineStart == -1）  ? 0 :  （lineStart - 1）;
  ta.replaceRange（"", lineStart, lineEnd）;
  lineStart =  （lineStart == 0）  ? 0 :  （lineStart + 2）;
  ta.setCaretPosition（lineStart）;
}
else if（ae.getSource（）  == menuEditDeleteWord）  {
  // 删除 Word 操作
```

```
    String str = ta.getText（）；
    int pos = ta.getCaretPosition（）；
    int wordStart = 0, wordEnd = 0;
    wordEnd = wordLocation（str, pos, true）；
    if（wordEnd == pos） {
      wordStart = pos;
      wordEnd = pos + 1;
      ta.replaceRange（"", wordStart, wordEnd）；
    }
    else {
      wordStart = wordLocation（str, pos, false）；
      if（wordStart == -1）
          ta.replaceRange（"", 0, wordEnd）；
      else
        ta.replaceRange（"", wordStart, wordEnd）；
    }
  }
  else if（ae.getSource（） == menuEditFind） {
    // Find 操作
    // 创建 Find 查找对话框
    JDialog ds = new JDialog（this, "Find", true）；
    ds.getContentPane（）.setLayout（new FlowLayout（））；
    ds.setResizable（false）；
    tfs = new TextField（15）；
    ds.getContentPane（）.add（tfs）；
    bs = new Button（"Find"）；
    bs.addActionListener（this）；
    Button bsc = new Button（"Cancel"）；
    bsc.addActionListener（new ActionListener（） {
                              public void actionPerformed（ActionEvent bOke） {
                                dispose（）；
                                foundCount = 0;
                              }
                            }）；
    ds.getContentPane（）.add（bs）；
    ds.getContentPane（）.add（bsc）；
    JLabel dsMessage1 = new JLabel（"Found counts:  "）；
    dsMessage2 = new JLabel（" 0"）；
    dsLoop = new Checkbox（"Loop"）；
    dsLoop.setState（findingLoop）；
    dsMatchCase = new Checkbox（"Match Case"）；
    ds.getContentPane（）.add（dsLoop）；
    ds.getContentPane（）.add（dsMatchCase）；
    ds.getContentPane（）.add（dsMessage1）；
    ds.getContentPane（）.add（dsMessage2）；
    ds.setLocation（120, 120）；
    ds.addWindowListener（new WindowAdapter（） {
                              public void windowClosing（WindowEvent e） {
                                dispose（）；
                                FindStartPos = 0;
                              }
                            }）；
    //显示 Find 对话框
    ds.setSize（250,110）；
```

```
    ds.setVisible（true）;
}
else if（ae.getSource（）  == bs）  {
    int a = 0, b = 0;
    String str1, str2, str3, str4, strA, strB;
    str1 = ta.getText（）;
    str2 = str1.toLowerCase（）;
    str3 = tfs.getText（）;
    str4 = str3.toLowerCase（）;
    if（dsMatchCase.getState（））  {
        strA = str1;
        strB = str3;
    }
    else {
        strA = str2;
        strB = str4;
    }
    a = strA.indexOf（strB, FindStartPos）;
    if（a > -1）  {
        ta.setCaretPosition（a）;
        b = tfs.getText（）.length（）;
        ta.select（a, a + b）;
        FindStartPos = a + b;
        foundCount++;
        dsMessage2.setText（foundCount+""）;
    }
    else {
        if（dsLoop.getState（））  {
            JOptionPane.showMessageDialog（null, "End of file.", "Find result",1）;
            FindStartPos = 0;
        }
        else {
            JOptionPane.showMessageDialog（null, "End of file.", "Find result",1）;
        }
        foundCount = 0;
    }
}
else if（ae.getSource（）  == menuEditReplace）  {
    // 替换操作
    // 创建 Replace 替换对话框
    JDialog dr = new JDialog（this, "Replace", true）;
    dr.getContentPane（）.setLayout（new GridBagLayout（））;
    GridBagConstraints gbc = new GridBagConstraints（）;
    gbc.gridx = 0;
    gbc.gridy = 0;
    gbc.gridwidth = 2;
    gbc.gridheight =1;
    gbc.fill = gbc.HORIZONTAL; gbc.anchor = gbc.CENTER;
    Panel p1 = new Panel（）;
    dr.getContentPane（）.add（p1）;
    drMatchCase = new Checkbox（"Match Case"）;
    gbc.gridx = 1;
    gbc.gridy = 1;
    gbc.gridwidth = 1;
    gbc.gridheight =1;
    dr.getContentPane（）.add（drMatchCase）;
```

```
gbc.gridx = 2;
gbc.gridy = 0;
gbc.gridwidth = 1;
gbc.gridheight =1;
gbc.fill = gbc.NONE; gbc.anchor = gbc.EAST;
Panel p2 = new Panel（）；
p1.setLayout（new GridLayout（5, 1））；
//设置布局方式
p2.setLayout（new GridLayout（4, 1））；
dr.getContentPane（）.add（p2）；
JLabel drMessage1 = new JLabel（"Replace:  "）；
JLabel drMessage2 = new JLabel（"With:  "）；
drMatchCase = new Checkbox（"Match Case"）；
tfro = new TextField（15）；
tfrn = new TextField（15）；
p1.add（drMessage1）；
p1.add（tfro）；
p1.add（drMessage2）；
p1.add（tfrn）；
p1.add（drMatchCase）；
//加入监听器
brf = new Button（"Find"）；
brf.addActionListener（this）；
brf.addActionListener（new ActionListener（） {
                    public void actionPerformed（ActionEvent brfe） {
                      int a = 0, b = 0;
                      String str1, str2, str3, str4, strA, strB;
                      str1 = ta.getText（）；
                      str2 = str1.toLowerCase（）；
                      str3 = tfro.getText（）；
                      str4 = str3.toLowerCase（）；
                      if（drMatchCase.getState（）） {
                        strA = str1;
                        strB = str3;
                      }
                      else {
                        strA = str2;
                        strB = str4;
                      }
                      a = strA.indexOf（strB, FindStartPos）；
                      if（a > -1） {
                        ta.setCaretPosition（a）；
                        b = tfro.getText（）.length（）；
                        ta.select（a, a + b）；
                        FindStartPos = a + b;
                        foundCount++;
                      }
                      else {
                        JOptionPane.showMessageDialog（null, "End of file.",
"Result",1）；
                        foundCount = 0;
                      }
                    }
                  }）；
brr = new Button（"Replace"）；
```

```
            //加入监听器
            brr.addActionListener（new ActionListener（） {
                                    public void actionPerformed（ActionEvent brre） {
                                      if（tfrn.getText（）.length（） == 0 && ta.getSelectedText
（） != null）
                                        ta.replaceSelection（""）;
                                      if（tfrn.getText（）.length（） > 0 && ta.getSelectedText
（） != null）
                                        ta.replaceSelection（tfrn.getText（））;
                                      int a = 0, b = 0;
                                      String str1, str2, str3, str4, strA, strB;
                                      str1 = ta.getText（）;
                                      str2 = str1.toLowerCase（）;
                                      str3 = tfro.getText（）;
                                      str4 = str3.toLowerCase（）;
                                      if（drMatchCase.getState（）） {
                                        strA = str1;
                                        strB = str3;
                                      }
                                      else {
                                        strA = str2;
                                        strB = str4;
                                      }
                                      a = strA.indexOf（strB, FindStartPos）;
                                      if（a > -1） {
                                        ta.setCaretPosition（a）;
                                        b = tfro.getText（）.length（）;
                                        ta.select（a, a + b）;
                                        FindStartPos = a + b;
                                        foundCount++;
                                      }
                                      else {
                                        JOptionPane.showMessageDialog（null, "End of file.",
"Result",1）;
                                        foundCount = 0;
                                      }
                                    }
                                  }）;
            brra = new Button（"Replace All"）;
            //加入监听器
            brra.addActionListener（new ActionListener（） {
                                    public void actionPerformed（ActionEvent brrae） {
                                      int a = 0;
                                      while（ a > -1） {
                                        int b = 0;
                                        String str1, str2, str3, str4, strA, strB;
                                        str1 = ta.getText（）;
                                        str2 = str1.toLowerCase（）;
                                        str3 = tfro.getText（）;
                                        str4 = str3.toLowerCase（）;
                                        if（drMatchCase.getState（）） {
                                          strA = str1;
                                          strB = str3;
                                        }
                                        else {
```

```
                                        strA = str2;
                                        strB = str4;
                                    }
                                    a = strA.indexOf（strB, FindStartPos）;
                                    if（a > -1） {
                                        ta.setCaretPosition（a）;
                                        b = tfro.getText（）.length（）;
                                        ta.select（a, a + b）;
                                        FindStartPos = a + b;
                                        foundCount++;
                                    }
                                    else {
                                        JOptionPane.showMessageDialog（null, "End of file.",
"Result",1）;
                                        foundCount = 0;
                                    }
                                    if（tfrn.getText（）.length（） == 0 && ta.getSelectedText
（） != null）
                                        ta.replaceSelection（""）;
                                    if（tfrn.getText（）.length（） > 0 && ta.getSelectedText
（） != null）
                                        ta.replaceSelection（tfrn.getText（））;
                                    }
                                }
                            }）;
            Button brc = new Button（"Cancel"）;
            //加入监听器
            brc.addActionListener（new ActionListener（） {
                                public void actionPerformed（ActionEvent brce） {
                                    dispose（）;
                                }
                            }）;
            p2.add（brf）;
            p2.add（brr）;
            p2.add（brra）;
            p2.add（brc）;
            dr.setResizable（false）;
            dr.setLocation（120, 120）;
            dr.addWindowListener（new WindowAdapter（） {
                                public void windowClosing（WindowEvent e） {
                                    dispose（）;
                                    FindStartPos = 0;
                                }
                            }）;
            dr.setSize（220,138）;
            //显示 Replane 查找对话框
            dr.setVisible（true）;

        }
        else if（ae.getSource（） == menuEditGoTo） {
            // GoTo 操作
            JDialog dg = new JDialog（this, "Line Number", true）;
            dg.getContentPane（）.setLayout（new FlowLayout（））;
            final JTextField dgtf = new JTextField（4）;
```

```
            Button dgOk = new Button（"Go To"）;
            //加入监听器
            dgOk.addActionListener（new ActionListener（） {
                                    public void actionPerformed（ActionEvent brce） {
                                      int totalLine = ta.getLineCount（）;
                                      int[] lineNumber = new int[totalLine + 1];
                                      String s = ta.getText（）;
                                      int pos = 0, t = 0;
                                      while（true） {
                                        pos = s.indexOf（'\12', pos）;
                                        if（pos == -1）
                                          break;
                                        lineNumber[t++] = pos++;
                                      }
                                      int gt = 1;
                                      try {
                                        gt = Integer.parseInt（dgtf.getText（））;
                                      } catch（NumberFormatException efe） {
                                        JOptionPane.showMessageDialog（null, "Please enter an
integer.", "Input",1）;
                                      }
                                      if（gt < 2 || gt >= totalLine） {
                                        if（gt < 2）
                                          ta.setCaretPosition（0）;
                                        else
                                          ta.setCaretPosition（s.length（））;
                                      }
                                      else
                                        ta.setCaretPosition（lineNumber[gt-2] + 1）;
                                      dispose（）;
                                    }
                                  }）;
            Button dgCancel = new Button（"Cancel"）;
            //加入监听器
            dgCancel.addActionListener（new ActionListener（） {
                                      public void actionPerformed（ActionEvent brce） {
                                        dispose（）;
                                      }
                                    }）;
            dg.getContentPane（）.add（dgtf）;
            dg.getContentPane（）.add（dgOk）;
            dg.getContentPane（）.add（dgCancel）;
            dg.setResizable（false）;
            dg.setLocation（120, 120）;
            dg.addWindowListener（new WindowAdapter（） {
                                  public void windowClosing（WindowEvent e） {
                                    dispose（）;
                                  }
                                }）;
            dg.setSize（180,60）;
            dg.setVisible（true）;
          }
          else if（ae.getSource（） == menuEditSelectAll） {
            // SelectAll 操作
```

```
    ta.selectAll（）；
}
else if（ae.getSource（）  == menuViewFont）  {
   // Font 操作
   MenuFont mf = new MenuFont（this, true）；
   ta.setFont（mf.myLayout（ta.getFont（））；
}
else if（ae.getSource（）  == menuViewColor）  {
   // Color 操作
   MenuColor mc = new MenuColor（this, true）；
   Color[]   fbgc = new Color[2];
   fbgc = mc.myLayout（ta.getForeground（）, ta.getBackground（））；
   ta.setForeground（fbgc[0]）；
   ta.setBackground（fbgc[1]）；
   ta.setCaretColor（fbgc[0]）；
}
else if（ae.getSource（）  == menuViewClassic）  {
   // 切换经典模式
   if（menuViewClassic.getState（））  {
      ta.setForeground（new Color（0, 255, 0））；
      ta.setBackground（new Color（45, 0, 45））；
      ta.setFont（new Font（"Serif", Font.BOLD, 16））；
      ta.setCaretColor（new Color（0, 255, 0））；
   }
   else {
      ta.setForeground（defaultForeground）；
      ta.setBackground（defaultBackground）；
      ta.setFont（defaultFont）；
      ta.setCaretColor（defaultCaretColor）；
   }
}
else if（ae.getSource（）  == menuViewStatus）  {
   // 显示状态
   if（menuViewStatus.getState（））  {
      showStatus（）；
   }
}
else if（ae.getSource（）  == menuViewWordWrap）  {
   if（menuViewWordWrap.getState（））  {
      ta.setLineWrap（true）；
   }
   else {
      ta.setLineWrap（false）；
   }
}
else if（ae.getSource（）  == menuViewDoubleSpace）  {
   int pos = 0, t = 0;
   String str = ta.getText（）；
   while（true）{
      pos = str.indexOf（'\15', pos）；
      if（pos == -1）  break;
      str = str.substring（0, pos）  + '\15' + '\12' + str.substring（pos）；
      pos = pos + 3;
   }
```

```
        ta.setText（str）；
    }
    else if（ae.getSource（）  == menuHelpAbout）  {
        // 帮助
        // 创建 About 对话框
        dl = new JDialog（this,"About MiniEditor", true）；
        dl.getContentPane（）.setLayout（new GridLayout（3,3））；
        dl.setBackground（new Color（212,208,200））；
        Button bOk = new Button（"OK"）；
        //加入监听器
        bOk.addActionListener（new ActionListener（）  {
                                    public void actionPerformed（ActionEvent bOke）  {
                                        dispose（）；
                                    }
                                }）；
        Label ver = new Label（"Version 1.0 "）；
        Label null1 = new Label（）；
        Label null2 = new Label（）；
        Label null3 = new Label（）；
        Label null4 = new Label（）；
        Label null5 = new Label（）；
        Label null6 = new Label（）；
        Label null7 = new Label（）；
        dl.getContentPane（）.add（null1）；
        dl.getContentPane（）.add（ver）；
        dl.getContentPane（）.add（null2）；
        dl.getContentPane（）.add（null3）；
        dl.getContentPane（）.add（bOk）；
        dl.getContentPane（）.add（null4）；
        dl.getContentPane（）.add（null5）；
        dl.getContentPane（）.add（null6）；
        dl.getContentPane（）.add（null7）；
        bOk.addActionListener（this）；
        dl.addWindowListener（new WindowAdapter（）  {
                                    public void windowClosing（WindowEvent e）  {
                                        dispose（）；
                                    }
                                }）；
        dl.setLocation（120, 120）；
        dl.setResizable（false）；
        dl.setSize（200,80）；
        // 显示 About 对话框
        dl.setVisible（true）；

    }
}//actionPerformed 方法结束
public void removeUpdate（DocumentEvent e）  {
    String s;
    s = fileStatus.getText（）；
    if（!s.endsWith（"*"）  & beginTextListener & !isNewFile）  {
        fileStatus.setText（"*"）；
    }
    menuEditUndo.setEnabled（true）；
```

```
}
public void insertUpdate（DocumentEvent e） {
   String s;
   s = fileStatus.getText（）;
   if（!s.endsWith（"*"） & beginTextListener & !isNewFile） {
      fileStatus.setText（"*"）;
   }
   menuEditUndo.setEnabled（true）;
}
public void changedUpdate（DocumentEvent e） {
   String s;
   s = fileStatus.getText（）;
   if（!s.endsWith（"*"） & beginTextListener & !isNewFile） {
      fileStatus.setText（"*"）;
   }
   menuEditUndo.setEnabled（true）;
}

//caret 监听方法
public void caretUpdate（CaretEvent e） {
   if（menuViewStatus.getState（））
      showStatus（）;
}// caretUpdate 方法结束

// 键盘监听方法
public void keyPressed（KeyEvent e） {
   if（e.getKeyCode（） == '\10'） {
      BACKSPACE = true;
   }
   //if（menuViewStatus.getState（））
   //   showStatus（）;
}
public void keyReleased（KeyEvent e） {
   if（e.getKeyCode（） == 155） { //ESCAPE = 155
      if（INSERTMODE）
         INSERTMODE = false;
      else
         INSERTMODE = true;
   }
   if（menuViewStatus.getState（））
      showStatus（）;
}
public void keyTyped（KeyEvent e） {
   beginTextListener = true;
   isNewFile = false;
   if（!BACKSPACE） {
      if（!INSERTMODE） {
         int pos = ta.getCaretPosition（）;
         char c = ta.getText（）.charAt（pos）;
         if（c == '\12'） {
         }
         else if（c == '\15'） {
         }
         else {
            ta.replaceRange（"", pos, pos + 1）;
```

```
            }
        }
    }
    BACKSPACE = false;
}// keyTyped 方法结束
// 显示当前光标位置以及编辑属性
void showStatus（） {
    int rows, cols, from, current, to, fileSize;
    rows = cols = from = current = 0;
    to = ta.getCaretPosition（）;
    fileSize = 0;
    String str = ta.getText（）;
    cols = to - str.substring（0, to）.lastIndexOf（10）;
    fileSize = str.length（）;
    String mode;
    if（INSERTMODE） {
        mode = "INSERT";
    }
    else {
        mode = "OVERLAY";
    }
    try {
        rows = ta.getLineOfOffset（to） + 1;
    } catch（BadLocationException ble） {
    }
    statusRow.setText（"Row: " + rows）;
    statusCol.setText（"Col: " + cols）;
    statusMode.setText（"Mode: " + mode）;
    statusSize.setText（"Size: " + fileSize）;
    //fileStatus.setText（"file status:                                    "）;
} //showStatus 方法结束

// 单词定位方法
int wordLocation（String str, int pos, boolean isToRight） {
    char c;
    if（isToRight） {
        c = str.charAt（pos）;
        while（true） {
            if（c < 48） break;
            else if（c > 57 & c < 65） break;
            else if（c > 90 & c < 97） break;
            else if（c > 122） break;
            pos++;
            c = str.charAt（pos）;
        }
        return pos--;
    }
    else {
        pos--;
        c = str.charAt（pos）;
        while（true） {
            if（c < 48） break;
            else if（c > 57 & c < 65） break;
            else if（c > 90 & c < 97） break;
            else if（c > 122） break;
```

```
            pos--;
            c = str.charAt（pos）;
          }
          return pos++;
        }
      }
      //UndoHandler 的声明
    class UndoHandler implements UndoableEditListener {
      public void undoableEditHappened（UndoableEditEvent uee） {
        undo.addEdit（uee.getEdit（））;
      }
    }

    }//MiniEditor 类声明结束
```

3.2 MenuColor 类的设计

MenuColor 用于编辑文本框中的颜色显示属性。

1. 父类和主要接口

MenuColor 被设计为对话框形式，它的窗体特性继承自 JDialog。

为了综合对颜色进行设置，MenuColor 当中使用了多种 GUI 组件，包括 JTextField、JScrollBar 等，为此，需要实现一些相应监听接口来实现对各种事件的响应，这些接口包括 ItemListener、TextListener、ActionListener 和 AdjustmentListener。

2. 主要方法

MenuColor 类的主要方法及其功能描述如表 2 所示。

表 2 MenuColor 类的主要方法及功能描述

方法	功能描述
void actionPerformed(ActionEvent e)	用于监听用户命令并响应
void adjustmentValueChanged(AdjustmentEvent ade)	色彩数值调整监听方法
void itemStateChanged(ItemEvent ie)	Item 事件监听方法
void textValueChanged(TextEvent te)	文字改变监听方法

3. 基本效果

如图 2 所示为颜色编辑效果。

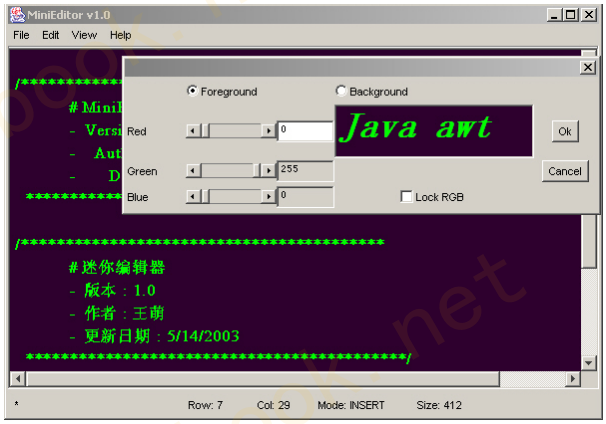

图 2 颜色编辑效果

4. 代码分析

MenuColor.java

```
// MenuColor.java
/*
 * 文件名:MenuColor.java
```

```
 * 说　明:实现文字颜色的编辑功能
 */
// 导入相关包
import java.awt.*;
import java.awt.event.*;

import javax.swing.*;
// 主类 MenuColor
public class MenuColor extends JDialog
                        implements ItemListener,
                                   ActionListener,
                                   TextListener,
                                   AdjustmentListener {
  // AWT 组件声明
  CheckboxGroup gp;
  Checkbox fore, back;
  Scrollbar scrollbarRed, scrollbarGreen, scrollbarBlue;
  TextField textFieldRed, textFieldGreen, textFieldBlue;
  Button colorButtonOk, colorButtonCancel;
  Checkbox colorCheckbox;
  TextField colorTextField;
  // 改变标记
  boolean changed = true;
  // 同步标记
  boolean synchronism = false;
  // 颜色数组
  Color[] fbgc = new Color[2];
  Color[] fbgcOld = new Color[2];
  // 前景色标记
  boolean isFore = true;

  // 构造方法
  MenuColor（Frame frame, boolean modal） {
    super（frame, modal）;
  }

  // myLayout 方法，窗体布局
  public Color[] myLayout（Color fgc, Color bgc） {
    fbgc[0] = fgc;
    fbgc[1] = bgc;
    fbgcOld[0] = fgc;
    fbgcOld[1] = bgc;
    this.getContentPane（）.setLayout（new GridBagLayout（））;
    GridBagConstraints gbc = new GridBagConstraints（）;
    gbc.gridwidth = 1;
    gbc.gridheight =1;
    gbc.weightx = 1;
    gbc.weighty = 1;
    gbc.fill = gbc.HORIZONTAL;
    gbc.anchor = gbc.CENTER;
    // 初始化色调选择组件
    gp = new CheckboxGroup（）;
    fore = new Checkbox（"Foreground", true, gp）;
    back = new Checkbox（"Background", false, gp）;
    gbc.gridx = 1;
    gbc.gridy = 0;
    getContentPane（）.add（fore, gbc）;
```

```
gbc.gridx = 3;
gbc.gridy = 0;
getContentPane（）.add（back, gbc）;
fore.addItemListener（this）;
back.addItemListener（this）;
// 初始化 Red 标签
Label labelRed = new Label（"Red"）;
gbc.gridx = 0;
gbc.gridy = 1;
getContentPane（）.add（labelRed, gbc）;
scrollbarRed = new Scrollbar（Scrollbar.HORIZONTAL, 0, 1, 0, 256）;
scrollbarRed.setValue（fgc.getRed（））;
gbc.gridx = 1;
gbc.gridy = 1;
getContentPane（）.add（scrollbarRed, gbc）;
scrollbarRed.addAdjustmentListener（this）;
textFieldRed = new TextField（3）;
textFieldRed.setText（scrollbarRed.getValue（） + ""）;
textFieldRed.addTextListener（this）;
textFieldRed.setEditable（false）;
gbc.gridx = 2;
gbc.gridy = 1;
getContentPane（）.add（textFieldRed, gbc）;
// 初始化 Green 标签
Label labelGreen = new Label（"Green"）;
gbc.gridx = 0;
gbc.gridy = 2;
getContentPane（）.add（labelGreen, gbc）;
scrollbarGreen = new Scrollbar（Scrollbar.HORIZONTAL, 0, 1, 0, 256）;
scrollbarGreen.setValue（fgc.getGreen（））;
gbc.gridx = 1;
gbc.gridy = 2;
getContentPane（）.add（scrollbarGreen, gbc）;
scrollbarGreen.addAdjustmentListener（this）;
textFieldGreen = new TextField（3）;
textFieldGreen.setText（scrollbarGreen.getValue（） + ""）;
textFieldGreen.addTextListener（this）;
textFieldGreen.setEditable（false）;
gbc.gridx = 2;
gbc.gridy = 2;
getContentPane（）.add（textFieldGreen, gbc）;
// 初始化 Blue 标签
Label labelBlue = new Label（"Blue"）;
gbc.gridx = 0;
gbc.gridy = 3;
getContentPane（）.add（labelBlue, gbc）;
scrollbarBlue = new Scrollbar（Scrollbar.HORIZONTAL, 0, 1, 0, 256）;
scrollbarBlue.setValue（fgc.getBlue（））;
gbc.gridx = 1;
gbc.gridy = 3;
getContentPane（）.add（scrollbarBlue, gbc）;
scrollbarBlue.addAdjustmentListener（this）;
textFieldBlue = new TextField（3）;
textFieldBlue.setText（scrollbarBlue.getValue（） + ""）;
textFieldBlue.addTextListener（this）;
```

```
    textFieldBlue.setEditable（false）;
    gbc.gridx = 2;
    gbc.gridy = 3;
    getContentPane（）.add（textFieldBlue, gbc）;
    // 初始化 Ok 按钮
    colorButtonOk = new Button（"Ok"）;
    gbc.gridx = 4;
    gbc.gridy = 1;
    gbc.fill = gbc.NONE;
    getContentPane（）.add（colorButtonOk,gbc）;
    colorButtonOk.addActionListener（this）;
    // 初始化 Cancel 按钮
    colorButtonCancel = new Button（"Cancel"）;
    gbc.gridx = 4;
    gbc.gridy = 2;
    getContentPane（）.add（colorButtonCancel,gbc）;
    colorButtonCancel.addActionListener（this）;
    // 初始化 Lock RGB 按钮
    colorCheckbox = new Checkbox（"Lock RGB", false）;
    gbc.gridx = 3;
    gbc.gridy = 3;
    getContentPane（）.add（colorCheckbox,gbc）;
    colorCheckbox.addItemListener（this）;
    // 初始化颜色预览文本框
    colorTextField = new TextField（"Java awt"）;
    colorTextField.setSize（90, 60）;
    colorTextField.setForeground（fgc）;
    colorTextField.setBackground（bgc）;
    colorTextField.setFont（new Font（"Courier", Font.BOLD + Font.ITALIC, 36））;
    gbc.gridx = 3;
    gbc.gridy = 1;
    gbc.fill = gbc.BOTH;
    getContentPane（）.add（colorTextField, gbc）;
    // 添加窗体监听器
    this.addWindowListener（new WindowAdapter（） {
                                public void windowClosing（WindowEvent e） {
                                  dispose（）;
                                }
                              }）;
    this.setLocation（120, 120）;
    this.setResizable（false）;
    this.setSize（480,160）;
    // 显示窗体
    this.setVisible（true）;
    if（changed） {
      return fbgc;
    }
    else
      return fbgcOld;
}

// 事件监听方法
public void actionPerformed（ActionEvent ae） {
    if（ae.getSource（） == colorButtonOk） {
      changed = true;
```

```
            dispose（）；
        }
        else if（ae.getSource（）  == colorButtonCancel） {
            changed = false;
            dispose（）；
        }
    }

    // Item 事件监听方法
    public void itemStateChanged（ItemEvent ie） {
        if（ie.getSource（）  == fore） {
            // 前景色改变
            isFore = true;
            scrollbarRed.setValue（fbgc[0].getRed（））；
            textFieldRed.setText（scrollbarRed.getValue（）  + ""）；
            scrollbarGreen.setValue（fbgc[0].getGreen（））；
            textFieldGreen.setText（scrollbarGreen.getValue（）  + ""）；
            scrollbarBlue.setValue（fbgc[0].getBlue（））；
            textFieldBlue.setText（scrollbarBlue.getValue（）  + ""）；
            colorTextField.setForeground（setColorVarSrollbar（））；
        }
        else if（ie.getSource（）  == back） {
            // 背景色改变
            isFore = false;
            scrollbarRed.setValue（fbgc[1].getRed（））；
            textFieldRed.setText（scrollbarRed.getValue（）  + ""）；
            scrollbarGreen.setValue（fbgc[1].getGreen（））；
            textFieldGreen.setText（scrollbarGreen.getValue（）  + ""）；
            scrollbarBlue.setValue（fbgc[1].getBlue（））；
            textFieldBlue.setText（scrollbarBlue.getValue（）  + ""）；
            colorTextField.setBackground（setColorVarSrollbar（））；
        }
        else if（ie.getSource（）  == colorCheckbox） {
            if（colorCheckbox.getState（）） {
                synchronism = true;
            }
            else {
                synchronism = false;
            }
        }
    }

    //数值调整监听方法
    public void adjustmentValueChanged（AdjustmentEvent ade） {
        if（ade.getSource（）  == scrollbarRed） {
            //红色数值调整
            if（synchronism） {
                textFieldRed.setText（scrollbarRed.getValue（）  + ""）；
                scrollbarGreen.setValue（scrollbarRed.getValue（））；
                textFieldGreen.setText（scrollbarRed.getValue（）  + ""）；
                scrollbarBlue.setValue（scrollbarRed.getValue（））；
                textFieldBlue.setText（scrollbarRed.getValue（）  + ""）；
                if（isFore） {
                    colorTextField.setForeground（setColorVarSrollbar（））；
```

```
            fbgc[0] = setColorVarSrollbar（）；
        }
        else {
            colorTextField.setBackground（setColorVarSrollbar（））；
            fbgc[1] = setColorVarSrollbar（）；
        }
    }
    else {
        textFieldRed.setText（scrollbarRed.getValue（）+""）；
        if（isFore） {
            colorTextField.setForeground（setColorVarSrollbar（））；
            fbgc[0] = setColorVarSrollbar（）；
        }
        else {
            colorTextField.setBackground（setColorVarSrollbar（））；
            fbgc[1] = setColorVarSrollbar（）；
        }
    }
}
else if（ade.getSource（） == scrollbarGreen） {
    //绿色数值调整
    if（synchronism） {
        textFieldGreen.setText（scrollbarGreen.getValue（） + ""）；
        scrollbarBlue.setValue（scrollbarGreen.getValue（））；
        textFieldBlue.setText（scrollbarGreen.getValue（） + ""）；
        scrollbarRed.setValue（scrollbarGreen.getValue（））；
        textFieldRed.setText（scrollbarGreen.getValue（） + ""）；
        if（isFore） {
            colorTextField.setForeground（setColorVarSrollbar（））；
            fbgc[0] = setColorVarSrollbar（）；
        }
        else {
            colorTextField.setBackground（setColorVarSrollbar（））；
            fbgc[1] = setColorVarSrollbar（）；
        }
    }
    else {
            textFieldGreen.setText（scrollbarGreen.getValue（） + ""）；
        if（isFore） {
            colorTextField.setForeground（setColorVarSrollbar（））；
            fbgc[0] = setColorVarSrollbar（）；
        }
        else {
            colorTextField.setBackground（setColorVarSrollbar（））；
            fbgc[1] = setColorVarSrollbar（）；
        }
    }
}
else if（ade.getSource（） == scrollbarBlue） {
    //蓝色数值调整
    if（synchronism） {
        textFieldBlue.setText（scrollbarBlue.getValue（） + ""）；
        scrollbarRed.setValue（scrollbarBlue.getValue（））；
        textFieldRed.setText（scrollbarBlue.getValue（） + ""）；
        scrollbarGreen.setValue（scrollbarBlue.getValue（））；
```

```
            textFieldGreen.setText（scrollbarBlue.getValue（）  + ""）；
            if（isFore）  {
               colorTextField.setForeground（setColorVarSrollbar（））；
               fbgc[0] = setColorVarSrollbar（）；
            }
            else {
               colorTextField.setBackground（setColorVarSrollbar（））；
               fbgc[1] = setColorVarSrollbar（）；
            }
         }
         else {
            textFieldBlue.setText（scrollbarBlue.getValue（）  + ""）；
            if（isFore）  {
               colorTextField.setForeground（setColorVarSrollbar（））；
               fbgc[0] = setColorVarSrollbar（）；
            }
            else {
               colorTextField.setBackground（setColorVarSrollbar（））；
               fbgc[1] = setColorVarSrollbar（）；
            }
         }
      }
   }

   //文字改变监听方法
   public void textValueChanged（TextEvent te）  {
      if（te.getSource（）  == textFieldRed）  {
         int i = 0;
         try {
            i = Integer.parseInt（textFieldRed.getText（））；
         } catch（NumberFormatException nfe）  {
         }
         if（i < 0）
            i = 0;
         if（i > 255）
            i = 255;
         scrollbarRed.setValue（i）；
         if（synchronism）  {
            scrollbarGreen.setValue（scrollbarRed.getValue（））；
            textFieldGreen.setText（scrollbarRed.getValue（）  + ""）；
            scrollbarBlue.setValue（scrollbarRed.getValue（））；
            textFieldBlue.setText（scrollbarRed.getValue（）  + ""）；
            if（isFore）  {
               colorTextField.setForeground（setColorVarSrollbar（））；
               fbgc[0] = setColorVarSrollbar（）；
            }
            else {
               colorTextField.setBackground（setColorVarSrollbar（））；
               fbgc[1] = setColorVarSrollbar（）；
            }
         }
         else {
            if（isFore）  {
               colorTextField.setForeground（setColorVarSrollbar（））；
               fbgc[0] = setColorVarSrollbar（）；
```

```
            }
            else {
               colorTextField.setBackground（setColorVarSrollbar（））;
               fbgc[1] = setColorVarSrollbar（）;
            }
          }

        }
      }

      private Color setColorVarSrollbar（）  {
       return new Color（scrollbarRed.getValue（）,
                        scrollbarGreen.getValue（）,
                        scrollbarBlue.getValue（））;
      }
    }
```

3.3 MenuFont 类的设计

MenuFont 类主要实现编辑器字体的选取设置操作。

1. 父类和主要接口

MenuFont 被设计为对话框形式，它的窗体特性继承自 JDialog。

为了综合对字体进行设置，MenuColor 当中使用了多种 GUI 组件，包括 JTextField、JCheckBox 等，为此，需要实现一些相应监听接口来实现对各种事件的响应，这些接口包括 ItemListener、TextListener 和 ActionListener。

2. 主要方法

MenuFont 类的主要方法及其功能描述如表 3 所示。

表 3 MenuFont 类的主要方法及功能描述

方法	功能描述
void actionPerformed(ActionEvent e)	用于监听用户命令并响应
Font returnFont()	返回当前使用的字体
void itemStateChanged(ItemEvent ie)	Item 事件监听方法
void textValueChanged(TextEvent te)	文字改变监听方法

3. 基本效果

如图 3 所示为文本字体更改效果。

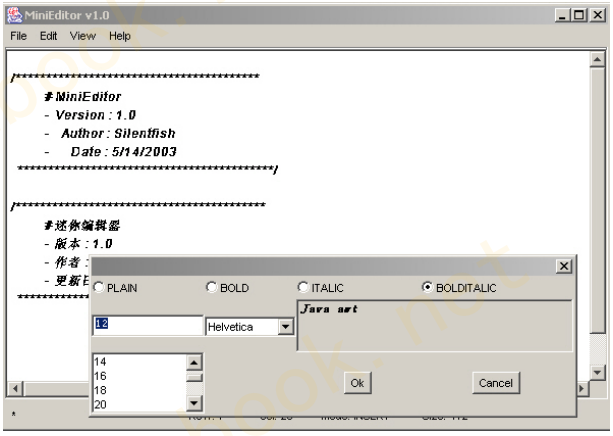

图 3 文本字体更改效果

4. 代码分析

MenuFont.java 代码如下：

```
// MenuFont.java
/*
```

```
 * 文件名:MenuFont.java
 * 说　明:实现文本字体编辑功能
 */
// 导入相关包
import java.awt.*;
import java.awt.event.*;

import javax.swing.*;
// 主类 MenuFont
public class MenuFont extends JDialog
                              implements ItemListener,
                                         ActionListener,
                                         TextListener {
  // AWT 组件声明
  CheckboxGroup gp;
  Checkbox plain, bold, italic, boldItalic;
  Choice fontNameChoice;
  TextField fontSizeTextField;
  List fontSizeList;
  TextField fontTextField;
  Button fontButtonOk;
  Button fontButtonCancel;
  // 默认字体
  int fontStyleInt = 0;
  // 更改标记
  boolean changed = true;
  // 字号范围
  int fontSizeMin = 10, fontSizeMax = 36, fontSizeChangedStep = 2;

  // 构造方法
  MenuFont（Frame frame, boolean modal） {
    super（frame, modal）;
  }

  // myLayout 方法,窗体布局
  public Font myLayout（Font taFont） {
    this.getContentPane（）.setLayout（new GridBagLayout（））;
    GridBagConstraints gbc = new GridBagConstraints（）;
    gbc.gridwidth = 1;
    gbc.gridheight =1;
    gbc.weightx = 1;
    gbc.weighty = 1;
    gbc.fill = gbc.HORIZONTAL;
    gbc.anchor = gbc.CENTER;
    //初始化字号文本框
    fontSizeTextField = new TextField（taFont.getSize（） + "",14）;
    fontSizeTextField.selectAll（）;
    fontSizeTextField.addTextListener（this）;
    gbc.gridx = 0;
    gbc.gridy = 1;
    getContentPane（）.add（fontSizeTextField, gbc）;
    //初始化字体复选框
    gp = new CheckboxGroup（）;
    if（taFont.getStyle（） == 0）
      plain = new Checkbox（"PLAIN", true, gp）;
    else
```

```
    plain = new Checkbox（"PLAIN", false, gp）;
plain.addItemListener（this）;
if（taFont.getStyle（）  == 1）
    bold = new Checkbox（"BOLD", true, gp）;
else
    bold = new Checkbox（"BOLD", false, gp）;
bold.addItemListener（this）;
if（taFont.getStyle（）  == 2）
    italic = new Checkbox（"ITALIC", true, gp）;
else
    italic = new Checkbox（"ITALIC", false, gp）;
italic.addItemListener（this）;
if（taFont.getStyle（）  == 3）
    boldItalic = new Checkbox（"BOLDITALIC", true, gp）;
else
    boldItalic = new Checkbox（"BOLDITALIC", false, gp）;
boldItalic.addItemListener（this）;
gbc.gridx = 0;
gbc.gridy = 0;
getContentPane（）.add（plain, gbc）;
gbc.gridx = 1;
gbc.gridy = 0;
getContentPane（）.add（bold, gbc）;
gbc.gridx = 2;
gbc.gridy = 0;
getContentPane（）.add（italic, gbc）;
gbc.gridx = 3;
gbc.gridy = 0;
getContentPane（）.add（boldItalic, gbc）;
//初始化字号列表
fontSizeList = new List（4, false）;
fontSizeList.addItemListener（this）;
int indexOfList = 0;
boolean breakHere = false;
for（int i = fontSizeMin; i <= fontSizeMax; i = i + fontSizeChangedStep）  {
    if（!breakHere）  {
        if（i != taFont.getSize（））  {
            indexOfList++;
        }
        else {
            breakHere = true;
        }
    }
    fontSizeList.add（i + ""）;
}
if（indexOfList < 0 ||
```

```
    indexOfList > （fontSizeMax - fontSizeMin）/fontSizeChangedStep + 1）
   fontSizeList.select（0）;
else
   fontSizeList.select（indexOfList）;
gbc.gridx = 0;
gbc.gridy = 2;
getContentPane（）.add（fontSizeList, gbc）;

gbc.gridx = 1;
gbc.gridy = 1;
fontNameChoice = new Choice（）;
fontNameChoice.addItemListener（this）;
fontNameChoice.add（"Serif"）;
fontNameChoice.add（"Courier"）;
fontNameChoice.add（"Helvetica"）;
fontNameChoice.add（"TimesRoman"）;
fontNameChoice.select（taFont.getName（））;
getContentPane（）.add（fontNameChoice, gbc）;
//初始化 Ok 按钮
fontButtonOk = new Button（"Ok"）;
fontButtonOk.addActionListener（this）;
gbc.gridx = 2;
gbc.gridy = 2;
gbc.fill = gbc.NONE;
getContentPane（）.add（fontButtonOk, gbc）;
//初始化 Cancel 按钮
fontButtonCancel = new Button（"Cancel"）;
fontButtonCancel.addActionListener（this）;
gbc.gridx = 3;
gbc.gridy = 2;
getContentPane（）.add（fontButtonCancel, gbc）;
//初始化字体选择预览文本框
fontTextField = new TextField（"Java awt"）;
fontTextField.setEditable（false）;
fontTextField.setSize（90, 60）;
fontTextField.setFont（taFont）;
gbc.weightx = 100;
gbc.weighty = 230;
gbc.gridx = 2;
gbc.gridy = 1;
gbc.gridwidth = 2;
gbc.gridheight = 1;
gbc.fill = gbc.BOTH;
getContentPane（）.add（fontTextField, gbc）;
//添加窗体监听器
this.addWindowListener（new WindowAdapter（） {
```

```
                    public void windowClosing（WindowEvent e） {
                        dispose（）;
                    }
                });
        this.setLocation（120, 120）;
        this.setResizable（false）;
        this.setSize（480,160）;
        //显示窗体
        this.setVisible（true）;
        if（changed）
            return returnFont（）;
        else
            return taFont;
    }

    //Item 监听方法
    public void itemStateChanged（ItemEvent ie） {
        if（ie.getSource（） == plain） {
            updateFontTextField（）;
        }
        else if（ie.getSource（） == bold） {
            updateFontTextField（）;
        }
        else if（ie.getSource（） == italic） {
            updateFontTextField（）;
        }
        else if（ie.getSource（） == boldItalic） {
            updateFontTextField（）;
        }
        else if（ie.getSource（） == fontNameChoice） {
            updateFontTextField（）;
        }
        else if（ie.getSource（） == fontSizeList） {
            List lf = （List）ie.getSource（）;
            fontSizeTextField.setText（lf.getSelectedItem（））;
            updateFontTextField（）;
        }

    }

    //文本改变监听方法
    public void textValueChanged（TextEvent e） {
        int indexOfList = 0;
        boolean breakHere = false;
        int thisNum = 0;
        for（int i = fontSizeMin; i <= fontSizeMax; i = i + fontSizeChangedStep） {
            if（!breakHere） {
                try {
                    thisNum = Integer.parseInt（fontSizeTextField.getText（））;
                } catch（NumberFormatException nfe） {
                    thisNum = 0;
                }
                if（i != thisNum） {
                    indexOfList++;
```

```
            }
            else {
               breakHere = true;
            }
         }
      }
      if（indexOfList < 0 ||
          indexOfList >  （fontSizeMax - fontSizeMin）/fontSizeChangedStep + 1）
         fontSizeList.select（0）;
      else
         fontSizeList.select（indexOfList）;
      updateFontTextField（）;
   }

   //事件监听器
   public void actionPerformed（ActionEvent ae）  {
      if（ae.getSource（）  == fontButtonOk）  {
          changed = true;
          dispose（）;
      }
      else if（ae.getSource（）  == fontButtonCancel）  {
         changed = false;
         dispose（）;
      }
   }
   //返回当前字体
   public Font returnFont（）  {
      updateFontTextField（）;
      return new Font（fontNameChoice.getSelectedItem（）,
                       fontStyleInt,
                       Integer.parseInt（fontSizeList.getSelectedItem（））;
   }
   //更新字体
   private void updateFontTextField（）  {
      if（gp.getSelectedCheckbox（）.getLabel（）.equals（"PLAIN"））{
         fontStyleInt = Font.PLAIN;
      }
      else if（gp.getSelectedCheckbox（）.getLabel（）.equals（"BOLD"））{
         fontStyleInt = Font.BOLD;
      }
      else if（gp.getSelectedCheckbox（）.getLabel（）.equals（"ITALIC"））{
         fontStyleInt = Font.ITALIC;
      }
      else if（gp.getSelectedCheckbox（）.getLabel（）.equals（"BOLDITALIC"））{
         fontStyleInt = Font.BOLD + Font.ITALIC;
      }
      fontTextField.setFont（new Font（fontNameChoice.getSelectedItem（）,
                                       fontStyleInt,
                                       Integer.parseInt（fontSizeList.getSelectedItem（））））;
   }

}
```

3.4 PrintableTextArea 类的设计

PrintableTextArea 是编辑器中用于编辑操作的文字编辑区域，此区域中的内容可打印。

1. 父类和主要接口

PrintableTextArea 的父类为 JTextArea。

为了使得 PrintableTextArea 中的文字可打印，该类需实现 Printable 接口。

2. 主要方法

PrintableTextArea 类的主要方法及其功能描述如表 4 所示。

表 4 PrintableTextArea 类的主要方法及功能描述

方法	功能描述
int print(Graphics g, PageFormat pf, int page)	用于打印文本框中的内容
void printIt(String str, Font font)	用于打印文本框中的内容

3. 基本效果

如图 4 所示为打印设置效果。

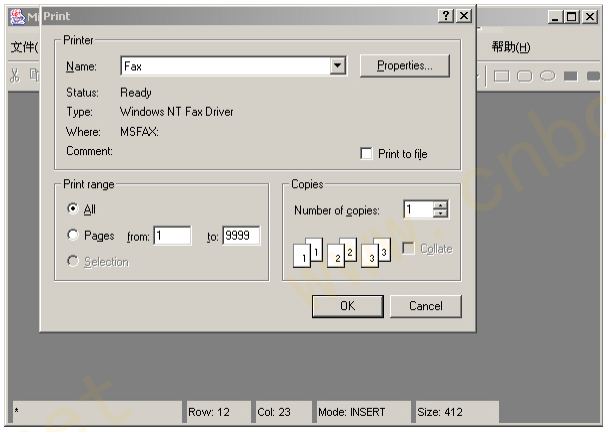

图 4 打印设置效果

4. 代码分析

PrintableTextArea.java 代码如下：

```
// PrintableTextArea.java
/*
 * 文件名:PrintableTextArea.java
 * 说　明:文本编辑框,实现了 Printable 接口,可打印文字
 */
// 导入相关包
import java.awt.*;
import java.io.*;
import java.awt.print.*;

import javax.swing.*;
import javax.swing.text.*;

// 主类 PrintableTextArea
class PrintableTextArea extends JTextArea
                        implements Printable {
  // 属性声明
  String str;
  Font font;
  PrinterJob pj;
  PageFormat defaultFormat;
  int visiblePageWidth = 0, visiblePageHeight = 0;
  final static int PageX = 10, PageY = 10;

  // 构造方法
  PrintableTextArea（int rows, int columns） {
```

```
    super（rows, columns）;
}

// 打印方法
public void printIt（String str, Font font） {
    this.str = str;
    this.font = font;
    pj = PrinterJob.getPrinterJob（）;
    defaultFormat = pj.defaultPage（）;
    pj.setPrintable（this, defaultFormat）;
    visiblePageWidth = （int）defaultFormat.getImageableWidth（）;
    visiblePageHeight = （int）defaultFormat.getImageableHeight（）;
    if（pj.printDialog（）） {
        try {
            pj.print（）;
        } catch（PrinterException e） {
        }
    }
}
// print 方法
public int print（Graphics g, PageFormat pf, int page） throws PrinterException {
    if（page >= 1）
        return Printable.NO_SUCH_PAGE;
    Graphics2D g2 = （Graphics2D） g;
    g2.translate（（int）pf.getImageableX（）, （int）pf.getImageableY（））;
    g2.setFont（font）;
    int fontHeight = 0, fontStringWidth = 0;
    int line = 0;
    String s = null;
    FontMetrics fm = getFontMetrics（font）;
    fontHeight = fm.getHeight（）;
    try{
        DataOutputStream osw = new DataOutputStream（new FileOutputStream（"$temp.$$p"））;
        osw.writeBytes（str）;
        osw.close（）;
    } catch（IOException e） {
    }
    try {
        BufferedReader br = new BufferedReader（new FileReader（"$temp.$$p"））;
        s = br.readLine（）;
        while（s != null） {
            if（s.length（） == 0）
                g2.drawString（" ", PageX, PageY + fontHeight*（line++））;
            else {
                //g2.drawString（s, PageX, PageY + fontHeight*（line++））;
                fontStringWidth = fm.stringWidth（s）;
                if（fontStringWidth > visiblePageWidth） {
                    String s1, s2;
                    int goBack = 0;
                    while（ fm.stringWidth（s1 = s.substring（0, s.length（） - goBack））>
                            visiblePageWidth） {
                        goBack++;
                    }
                    s1 = s.substring（0, s.length（） - goBack）;
                    s2 = s.substring（s.length（） - goBack - 1）;
```

```
                g2.drawString（s1, PageX, PageY + fontHeight*（line++））;
                g2.drawString（s2, PageX, PageY + fontHeight*（line++））;
              }
              else {
                g2.drawString（s, PageX, PageY + fontHeight*（line++））;
              }
            }
            s = br.readLine（）;
          }
          br.close（）;
        } catch（IOException e） {
        }
        return Printable.PAGE_EXISTS;
    }

}
```