《Java 程序设计》实验报告十

学生姓名：李季鸿　　班级：2021 级计本（3）班　　学号：20213002624

实验地点：线上　　指导教师：张春元

实验日期：2023-05-08　　共 2 学时

实验环境：Win10+JDK1.8+IntelliJ IDEA 2022.1.1

1. 实验目的

学习图形化界面编程，掌握事件驱动编程技巧；学习 I/O 编程。

2. 实验内容

（1）用集成化开发工具完成实验教材 P87 实验 3 内容。

（2）用集成化开发工具完成实验教材 P91 实验 4 内容。

（3）用集成化开发工具完成实验教材 P98 实验 1 内容。

3. 实验过程

报告撰写具体要求：截屏显示或直接写出实验 1 至实验 3 的源码和运行结果。

实验内容（1）：

LetterLabel.java:

```
package _10_.shiyan1;

/**
 * \* Created with IntelliJ IDEA.
 * \* @ProjectName: java_study_codes
 * \* @FileName: LetterLable
 * \* @author: li-jihong
 * \* Date: 2023-05-15 16:26
 */

import javax.swing.*;
import java.awt.*;
import java.awt.event.FocusEvent;
import java.awt.event.FocusListener;

public class LetterLabel extends JTextField implements FocusListener {
    LetterLabel() {
        setEditable(false);
        addFocusListener(this); //将当前对象注册为自身的焦点视器
        setBackground(Color.white);
        setFont(new Font("Arial", Font.PLAIN, 30));
    }
```

```
    public static LetterLabel[] getLetterLabel(int n) {
        LetterLabel a[] = new LetterLabel[n];
        for (int k = 0; k < a.length; k++)
            a[k] = new LetterLabel();
        return a;
    }

    public void focusGained(FocusEvent e) {
        setBackground(Color.cyan);
    }

    public void focusLost(FocusEvent e) {
        setBackground(Color.white);
    }

    public void setText(char c) {
        setText("" + c);
    }
}
```

RondomString.java:

```
package _10_.shiyan1;

/**
 * \* Created with IntelliJ IDEA.
 * \* @ProjectName: java_study_codes
 * \* @FileName: RondomString
 * \* @author: li-jihong
 * \* Date: 2023-05-15 16:30
 */
    public class RondomString { //负责随机排列单词中的字母
    String str = "";

    public String getRondomString(String s) {
        StringBuffer strBuffer = new StringBuffer(s);
        int m = strBuffer.length();
        for (int k = 0; k < m; k++) {
            int index = (int) (Math.random() * strBuffer.length());//Math.random()
返回(0,1)之间的随机数
            char c = strBuffer.charAt(index);
            str = str + c;
            strBuffer = strBuffer.deleteCharAt(index);
        }
```

```
            return str;
        }
}
```

SpellingWordFrame.java:

```
package _10_.shiyan1;

/**
 * \* Created with IntelliJ IDEA.
 * \* @ProjectName: java_study_codes
 * \* @FileName: SpellingWordFrame
 * \* @author: li-jihong
 * \* Date: 2023-05-15 16:23
 */

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;

public class SpellingWordFrame extends JFrame implements KeyListener, ActionListener {
    JTextField inputWord;
    JButton button;
    LetterLabel label[];
    JPanel northP, centerP;
    Box wordBox;
    String hintMessage = "用鼠标单击字母，按左右箭头交换字母，将其排列成所输入的单词";
    JLabel messaageLabel = new JLabel(hintMessage);
    String word = "";

    SpellingWordFrame() {
        inputWord = new JTextField(12);
        button = new JButton("确定");
        button.addActionListener(this);
        inputWord.addActionListener(this);
        northP = new JPanel();
        northP.add(new JLabel("输入单词:"));
        northP.add(inputWord);
        northP.add(button);
        centerP = new JPanel();
```

```
        wordBox = Box.createHorizontalBox();
        centerP.add(wordBox);
        add(northP, BorderLayout.NORTH);
        add(centerP, BorderLayout.CENTER);
        add(messaageLabel, BorderLayout.SOUTH);
        setBounds(100, 100, 350, 180);
        setVisible(true);
        validate();
        setDefaultCloseOperation(DISPOSE_ON_CLOSE);
    }

    public void actionPerformed(ActionEvent e) {
        word = inputWord.getText();
        int n = word.length();
        RondomString rondom = new RondomString();
        String randomWord = rondom.getRondomString(word);
        wordBox.removeAll();
        messaageLabel.setText(hintMessage);
        if (n > 0) {
            label = LetterLabel.getLetterLabel(n);
            for (int k = 0; k < label.length; k++) {
                label[k].setText("" + randomWord.charAt(k));
                wordBox.add(label[k]);
                label[k].addKeyListener(this);  //将当前窗口注册为 label[k]的键
盘监视器
            }
            validate();
            inputWord.setText(null);
            label[0].requestFocus();
        }
    }

    public void keyPressed(KeyEvent e) {
        LetterLabel sourceLabel = (LetterLabel) e.getSource();
        int index = -1;
        if (e.getKeyCode() == KeyEvent.VK_LEFT) {
            for (int k = 0; k < label.length; k++) {
                if (label[k] == sourceLabel) {
                    index = k;
                    break;
                }
            }
            if (index != 0) { //交换文本框中的字母
```

```
                String temp = label[index].getText();
                label[index].setText(label[index - 1].getText());
                label[index - 1].setText(temp);
                label[index - 1].requestFocus();
            }
        } else if (e.getKeyCode() == KeyEvent.VK_RIGHT) { //判断按下的是否是→键
            for (int k = 0; k < label.length; k++) {
                if (label[k] == sourceLabel) {
                    index = k;
                    break;
                }
            }
            if (index != label.length - 1) {
                String temp = label[index].getText();
                label[index].setText(label[index + 1].getText());
                label[index + 1].setText(temp);
                label[index + 1].requestFocus();
            }
        }
        validate();
    }

    public void keyTyped(KeyEvent e) {
    }

    public void keyReleased(KeyEvent e) {
        String success = "";
        for (int k = 0; k < label.length; k++) {
            String str = label[k].getText();
            success = success + str;
        }
        if (success.equals(word)) {
            messaageLabel.setText("恭喜你，你成功了");
            for (int k = 0; k < label.length; k++) {
                label[k].removeKeyListener(this);
                label[k].removeFocusListener(label[k]);
                label[k].setBackground(Color.white);
            }
            inputWord.requestFocus();
        }
    }
}
```

WordMainClass.java:

```
package _10_.shiyan1;

/**
 * \* Created with IntelliJ IDEA.
 * \* @ProjectName: java_study_codes
 * \* @FileName: WordMainClass
 * \* @author: li-jihong
 * \* Date: 2023-05-15 16:22
 */
public class WordMainClass {
    public static void main(String[] args) {
        new SpellingWordFrame();
    }
}
```

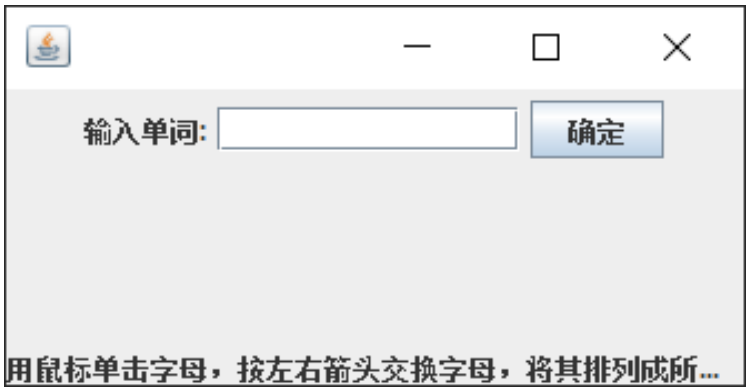


图3-1 程序运行结果

解释：再主类中创建SpellingWordFrame类，在该类中创建窗口，同时在窗口输出相关文本，同时在其中定义出能够令字母调换位置的方法，并且在字母位置正确之后，可以在窗口上输出成功的提示。在RondomString类中定义能够令字母随机生成位置的方法，从而使得屏幕上显示出随机排列好的字母。

实验后的练习

（1）增加记录用户移动字母次数的功能，即当用户拼写成功后，messageLabel标签显示的信息中包含用户移动字母的次数。

程序部分源代码：

```
public void keyPressed(KeyEvent e) {
        LetterLabel sourceLabel=(LetterLabel)e.getSource();
        int index=-1;
        if(e.getKeyCode()==KeyEvent.VK_LEFT) {
            count++;
            for(int k=0;k<label.length;k++) {
               if(label[k]==sourceLabel) {
                    index=k;
                    break;
               }
           }
```

```
            if(index!=0) { //交换文本框中的字母
                String temp=label[index].getText();
                label[index].setText(label[index-1].getText());
                label[index-1].setText(temp);
                label[index-1].requestFocus();
            }
        }
        else if(e.getKeyCode()==KeyEvent.VK_RIGHT) { //判断按下的是否是→键
             count++;
             for(int k=0;k<label.length;k++) {
                if(label[k]==sourceLabel) {
                    index=k;
                    break;
                }
            }
            if(index!=label.length-1) {
               String temp=label[index].getText();
               label[index].setText(label[index+1].getText());
               label[index+1].setText(temp);
               label[index+1].requestFocus();
            }
        }
        validate();
    }
    public void keyTyped(KeyEvent e){}
    public void keyReleased(KeyEvent e) {
        String success="";
        for(int k=0;k<label.length;k++) {
             String str=label[k].getText();
             success=success+str;
        }
        if(success.equals(word)) {
            messaageLabel.setText("恭喜你，你成功了,共计移动"+count+"次");
            for(int k=0;k<label.length;k++) {
                 label[k].removeKeyListener(this);
                 label[k].removeFocusListener(label[k]);
                 label[k].setBackground(Color.white);
            }
            inputWord.requestFocus();
        }
    }
```

程序运行结果：

图3-2 输出移动次数

解释：在keyPressed(KeyEvent e)方法中，在用户通过键盘输入“←”和“→”之后，通过count来对其进行记录，在用户调整成功后，将用户移动字母的次数输出到窗口。

实验内容（2）：

FontDialog.java:

```
package _10_.shiyan2;

/**
 * \* Created with IntelliJ IDEA.
 * \* @ProjectName: java_study_codes
 * \* @FileName: FontDialog
 * \* @author: li-jihong
 * \* Date: 2023-05-15 16:44
 */
import java.awt.event.*;
import java.awt.*;
import javax.swing.*;
public class FontDialog extends JDialog    {
        FontFamilyNames fontFamilyNames;
        int fontSize=38;
        String fontName;
        JComboBox fontNameList,fontSizeList;
        JLabel label;
        Font font;
        JButton yes,cancel;
        static int YES=1,NO=0;
        int state=-1;
        FontDialog(JFrame f) {
                super(f);
                setTitle("字体对话框");
```

```
font=new Font("宋体",Font.PLAIN,12);
fontFamilyNames=new FontFamilyNames();
setModal(true);       //当前对话框调用 setModal(boolean b)设置为有模式
yes=new JButton("Yes");
cancel=new JButton("cancel");
yes.addActionListener(new ActionListener() {
      @Override
      public void actionPerformed(ActionEvent e) {
            if(e.getSource()==yes) {
                  state=YES;
                  setVisible(false);            //对话框设置为不可见
            }
            else if(e.getSource()==cancel) {
                  state=NO;
                  setVisible(false);            //对话框设置为不可见
            }
      }
});
cancel.addActionListener(new ActionListener() {
      @Override
      public void actionPerformed(ActionEvent e) {
            if(e.getSource()==yes) {
                  state=YES;
                  setVisible(false);            //对话框设置为不可见
            }
            else if(e.getSource()==cancel) {
                  state=NO;
                  setVisible(false);            //对话框设置为不可见
            }
      }
});
label=new JLabel("hello,奥运",JLabel.CENTER);
fontNameList=new JComboBox();
fontSizeList=new JComboBox();
String name[]=fontFamilyNames.getFontName();
fontNameList.addItem("字体");
for(int k=0;k<name.length;k++)
      fontNameList.addItem(name[k]);
fontSizeList.addItem("大小");
for(int k=8;k<72;k=k+2)
      fontSizeList.addItem(new Integer(k));
fontNameList.addItemListener(new ItemListener() {
      @Override
```

```
        public void itemStateChanged(ItemEvent e) {
            if(e.getSource()==fontNameList) {
                fontName=(String)fontNameList.getSelectedItem();
                font=new Font(fontName,Font.PLAIN,fontSize);
            }
            else if(e.getSource()==fontSizeList) {
                Integer m=(Integer)fontSizeList.getSelectedItem();
                fontSize=m.intValue();
                font=new Font(fontName,Font.PLAIN,fontSize);
            }
            label.setFont(font);
            label.repaint();
            validate();
        }
    });
    fontSizeList.addItemListener(new ItemListener() {
        @Override
        public void itemStateChanged(ItemEvent e) {
            if(e.getSource()==fontNameList) {
                fontName=(String)fontNameList.getSelectedItem();
                font=new Font(fontName,Font.PLAIN,fontSize);
            }
            else if(e.getSource()==fontSizeList) {
                Integer m=(Integer)fontSizeList.getSelectedItem();
                fontSize=m.intValue();
                font=new Font(fontName,Font.PLAIN,fontSize);
            }
            label.setFont(font);
            label.repaint();
            validate();
        }
    });
    JPanel pNorth=new JPanel();
    pNorth.add(fontNameList);
    pNorth.add(fontSizeList);
    add(pNorth,BorderLayout.NORTH);
    add(label,BorderLayout.CENTER);
    JPanel pSouth=new JPanel();
    pSouth.add(yes);
    pSouth.add(cancel);
    add(pSouth,BorderLayout.SOUTH);
    setBounds(100,100,280,170);
    setDefaultCloseOperation(DISPOSE_ON_CLOSE);
```

```
            validate();
        }


        public int getState() {
            return state;
        }
        public Font getFont() {
            return font;
        }
}
```

FontFamilyNames.java:

```
package _10_.shiyan2;

/**
 * \* Created with IntelliJ IDEA.
 * \* @ProjectName: java_study_codes
 * \* @FileName: FontFamilyNames
 * \* @author: li-jihong
 * \* Date: 2023-05-15 16:45
 */
import java.awt.GraphicsEnvironment;
public class FontFamilyNames {
        String allFontNames[];
        public String [] getFontName(){
            GraphicsEnvironment ge =
GraphicsEnvironment.getLocalGraphicsEnvironment();
            allFontNames = ge.getAvailableFontFamilyNames();
            return allFontNames;
        }
}
```

FrameHaveDialog.java:

```
package _10_.shiyan2;

/**
 * \* Created with IntelliJ IDEA.
 * \* @ProjectName: java_study_codes
 * \* @FileName: FrameHaveDialog
 * \* @author: li-jihong
 * \* Date: 2023-05-15 16:45
 */
import java.awt.event.*;
import java.awt.*;
```

```
import javax.swing.*;

public class FrameHaveDialog extends JFrame {
    JTextArea text;
    JButton buttonFont;
    FontDialog dialog = new FontDialog(this);

    FrameHaveDialog() {
        buttonFont = new JButton("设置字体");
        text = new JTextArea("Java 2 实用教程（第四版）");
        buttonFont.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                if (e.getSource() == buttonFont) {

                    dialog.setVisible(true);
                    if (dialog.getState() == FontDialog.YES) {
                        text.setFont(dialog.getFont());
                        text.repaint();
                    }
                    if (dialog.getState() == FontDialog.NO) {
                        text.repaint();
                    }
                }
            }
        });
        add(buttonFont, BorderLayout.NORTH);
        add(text);
        setBounds(60, 60, 300, 300);
        setVisible(true);
        validate();
        setDefaultCloseOperation(DISPOSE_ON_CLOSE);
    }
}
```

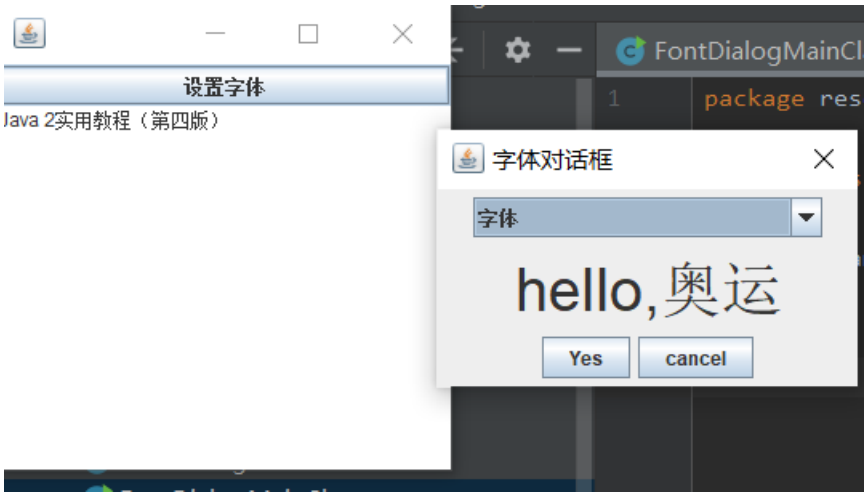
FontDialogMainClass.java:

```
package _10_.shiyan2;

/**
 * \* Created with IntelliJ IDEA.
 * \* @ProjectName: java_study_codes
 * \* @FileName: FontDialogMainClass
 * \* @author: li-jihong
 * \* Date: 2023-05-15 16:45
```

```
 */
public class FontDialogMainClass {
    public static void main(String args[]){
        FrameHaveDialog win = new FrameHaveDialog();
    }
}
```

实验结果：



解释：在主类中创建出FrameHaveDialog对象，而后生成窗口，并输出要进行字体转换的文本。然后调用FontDialog类中的方法，使得用户可以进行字体和字号的选择；最后在设置完毕后，将更改过字体和字号的文本重新输出在窗口上。

实验后的练习

（1）给上述实验中的对话框设置字体的字形（常规是LPAIN，加重BOLD，斜体ITALIC）功能。可以对Font类的static常量LPAIN、BOLD、ITALIC进行有效的运算，例如Font.BOLD+Font.ITALIC就是加重的斜体字形。

程序源代码：

```
import java.awt.event.*;
import java.awt.*;
import javax.swing.*;
public class FontDialog extends JDialog implements ItemListener,ActionListener {
    FontFamilyNames fontFamilyNames;
    int fontSize=38;
    String fontName;
    JComboBox fontNameList,fontSizeList;
    JCheckBox bold,italic;
    JLabel label;
    Font font;
```

```
JButton yes,cancel;
static int YES=1,NO=0;
int state=-1;
int textType;
FontDialog(JFrame f) {
    super(f);
    setTitle("字体对话框");
    font=new Font("宋体",Font.PLAIN,12);
    fontFamilyNames=new FontFamilyNames();
    setModal(true);//当前对话框调用setModal(boolean b)设置为有模式
    yes=new JButton("Yes");
    cancel=new JButton("cancel");
    yes.addActionListener(this);
    cancel.addActionListener(this);
    label=new JLabel("hello,奥运",JLabel.CENTER);
    fontNameList=new JComboBox();
    fontSizeList=new JComboBox();
    bold=new JCheckBox("粗体",false);
    italic=new JCheckBox("斜体",false);
    String name[]=fontFamilyNames.getFontName();
    fontNameList.addItem("字体");
    for(int k=0;k<name.length;k++)
        fontNameList.addItem(name[k]);
    fontSizeList.addItem("大小");
    for(int k=8;k<72;k=k+2)
        fontSizeList.addItem(new Integer(k));
    fontNameList.addItemListener(this);
    fontSizeList.addItemListener(this);
    bold.addItemListener(this);
    italic.addItemListener(this);
    JPanel pNorth=new JPanel();
    pNorth.add(fontNameList);
    pNorth.add(fontSizeList);
    pNorth.add(bold);
    pNorth.add(italic);
    add(pNorth,BorderLayout.NORTH);
    add(label,BorderLayout.CENTER);
    JPanel pSouth=new JPanel();
    pSouth.add(yes);
    pSouth.add(cancel);
    add(pSouth,BorderLayout.SOUTH);
    setBounds(100,100,280,170);
    setDefaultCloseOperation(DISPOSE_ON_CLOSE);
```

```
        validate();
    }
    public void itemStateChanged(ItemEvent e) {
        if(e.getSource()==fontNameList) {
            fontName=(String)fontNameList.getSelectedItem();
            font=new Font(fontName,Font.PLAIN,fontSize);
        }
        else if(e.getSource()==fontSizeList) {
            Integer m=(Integer)fontSizeList.getSelectedItem();
            fontSize=m.intValue();
            font=new Font(fontName,Font.PLAIN,fontSize);
        }
        else if(e.getSource()==bold) {
            if(bold.isSelected()) {
                if(textType==Font.PLAIN)
                    textType=Font.BOLD;
                else if(textType==Font.ITALIC)
                    textType=Font.BOLD+Font.ITALIC;
            }
            else {
                if(textType==Font.BOLD)
                    textType=Font.PLAIN;
                else if(textType==Font.BOLD+Font.ITALIC)
                    textType=Font.ITALIC;
            }
            font=new Font(fontName,textType,fontSize);
        }
        else if(e.getSource()==italic) {
            if(italic.isSelected()) {
                if(textType==Font.PLAIN)
                    textType=Font.ITALIC;
                else if(textType==Font.BOLD)
                    textType=Font.BOLD+Font.ITALIC;
            }
            else {
                if(textType==Font.ITALIC)
                    textType=Font.PLAIN;
                else if(textType==Font.BOLD+Font.ITALIC)
                    textType=Font.BOLD;
            }
            font=new Font(fontName,textType,fontSize);
        }
        label.setFont(font);
```

```
            label.repaint();
            validate();
        }
        public void actionPerformed(ActionEvent e) {
            if(e.getSource()==yes) {
                state=YES;
                setVisible(false); //对话框设置为不可见
            }
            else if(e.getSource()==cancel) {
                state=NO;
                setVisible(false);//对话框设置为不可见
            }
        }
        public int getState() {
            return state;
        }
        public Font getFont() {
            return font;
        }
}
```

程序运行结果：

图1-2 正常字体效果

图1-3 字体加粗效果

图1-4 字体倾斜效果

图1-5 字体加粗倾斜效果

解释：在FontDialog类中加入JCheckBox类的对象bold和italic，用于选择是否将字体进行加粗或者倾斜，最后在itemStateChanged(ItemEvent e)方法中用if语句进行加粗和倾斜的选择。

实验内容（3）：待第 10 章明日讲完再做

AnalysisResult.java:

```
package _10_.shiyan3;

import java.io.*;

/**
 * \* Created with IntelliJ IDEA.
 * \* @ProjectName: java_study_codes
 * \* @FileName: AnalysisResult
 * \* @author: li-jihong
 * \* Date: 2023-05-15 16:54
 */
public class AnalysisResult {
    public static void main(String args[]) {
        File fRead = new File("src/score.txt");
        File fWrite = new File("src/scoreAnalysis.txt");
        try {
            Writer out = new FileWriter(fWrite, true);
            BufferedWriter bufferWrite = new BufferedWriter(out);
            Reader in = new FileReader(fRead);
            BufferedReader butterRead = new BufferedReader(in);
            String str = null;
            while ((str = butterRead.readLine()) != null) {
                double totalScore = Fenxi.getTotalScore(str);
                str = str + "  总分:" + totalScore;
                System.out.println(str);
                bufferWrite.write(str);
                bufferWrite.newLine();
            }
```

```
                bufferWrite.close();
                bufferWrite.close();
            } catch (IOException e) {
                System.out.println(e.toString());
            }
        }
}
```

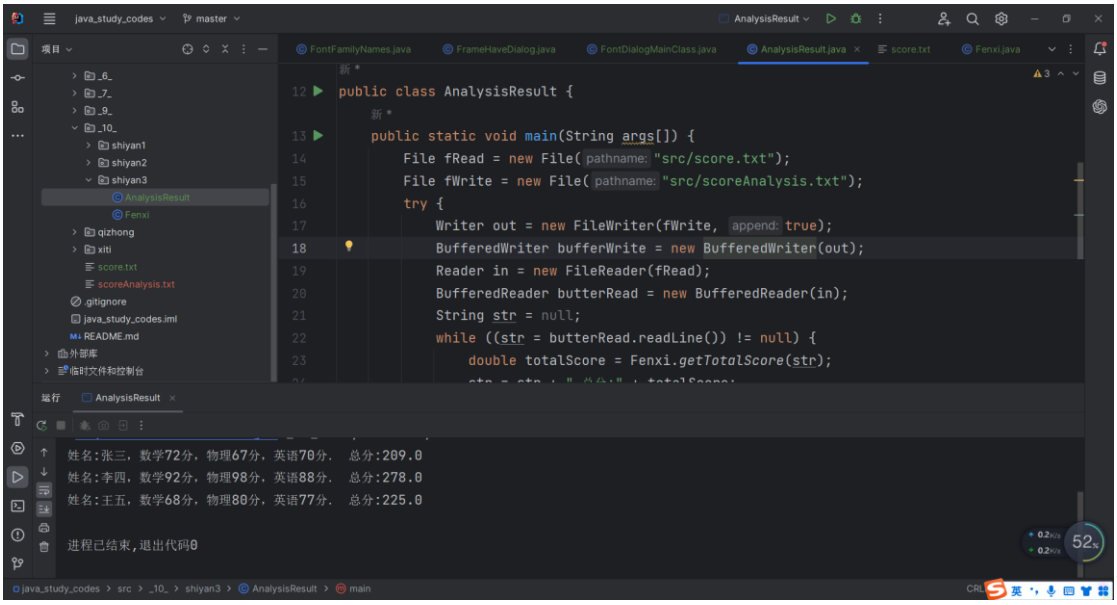
Fenxi.java:

```
package _10_.shiyan3;

/**
 * \* Created with IntelliJ IDEA.
 * \* @ProjectName: java_study_codes
 * \* @FileName: Fenxi
 * \* @author: li-jihong
 * \* Date: 2023-05-15 16:55
 */

import java.util.InputMismatchException;
import java.util.Scanner;

    public class Fenxi {
    public static double getTotalScore(String s) {
        Scanner scanner = new Scanner(s);
        scanner.useDelimiter("[^0123456789.]+");
        double totalScore = 0;
        while (scanner.hasNext()) {
            try {
                double score = scanner.nextDouble();
                totalScore = totalScore + score;
            } catch (InputMismatchException exp) {
                String t = scanner.next();
            }
        }
        return totalScore;
    }
}
```



现在有如下格式的货物明细（文本格式）goods.txt：

品名：电视，length：102cm，width：89cm，height：56cm.

品名：轿车，length：4502m，width：178cm，height：156cm.

品名：桌子，length：125m，width：78cm，height：68cm.

编写程序，按行读入货品明细，并在改行的后面尾加上该货品的体积，然后再将改行写入到一个名字为goodsVolume.txt的文件中。

程序源代码：

AnalysisResult.java

```
import java.io.*;
import java.util.*;
public class AnalysisResult {
    public static void main(String args[]) {
        File fRead = new File("D:\\计算机实验\\java\\实验10.1\\src\\goods.txt");
        File fWrite = new File("D:\\计算机实验\\java\\实验10.1\\src\\goodsVolume.txt");
        try{
            Writer out = new FileWriter(fWrite,true); //以尾加方式创建指向文件fWrite的out流
            BufferedWriter bufferWrite = new BufferedWriter(out); //创建指向out的bufferWrite流
```

```
                Reader in = new FileReader(fRead); //创建指向文件fRead的in流
                BufferedReader bufferRead = new BufferedReader(in); //创建指向in的bufferRead流
                String str = null;
                while((str=bufferRead.readLine())!=null) {
                    double volume=Fenxi.getTotalScore(str);
                    str = str+"  体积:"+volume;
                    System.out.println(str);
                    bufferWrite.write(str);
                    bufferWrite.newLine();
                }
                bufferRead.close();
                bufferWrite.close();
            }
            catch(IOException e) {
                System.out.println(e.toString());
            }
        }
    }
```

AnalysisResult.java

```
import java.util.*;
public class Fenxi {
    public static double getTotalScore(String s) {
        Scanner scanner = new Scanner(s);
        scanner.useDelimiter("[^0123456789.]+"); //p192 C8.3scanner e13
        double volume=1;
        while(scanner.hasNext()){
            try{ double data = scanner.nextDouble();
                volume = volume*data;
            }
            catch(InputMismatchException exp){
                String t = scanner.next();
```

```
                }
            }
            return volume;
        }
    }
```

程序运行结果：

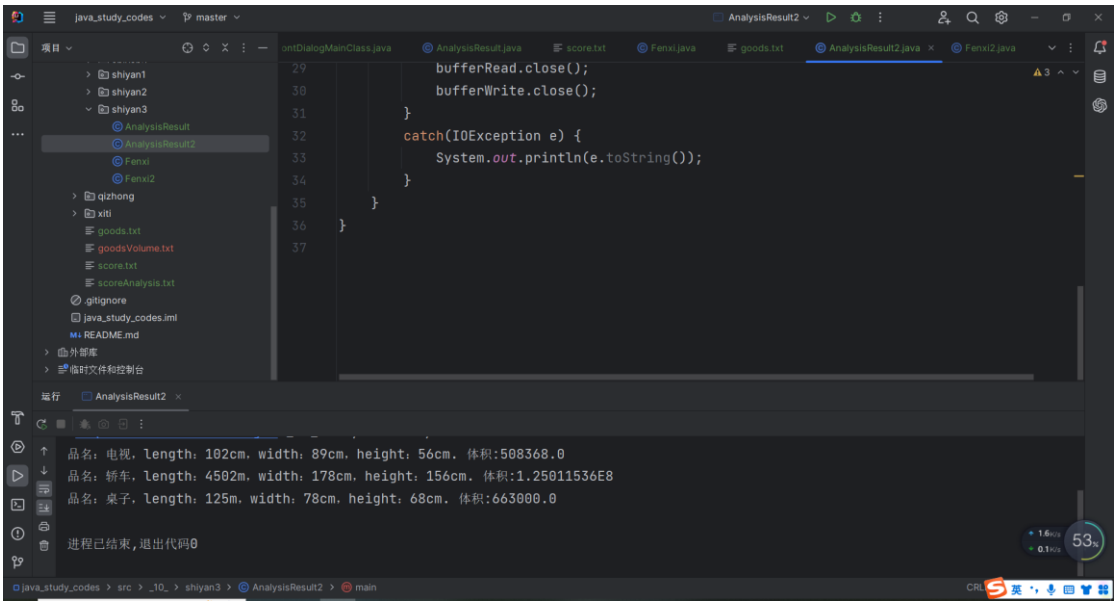


图3-2 程序运行结果

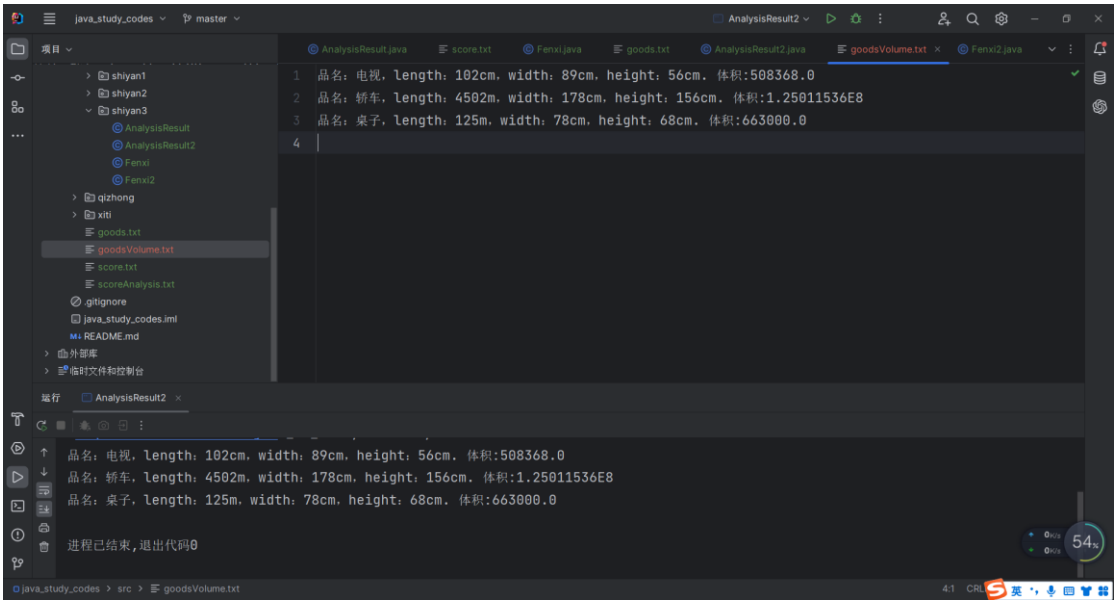


图 3-3 生成的 goodsVolume.txt 文件

4. 实验总结

写出实验中的心得体会（对第 9 章理论课重点简述）。

（1）文本输入组件

1、JTextField（文本框）

2 、JPasswordField（口令框）

3 、JTextArea（文本区）

（2）按钮与标签

1、JButton （按钮）

JButton 常用成员方法

public void setText(String text) 重新设置当前按钮的名字为 text。

public String getText() 获取当前按钮上的名字。

public void setIcon(Icon icon) 重新设置当前按钮的图标为 icon。

2、JLabel （标签）

标签是用户用来显示一小段文本、图片或两者皆有的显示区域，它只起到信息说明的作用，它本身不响应输入事件

（3）常用的选择组件有：

1、 JCheckBox（复选框）

2、JRadioButton（单选按钮） 多个单选按钮应使用 ButtonGroup 进行归组，归到同一组的单选按钮同一时刻只能有一个被选中。

3、JList（列表框）

4、JCombox（组合框）

5、JTabbedPane（选项卡）

6、JScrollbar（滚动条）

好的，以下是更具体的知识点总结：

图形化界面编程：

1. 了解图形化界面编程的基本概念和应用场景，包括窗口、控件、布局等概念。
2. 掌握常用的图形化界面编程工具或库，例如 Swing 等。
3. 学习使用图形化界面编程工具或库创建窗口、添加控件、布局、设置事件响应等。
4. 掌握常用控件的属性和方法，例如按钮、文本框、下拉列表等。
5. 学习使用布局管理器实现自适应窗口布局。
6. 了解如何将图形化界面程序打包成可执行文件。

事件驱动编程：

1. 了解事件驱动编程的基本概念和原理，即程序会根据用户的操作产生事件，并根据不同的事件执行相应的代码。
2. 掌握如何绑定控件的事件和响应函数。
3. 了解事件处理函数的基本结构和参数，例如事件对象、控件对象等。
4. 学习如何在事件处理函数中实现程序逻辑，例如读取用户输入、修改控件属性、执行计算等。

I/O 编程：

1. 了解 I/O 编程的基本概念和应用场景，包括文件读写、网络通信、进程和线程管理等。

2. 掌握 java 的文件操作，包括打开文件、读取和写入文件、关闭文件等。