第01讲 文件处理

主要内容

1. 掌握在Java文件和流的概念；
2. 掌握File类主要方法和类，Java的输入、输出流的方法和类；
3. 通过流、文件访问Java平面文件。
4. Java的泛型设计

重点内容

- 利用Java的流，访问文件与目录，掌握平面文件流的输入和输出。
-

难点分析

- 装饰器设计模式（Decorator Design Pattern）

1 Ubuntu 18.04安装e(fx)clipse

Ubuntu系统安装JDK包括以下步骤：

1、下载JDK 下载安装JDK-8u40-linux-x64.tar.gz（64位）或JDK-8u40-linux-i586.tar.gz（32位）软件。

2、解压移动到指定的文件夹

```
cd Downloads/
ls
tar zxf jdk-8u40-linux-x64.gz
```

3、切换到root用户下，将JDK移动到/usr/local/目录。

```
sudo passwd root
6083836
su
#mv jdk1.8.0_40 /usr/local/
#exit
```

4、Java环境变量配置

在~/.bashrc文件中设置PATH和JAVA_HOME变量。

```
export JAVA_HOME=/usr/local/jdk1.8.0_40
export PATH=$PATH:$JAVA_HOME/bin
```

利用Source命令让环境变量生效。

```
source ~/.bashrc
```

5、出现以下结果，表示JDK配置成功。

```
(base) ubuntu18@ubuntu18:~$ javac -version
javac 1.8.0_40
(base) ubuntu18@ubuntu18:~$ java -version
java version "1.8.0_40"
Java(TM) SE Runtime Environment (build 1.8.0_40-b26)
Java HotSpot(TM) 64-Bit Server VM (build 25.40-b25, mixed mode)
```

2、安装SeceneBuilder

1、安装javafx_scenebuilder-2_0-linux-x64.deb

2、安装后，默认的路径是

```
/opt/JavaFXSceneBuilder2.0/JavaFXSceneBuilder2.0
```

2、File对象操作

File类提供了许多与平台无关的方法来对磁盘上的文件或目录进行操作，Java为了能够做到与平台无关，克服不同系统对文件路径描述的差异。在Java语言中，使用抽象路径等概念，利用抽象路径，Java可以自动进行不同系统平台的文件路径描述与抽象文件路径之间的转换。文件类的主要作用是操作目录和目录中文件的方法，文件类不涉及文件内容的操作。文件类的方法有：删除文件，创建目录，目录文件显示，查询文件的修改时间和尺寸。Java的File类提供了方法来操纵文件和获得一个文件的信息。如getName()返回文件名，getParent()返回父目录名，exists()测试文件或目录是否存在。然而，File类是不对称的，说它不对称，意思是虽然存在允许验证一个简单文件对象属性的很多方法，但是没有响应的方法来修改这些属性，即有get无set。另外，File类还可以对目录和文件进行删除、属性修改等工作。

File类构造方法	功能描述
File(String pathname)	pathname为参数，创建文件
File(File parent, String Child)	父文件parent为参数，创建文件
File(String parent, String Child)	父目录parent为参数，创建文件
File(URL uri)	以URL为参数，创建文件

2.1 File 管理

- [ ] boolean canExecute( ) 判断文件是否可执行
- [ ] boolean canRead( ) 判断文件是否可读
- [ ] boolean canWrite( ) 判断文件是否可写
- [ ] boolean exists( ) 判断文件是否存在
- [ ] boolean isDirectory ( ) 判断对象是否是目录
- [ ] boolean isFile ( ) 判断对象是否是文件
- [ ] boolean isHidden ( )
- [ ] boolean isAbsolute ( ) 判断是否是绝对路径文件不存在也能判断

```
import java.io.File;
```

```
import java.io.IOException;
import java.util.Scanner;
import java.io.BufferedReader;
import java.io.FileReader;

public class App1_1
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        String filename = sc.next();
        File f = new File(filename);
        if(f.isDirectory())
        {
            System.out.println("f is a directory");
        }
        if(f.isFile())
        {
            try
           {
           BufferedReader in = new BufferedReader(new FileReader(f));

              String line = null;
           while((line = in.readLine())!=null)
             System.out.println("Line " + line + "\r\n");
           in.close();
           }
           catch(IOException e)
           {
            e.printStackTrace();
           }
        }
        if(!f.exists())
        {
          f.mkdirs();
        }

    }
}
```

1.2 File管理

- [x] String getName ( )
- [ ] String getPat h ( )
- [ ] String getAbsolutePat h ( )
- [ ] String getParent ( ) //如果没有父目录返回null
- [ ] long lastModified ( ) //获取最后一次修改的时间
- [ ] long length( )
- [ ] boolean renameTo(File f)
- [ ] File[ ] liseRoots ( ) //获取机器盘符

```
import java.io.File;
import java.io.IOException;
```

```
import java.util.Scanner;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FilenameFilter;

public class App1_2
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        String file = sc.next();
        File f = new File(file);
        if(f.isDirectory())
        {
            String[] files = f.list(new FilenameFilter()
            {
              public boolean accept(File f, String filename)
               {
         return filename.endsWith("txt");
         }});
            for(int i=0;i<files.length;i++)
            {
           try
           {
           BufferedReader in = new BufferedReader(new FileReader(new
File(file,files[i])));
           String line = null;
           while((line = in.readLine())!=null)
             System.out.println("Content " + line + "\r\n");
           in.close();
           }
           catch(IOException e)
           {
            e.printStackTrace();
           }
           }
            System.out.println("Directory is closed");
        }
    }
}
```

1.3 目录管理

一个File 类对象既可以表示目录也可以表示文件。mkdir创建文件夹，mkdirs根据抽象路径名创建目录。

- ☐ public boolean mkdir( ) 以pathname 为参数，创建文件夹
- ☐ public boolean mkdirs( ) 根据抽象路径名创建目录

```
import java.io.File;
import java.io.IOException;
import java.util.Scanner;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.FileInputStream;
import java.io.FileOutputStream;
```

```
public class App1_3
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        String infile = sc.next();
        File fin = new File(infile);
        String outfile = fin.getAbsolutePath() + "copy" + fin.getName();
        File fout = new File(outfile);
        if(fin.exists())
        {
            if(fin.isFile())
            {
                try
                {
                InputStream in = new FileInputStream(fin);
                OutputStream out = new FileOutputStream(fout);
                int content;
                while((content=in.read())!=-1)
                        out.write(content);
                 in.close();
                 out.close();
                }
                catch(IOException e)
                {
                  e.printStackTrace();
                }

            }
        }
    }
}
```

1.4 FilenameFilter

FilenameFilter 能够对文件扩展名进行过滤，限制li s t ( ) 方法返回的文件，使它仅返回那些与一定的文件名方式或者过滤器（filter）相匹配的文件。为达到这样的目的，必须使用l i s t ( ) 方法的带参数的重载形式：

- [ ] public File[ ] listFiles( ) 返回抽象路径名下的所有文件，返回的是F i l e 类对象
- [ ] String[ ] list ( )
- [ ] String[ ] list (Filename Filterfilter)filter 是一个实现了FilenameFilter 接口的类的对象，该接口中有一个accept( ) 方法，只有该方法判断为true 的文件名会被list 方法返回。所以可以在accept 方法中设置筛选规则。
- [ ] public String[ ] list (Filename Filterfilter) 在该形式中，fi lter 是一个实现了FilenameFilter 接口的类的对象， 该接口中有一个accept ( ) 方法，只有该方法判断为true 的文件名会被list 方法返回。所以可以在ac cept 方法中设置筛选规则。

```
import java.io.File;
import java.io.IOException;
import java.util.Scanner;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.FileInputStream;
```

```
import java.io.FileOutputStream;

public class App1_4
{
    public static void main(String[] args)
    {
        String infile = args[0];
        File fin = new File(infile);
        String outfile = args[1];
        File fout = new File(outfile);
        if(fin.exists())
        {
            if(fin.isFile())
            {
                try
                {
                InputStream in = new FileInputStream(fin);
                OutputStream out = new FileOutputStream(fout);
                int content;
                while((content=in.read())!=-1)
                        out.write(content);
                 in.close();
                 out.close();
                }
                catch(IOException e)
                {
                  e.printStackTrace();
                }

            }
        }
    }
}
```

2.1 输入、输出流

1 字节流

在Java中对文件内容进行读写的类成为输入、输出流，这些流的来源与目的都可以是文件，也可以是网络连接，甚至是内存块。Java的流主要包括字节流和字符流，其中字节流以8位字节为一个输入、输出单位。抽象类InputStream和OutputStream构成有层次结构的输入或输出类的基础，最常用的字节流类是FileInputStream和FileOutputStream。

2 字符流

字节流可以处理任何类型输入/输出操作，但是不能直接操作Unicode 字符，Unicode使用两个字节来表示一个字符，即一个字符占16位。字符流可以支持Unicode字符，Reader和Writer抽象类是字符流流的基类。Reader和Writer类也有较多的子类，与字节流类似，它们用来创建具体的字符流对象进行I/O 操作，字符流的读写等方法与字节流的相应方法类似，但读写对象使用的是字符。

2.2 txt文件处理

2.2.1 txt文件读取

通过FileInputStream流读入TXT 文件，并转化为字节流，通过JTextArea显示TXT 文件中的内容。

```
import java.io.File;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.JTextArea;
import javax.swing.JLabel;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.JButton;
import java.awt.BorderLayout;

class MyFrame extends JFrame
{
    JPanel pan,pan1,pan2;
    JButton btn,btn1;
    JTextField tf;
    JTextArea ta;
    JLabel pathlab;
    BorderLayout blay;
    MyFrame(String title)
    {
        super(title);
        setSize(500,500);
        setLocationRelativeTo(null);
        blay = new BorderLayout();
        pan = new JPanel();
        pan1 = new JPanel();
        pan2 = new JPanel();
        pathlab = new JLabel("Pls input Directory");
        tf = new JTextField(15);
        btn = new JButton("Click");
        btn1 = new JButton("Clean");
        ta = new JTextArea(20,30);
        setContentPane(pan);
        pan.setLayout(blay);
        pan.add(pan1,BorderLayout.NORTH);
        pan1.add(tf);
        pan1.add(btn);
        pan1.add(btn1);
        pan.add(pan2,BorderLayout.CENTER);
        pan2.add(ta);
        ta.setLineWrap(true);
        ActionListener action = new MyAction();
        btn.addActionListener(action);
        btn1.addActionListener(action);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(true);
    }

    private class MyAction implements ActionListener
    {
    public void actionPerformed(ActionEvent e)
    {
```

```
        if(e.getSource()==btn)
        {
            String dirstr = tf.getText();
            File f = new File(dirstr);
            if(f.isDirectory())
            {
            String[] str = f.list();
            for(int i=0;i<str.length;i++)
            ta.append(str[i]+"\r\n");
            }
            if(f.isFile())
            {
            ta.append("Input Directory Path");
            }
        }
        if(e.getSource()==btn1)
        {
            tf.setText("");
            ta.setText("");
        }
    }
    }
}
public class App1_5
{
    public static void main(String[] args)
    {
      MyFrame frm = new MyFrame("list all file in directory");
    }
}
```

2.2.2 txt文件写入

通过FileInputStream流读入TXT文件，并转化为字节流，通过FileOutputStream流写入TXT 文件。

```
import java.io.File;
import java.io.FilenameFilter;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.JTextArea;
import javax.swing.JLabel;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.JButton;
import java.awt.BorderLayout;
import java.io.IOException;
import java.util.Date;

class MyFrame extends JFrame
{
    JPanel pan,pan1,pan2;
    JButton btn,btn1,btn2;
    JTextField tf;
    JTextArea ta;
    JLabel pathlab;
    BorderLayout blay;
```

```
    MyFrame(String title)
    {
        super(title);
        setSize(500,500);
        setLocationRelativeTo(null);
        blay = new BorderLayout();
        pan = new JPanel();
        pan1 = new JPanel();
        pan2 = new JPanel();
        pathlab = new JLabel("Pls input Directory");
        tf = new JTextField(15);
        btn = new JButton("Click");
        btn1 = new JButton("Create the file");
        btn2 = new JButton("Create Directory");
        ta = new JTextArea(20,30);
        setContentPane(pan);
        pan.setLayout(blay);
        pan.add(pan1,BorderLayout.NORTH);
        pan1.add(tf);
        pan1.add(btn);
        pan1.add(btn1);
        pan1.add(btn2);
        pan.add(pan2,BorderLayout.CENTER);
        pan2.add(ta);
        ta.setLineWrap(true);
        ActionListener action = new MyAction();
        btn.addActionListener(action);
        btn1.addActionListener(action);
        btn2.addActionListener(action);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(true);
    }

    private class MyAction implements ActionListener
    {
    public void actionPerformed(ActionEvent e)
    {
        if(e.getSource()==btn)
        {
            String dirstr = tf.getText();
            File f = new File(dirstr);
            if(f.isFile())
            {
            ta.append(f.getName()+"\r\n");
            ta.append(f.getPath()+"\r\n");
            ta.append(f.getAbsolutePath()+"\r\n");
            ta.append(f.getParent()+"\r\n");
            java.text.SimpleDateFormat df = new
java.text.SimpleDateFormat("yyyy-MM-dd HH:mm:ss.SSS");
            String dateTime=df.format(new Date(f.lastModified()));
            ta.append(dateTime+"\r\n");
            ta.append(Long.toString(f.length())+"\r\n");
            }
        }
        if(e.getSource()==btn1)
        {
            String dirstr = tf.getText();
            File f2 = new File(dirstr);
```

```
            File f3 = new File(f2.getParent(),f2.getName());
            try
            {
            f3.createNewFile();
            }
            catch(IOException error)
            {error.printStackTrace();}
        }
        if(e.getSource()==btn2)
        {
            String dirstr = tf.getText();
            File f1 = new File(dirstr);
            f1.mkdir();
        }
    }
    }
}
public class App1_6
{
    public static void main(String[] args)
    {
      MyFrame frm = new MyFrame("Create the File");
    }
}
```

2.2.3 csv文件处理

1. csv文件内容读取

   通过BufferedReader流读入CSV文件，由于CSV文件中含有,分隔符。

```
import java.io.File;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.JLabel;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.GridLayout;
import javax.swing.JButton;

class MyFrame extends JFrame
{
    private GridLayout glay;
    private JPanel pane,toppane,middlepane;
    private JLabel pathlab,filelab;
    private JTextField tf,tf1;
    private JButton btn;

    MyFrame(String title)
    {
        super(title);
        setSize(300,300);
        setLocationRelativeTo(null);
        glay = new GridLayout(3,2);
        pane = new JPanel();
        pathlab = new JLabel("Pls input Path");
        filelab = new JLabel("Pls input Directory");
```

```
        tf = new JTextField(15);
        tf1 = new JTextField(15);
        btn = new JButton("Click");
        setContentPane(pane);
        setLayout(glay);
        pane.add(pathlab);
        pane.add(tf);
        pane.add(filelab);
        pane.add(tf1);
        pane.add(btn);
        ActionListener action = new MyAction();
        btn.addActionListener(action);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(true);
    }

    private class MyAction implements ActionListener
    {
    public void actionPerformed(ActionEvent e)
    {
        if(e.getSource()==btn)
        {
            String dirname = tf.getText();
            String filename = tf1.getText();
            File f = new File(dirname,filename);
            f.mkdir();
        }
    }
    }
}
public class App1_7
{
    public static void main(String[] args)
    {
      MyFrame frm = new MyFrame("make directory");
    }
}
```

2. Csv文件写入

```
import java.io.File;
import java.io.FilenameFilter;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.JTextArea;
import javax.swing.JLabel;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.JButton;
import java.awt.BorderLayout;

class MyFrame extends JFrame
{
    JPanel pan,pan1,pan2;
    JButton btn,btn1;
    JTextField tf,tf1;
```

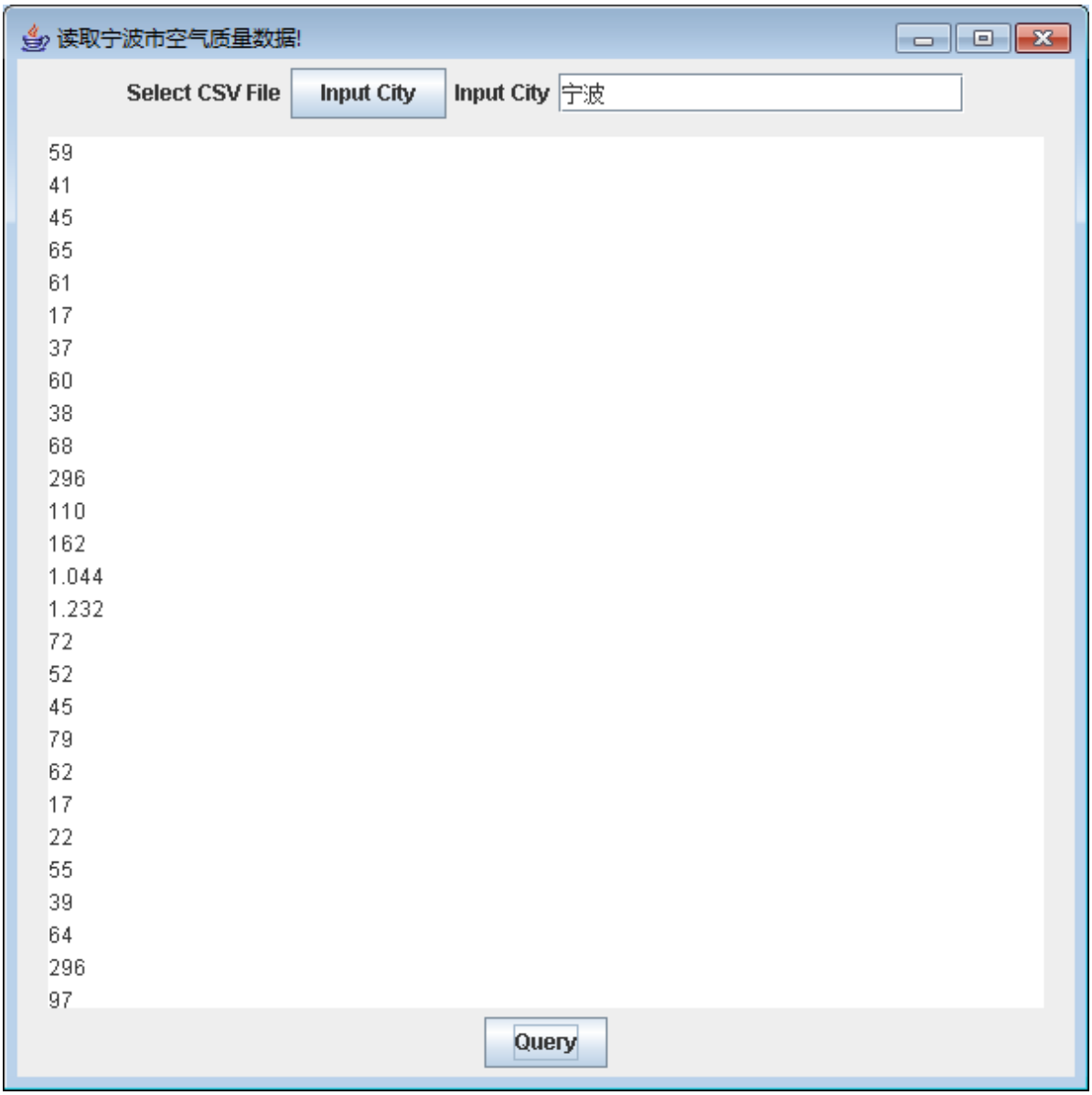
```
JTextArea ta;
JLabel pathlab;
BorderLayout blay;
MyFrame(String title)
{
    super(title);
    setSize(500,500);
    setLocationRelativeTo(null);
    blay = new BorderLayout();
    pan = new JPanel();
    pan1 = new JPanel();
    pan2 = new JPanel();
    pathlab = new JLabel("Pls input Directory");
    tf = new JTextField(15);
    tf1 = new JTextField(15);
    btn = new JButton("Click");
    btn1 = new JButton("Clean");
    ta = new JTextArea(20,30);
    setContentPane(pan);
    pan.setLayout(blay);
    pan.add(pan1,BorderLayout.NORTH);
    pan1.add(tf);
    pan1.add(tf1);
    pan1.add(btn);
    pan1.add(btn1);
    pan.add(pan2,BorderLayout.CENTER);
    pan2.add(ta);
    ta.setLineWrap(true);
    ActionListener action = new MyAction();
    btn.addActionListener(action);
    btn1.addActionListener(action);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setVisible(true);
}

private class MyAction implements ActionListener
{
public void actionPerformed(ActionEvent e)
{
    if(e.getSource()==btn)
    {
        String dirstr = tf.getText();
        File f = new File(dirstr);
        if(f.isDirectory())
        {
        String[] str = f.list(new  FilenameFilter()
        {
         public boolean accept(File file, String filename)
         {
            return filename.endsWith(tf1.getText());
         }
        });
        for(int i=0;i<str.length;i++)
        ta.append(str[i]+"\r\n");
        }
        if(f.isFile())
        {
        ta.append("Input Directory Path");
```

```
            }
        }
        if(e.getSource()==btn1)
        {
            tf.setText("");
            ta.setText("");
        }
    }
    }
}
public class App1_8
{
    public static void main(String[] args)
    {
      MyFrame frm = new MyFrame("Filter the File");
    }
}
```

3、课后作业

1、环境数据处理




```
import javax.swing.JFrame;
```

```
import javax.swing.JPanel;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.JFileChooser;
import java.awt.BorderLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.BufferedWriter;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.FileInputStream;
import java.io.IOException;

class MyFrame extends JFrame
{
    private JButton filebtn,querybtn;
    private JPanel pane,toppane,centerpane,bottompane;
    private JLabel lab,lab1;
    private JTextField tf;
    private JTextArea ta;
    private BorderLayout blay;
    MyFrame(String msg)
    {
        super(msg);
        filebtn = new JButton("Input City");
        querybtn = new JButton("Query");
        pane = new JPanel();
        toppane = new JPanel();
        centerpane = new JPanel();
        bottompane = new JPanel();
        lab = new JLabel("Select CSV File");
        lab1 = new JLabel("Input City");
        tf = new JTextField(20);
        ta = new JTextArea(30,50);
        blay = new BorderLayout();
        setSize(600,600);
        setLocationRelativeTo(null);
        setContentPane(pane);
        pane.setLayout(blay);
        pane.add(toppane,BorderLayout.NORTH);
        toppane.add(lab);
        toppane.add(filebtn);
        toppane.add(lab1);
        toppane.add(tf);
        ActionListener openAction = new openfileListener();
        filebtn.addActionListener(openAction);
        pane.add(centerpane,BorderLayout.CENTER);
        centerpane.add(ta);
        pane.add(bottompane,BorderLayout.SOUTH);
        bottompane.add(querybtn);
        ActionListener queryAction = new queryListener();
        querybtn.addActionListener(queryAction);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
        setVisible(true);
    }


private class openfileListener implements ActionListener
    {

        public void actionPerformed(ActionEvent e)
        {
            JFileChooser chooser = new JFileChooser();
            chooser.setCurrentDirectory(new File("."));
            chooser.setMultiSelectionEnabled(true);
            int result = chooser.showOpenDialog(null);
            if(result == JFileChooser.APPROVE_OPTION)
            {
                File filename = chooser.getSelectedFile();
                try
                {
                String line = null;
                BufferedReader fr =new BufferedReader(new InputStreamReader(new
FileInputStream(filename),"GBK"));
                while((line=fr.readLine())!=null)
                {
                     String[] item = line.split(",");
                     for(String a:item)
                        ta.append(a+"\n");
                }
                fr.close();
                }
                catch(IOException error)
                {
                    error.printStackTrace();
                }
            }

        }
    }

    private class queryListener implements ActionListener
    {

        public void actionPerformed(ActionEvent e)
        {
            JFileChooser chooser = new JFileChooser();
            chooser.setCurrentDirectory(new File("."));
            chooser.setMultiSelectionEnabled(true);
            int result = chooser.showOpenDialog(null);
            if(result == JFileChooser.APPROVE_OPTION)
            {
                File filename = chooser.getSelectedFile();
                try
                {
                String line = null;
                BufferedReader fr =new BufferedReader(new InputStreamReader(new
FileInputStream(filename),"GBK"));
                ta.setText("");
                int index=0;
                boolean flag = false;
```

```
                String name = tf.getText();
                while((line=fr.readLine())!=null)
                {
                     if(index==0&&flag==false)
                     {
                     String[] item = line.split(",");
                     for(int i =0 ; i<item.length; i++)
                     {
                        if(item[i].equals(name))
                        {
                            flag =true;
                          //  System.out.println( i + item[i]+"\n");
                            index = i;
                            break;
                        }
                     }
                     break;
                     }
                }
                while((line=fr.readLine())!=null)
                {
                     if(flag==true)
                     {
                     String[] item = line.split(",");
                     ta.append(item[index]+"\n");
                     }
                }
                fr.close();
                }
                catch(IOException error)
                {
                    error.printStackTrace();
                }
            }

        }
    }

}


public class App1_13
{
    public static void main(String[] args)
    {
        MyFrame frm = new MyFrame("读取宁波市空气质量数据!");
    }
}
```