

实验内容	第 13 周 Java 输入输出流			成绩	
姓 名	王秋锋	学号	2015111948	班 级	计算机 2015-03 班
专 业	计算机科学与技术			日 期	2017 年 12 月 1 日

【实验目的】--输入输出流

- ◆ 理解掌握 Java 基于字节流操作;
- ◆ 理解掌握 Java 基于字符流操作
- ◆ 理解掌握 File 类管理文件与文件夹

【实验内容】

1、编辑、编译、运行下面 java 程序（从 html 中传递参数）

理解掌握标准输入输出流

```
import java.io.*;
public class TestJava {
    public static void main(String[] args) {
        try {
            byte[] b=new byte[128];
            System.out.print("请输入字符: ");
            int count=System.in.read(b); //从标准输入流(in)读数据
            System.out.println("输入的是: ");
            for(int i=0;i<count;i++){
                System.out.print(b[i]+" ") //向标准输出流(out)写数据
            }
            System.out.println();
            for(int i=0;i<count-2;i++){
                System.out.print((char)b[i]+" ");
            }
            System.out.println();
            System.out.println("输入的字符个数为"+count);
            Class InClass=System.in.getClass(); //获取两个标准流的类信息
            Class OutClass=System.out.getClass();
            System.out.println("in 所在的类是: "+InClass.toString());
            System.out.println("out 所在的类是: "+OutClass.toString());
        } catch (IOException e) {e.printStackTrace();}
    }
}
```

要求：写出运行结果

【实验结果与分析】

实验结果：

```
请输入字符: 234asd
输入的是:
50 51 52 97 115 100 13 10
2 3 4 a s d
输入的字符个数为8
in所在的类是: class java.io.BufferedReader
out所在的类是: class java.io.PrintWriter
```

实验分析：

输出时会多出回车和换行字符，它们对应的 ASCII 码分别是 13 和 10，标准的输入输出流分别在 `java.io.BufferedReader` 和 `PrintStream` 类中

2. 编辑、编译、运行下面 java 程序

理解掌握基于自己的文件输入输出流

```
import java.io.*;

public class Write1
{
    public static void main(String args[])
    {
        try
        {
            System.out.print("please Input: ");
            int count,n=512; byte buffer[] = new byte[n];
            count = System.in.read(buffer); //读取标准输入流
            FileOutputStream wf= new FileOutputStream("d:\\test\\f1.txt", true);
            wf.write(buffer,0,count); //以追加方式写入文件输出流
            wf.close(); //关闭文件输出流
            System.out.println("Save to f1.txt!");
        }
        catch (IOException ioe)
        {
            System.out.println(ioe);
        }
        catch (Exception e)
        {
            System.out.println(e);
        }
    }
}
```

要求：分析该程序，给出运行结果：

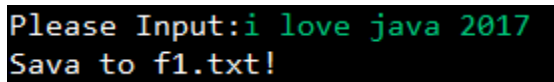
修改 `FileOutputStream wf= new FileOutputStream("d:\\test\\f1.txt", true);`

为 `FileOutputStream wf= new FileOutputStream("d:\\test\\f1.txt");`

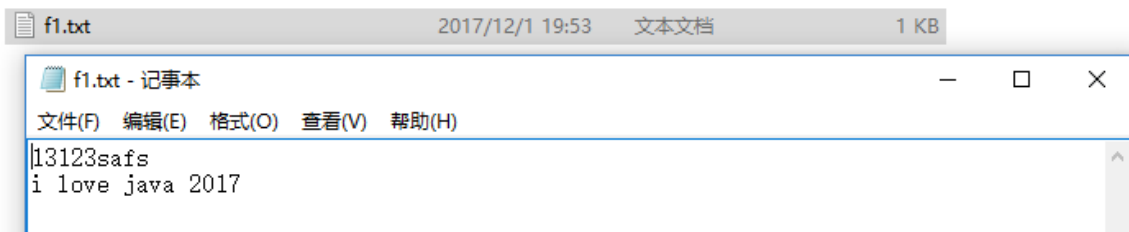
多次运行程序，理解追加方式写入与覆盖方式写入

【实验结果与分析】

实验结果：



打开 f1.txt 文件：

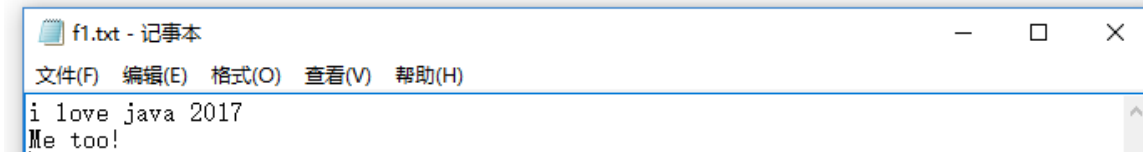


再次运行写入 Me too!

```
Please Input:Me too!  
Sava to f1.txt!
```

此时是向文件里追加内容:

f1.txt 2017/12/2 0:12 文本文档 1 KB



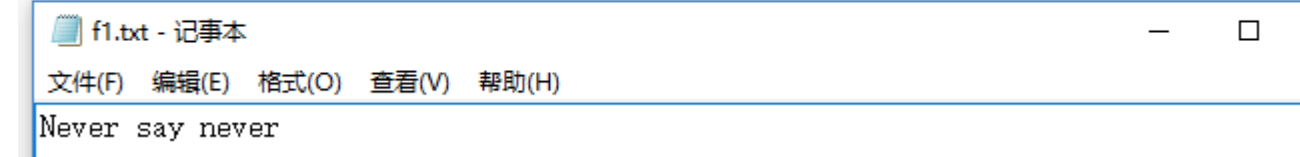
```
f1.txt - 记事本  
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)  
i love java 2017  
Me too!
```

FileOutputStream("G:\\test\\f1.txt", true)改成 FileOutputStream("G:\\test\\f1.txt"), 继续写入 Never say never

```
Please Input:Never say never  
Sava to f1.txt!
```

此时文件里的内容被覆盖:

f1.txt 2017/12/2 0:16 文本文档 1 KB



```
f1.txt - 记事本  
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)  
Never say never
```

实验分析:

1. FileOutputStream(File file)

是创建一个向指定 File 对象表示的文件中写入数据的文件输出流。

2. write(byte[] b, int off, int len)

将指定 byte 数组中从偏移量 off 开始的 len 个字节写入此文件输出流。

本程序就是将读入的数据写入到文件中, FileOutputStream("G:\\test\\f1.txt", true)有 true 就是追加方式写入, 没有就是覆盖方式写入。

3、编辑并运行下面程序, 理解掌握 File 类

```

import java.io.*;
public class TestJava
{
    public static void main(String[] args)
    {
        try{
            File curf = new File("f:\\tttt"); //在指定目录创建一文件对象不代表任何文件
            File curDir = new File(curf.getAbsolutePath()); //代表当前目录
            String[] list;
            Filter fil = new Filter(curDir, ".txt");
            // Filter fil = new Filter(curDir, ".class");
            list = curDir.listFiles(fil); //列出 curDir 中以 ".txt"结尾的文件
            for(int i = 0; i < list.length; i++) //输出清单中的文件名
                System.out.println(list[i]);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

class Filter implements FilenameFilter //实现过滤器接口的类
{
    //列出文件对象 f 表示目录的清单
    File f;
    String sf; //含过滤条件的字符串
    Filter(File f, String filter)
    {
        this.f = f;
        this.sf = filter;
    }
    public boolean accept(File dir, String sn)
    {
        return sn.endsWith(sf); //以串 sf 结束的文件满足条件
    }
}

```

要求：分析该程序，给出运行结果。

- 1、可以修改 `File curf = new File("f:\\tttt");` 指定相应的目录，对指定目录下的文件进行过滤查找。
- 2、设定不同的过滤条件，以 “..” 字符串开始的文件等。

【实验结果与分析】

实验结果：

Test 文件里有两个.txt 为后缀的文件

此电脑 > 办公 (G:) > test

名称	修改日期	类型	大小
..abc	2017/12/2 0:20	ABC 文件	0 KB
dfg.txt	2017/12/2 10:18	文本文档	0 KB
f1.txt	2017/12/2 0:16	文本文档	1 KB

程序运行后会把.txt 文件过滤出来

```

dfg.txt
f1.txt

```

将 `Filter fil = new Filter(curDir, ".txt");` 改成 `Filter fil = new Filter(curDir, "..");`
return `sn.endsWith(sf);` 改成 `return sn.startsWith(sf);` 就可以过滤出以“..”字符串开始的文件
运行之后，结果如下：



实验分析：

filter 过滤器主要用于前台向后台传递数据是的过滤操作，使用 `endsWith` 方法可以过滤出特定后缀结尾的文件，以 `startsWith` 可以过滤出特定前缀开始的文件。

4、编辑并运行下面程序，理解掌握随机文件读写操作 `RandomAccessFile`

```
import java.io.*;

class Employee { //name 共 16 个字节， age 共 4 个字节
    char name[]={'\u0000','\u0000','\u0000','\u0000',
                 '\u0000','\u0000','\u0000','\u0000'};

    int age;
    public Employee(String name,int age) throws Exception {
        if(name.toCharArray().length>8)
            System.arraycopy(name.toCharArray(),0,this.name,0,8);
        else
            System.arraycopy(name.toCharArray(),0,this.name,
                             0,name.toCharArray().length);

        this.age=age;
    }
}

public class TestJava {
    String Filename;
    public TestJava(String Filename) {
        this.Filename=Filename;
    }

    public void writeEmployee(Employee e,int n) throws Exception {
        RandomAccessFile ra=new RandomAccessFile(Filename,"rw");
        ra.seek(n*20); //将位置指示器移到指定位置上
        for(int I=0;I<8;I++) ra.writeChar (e.name[I]);
        ra.writeInt(e.age);
        ra.close();
    }

    public void readEmployee(int n) throws Exception {
        char buf[]=new char[8];
        RandomAccessFile ra=new RandomAccessFile(Filename,"r");
        ra.seek(n*20);
        for(int I=0;I<8;I++) buf[I]=ra.readChar();
        System.out.print("name:");
        System.out.println(buf);
        System.out.println("age:"+ra.readInt());
        ra.close();
    }
}
```

```
public static void main(String[] args) throws Exception {  
    TestJava t=new TestJava("f1.txt");  
    Employee e1=new Employee("ZhangSan",23);  
    Employee e2=new Employee("小不点",33);  
    Employee e3=new Employee("王华",19);  
    t.writeEmployee(e1,0);  
    t.writeEmployee(e3,2);  
    System.out.println("第一个雇员信息");  
    t.readEmployee(0);  
    System.out.println("第三个雇员信息");  
    t.readEmployee(2);  
    t.writeEmployee (e2,1);  
    System.out.println("第二个雇员信息");  
    t.readEmployee (1);  
}  
}
```

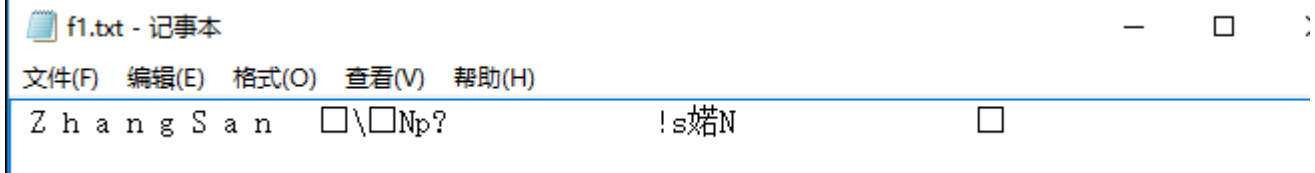
要求：分析该程序，给出运行结果

【实验结果与分析】

实验结果：

```
第一个雇员信息  
name:ZhangSan  
age:23  
第三个雇员信息  
name:王华  
age:19  
第二个雇员信息  
name:小不点  
age:33
```

打开文件：



实验分析：

RandomAccessFile 是用来访问那些保存数据记录的文件的，可以用 seek()方法来访问记录，并进行读写了。这些记录的大小不必相同；但是其大小和位置必须是可知的，但是该类仅限于操作文件。

5、编辑并运行下面程序，理解掌握缓冲流的使用

```
import java.io.*;

public class TestJava {

    public static void main(String []args){

        String f="f:\\tttt\\f.txt";
        String str="";
        int i=0;
        try{

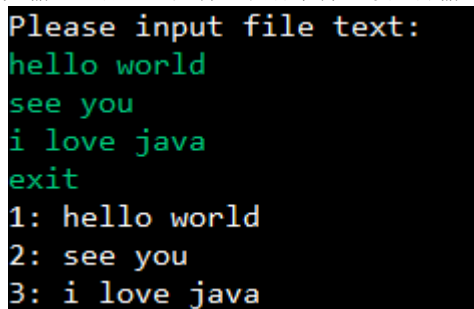
            BufferedReader keyIn=new BufferedReader(new InputStreamReader(System.in));
            BufferedWriter bw=new BufferedWriter(new FileWriter(f));
            BufferedReader br = new BufferedReader(new FileReader(f));
            System.out.println("Please input file text:");
            while(!(str=keyIn.readLine()).equals("exit"))
            {
                bw.write(str,0,str.length()); //键盘输入写入文件
                bw.newLine(); //写入一个行分隔符
            }
            bw.close();
            while((str=br.readLine())!=null) //从文件中每读一行
            {
                i++; System.out.println(i+": "+str);
            }
        }catch(IOException e){
        }
    }
}
```

要求：（1）运行程序，给出正确的程序运行结果，分析程序的功能。

【实验结果与分析】


实验结果：

先输入三行，f1 文件里没有字符，最后再输入一行 exit，打开文件，会发现写入了三行字符



```
Please input file text:
hello world
see you
i love java
exit
1: hello world
2: see you
3: i love java
```

打开文件：



```
f1.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
hello world
see you
i love java
```

实验分析：

BufferedWriter 和 BufferedReader 为带有默认缓冲的字符输出输入流。所以本程序刚开始输入字符串实现存入缓存区，直到输入“exit”，缓冲区的数据才会写入文件，然后用 BufferedReader 读出文件里的字符。

6、编辑并运行下面程序，理解掌类的串行化与对象流读写

```
import java.io.*;

class Book implements Serializable {
    int id;
    String name;
    String author;
    float price;
    public Book(int id,String name,String author,float price) {
        this.id=id;
        this.name=name;
        this.author=author;
        this.price=price;
    }
}

public class TestJava
{
    public static void main(String args[]) throws IOException,ClassNotFoundException
    {
        Book book=new Book(100032,"Java Programming Skills","Wang Sir",30);
        ObjectOutputStream oos=new ObjectOutputStream(
            new FileOutputStream("f:\\tttt\\book.dat"));

        oos.writeObject(book);
        oos.close();
        book=null;
        ObjectInputStream ois=new ObjectInputStream(
            new FileInputStream("f:\\tttt\\book.dat"));

        book=(Book)ois.readObject();
        ois.close();
        System.out.println("ID is:"+book.id);
        System.out.println("name is:"+book.name);
        System.out.println("author is:"+book.author);
        System.out.println("price is:"+book.price);
    }
}

/*
public class TestJava
{
    public static void main(String[] args) throws IOException,ClassNotFoundException
```



```
{  
  
    Student st=new Student(20010901,"黎明",20,"计算机系");//创建可串行化的 st 对象  
    FileOutputStream fout=new FileOutputStream("data.ser");  
    ObjectOutputStream sout=new ObjectOutputStream(fout);  
    try{    //输出流将对象状态存入文件"data.ser"  
        sout.writeObject(st);  
        sout.close();  
    }catch(IOException e){}  
  
    st=null;    // st 对象与 ObjectInputStream 联系起来  
    FileInputStream fin=new FileInputStream("data.ser");  
    ObjectInputStream sin=new ObjectInputStream(fin);  
    try{    //输入流将对象状态从文件"data.ser"读出  
        st=(Student)sin.readObject();  
        sin.close();  
    }catch(IOException e){}  
  
    System.out.println("学生信息:"); //输出验证  
    System.out.print("学号: " + st.id +"姓名: " + st.name+"年龄: " + st.age);  
    System.out.println("院系: " + st.department);  
}  
}  
class Student implements Serializable  
{  
    int id;  
    String name;  
    int age;  
    String department;  
    public Student (int id, String name, int age,  String department)  
    {  
        this.id=id;  
        this.name=name;  
        this.age=age;  
        this.department=department;  
    }  
    //一般可不重写，定制串行化时需要重写 writeObject()方法  
    private void writeObject(ObjectOutputStream out)throws IOException  
    {  
        out.writeInt(id);  
        out.writeInt(age);  
        out.writeUTF(name);  
        out.writeUTF(department);  
    }  
}
```

```

    }

    //一般可不重写，定制串行化时需要重写 readObject()方法
    private void readObject(ObjectInputStream in)throws IOException

    {
        id=in.readInt();
        age=in.readInt();
        name=in.readUTF();
        department=in.readUTF();
    }
}

*/

```

要求: (1) 运行程序, 给出正确的程序运行结果, 分析程序的功能。

(2) 重新定义一个 **Student** 类，至少包括学号，姓名，年龄等信息，重新运行程序，实现对象流基于文件的读写，并给出源码与程序运行结果

【实验结果与分析】

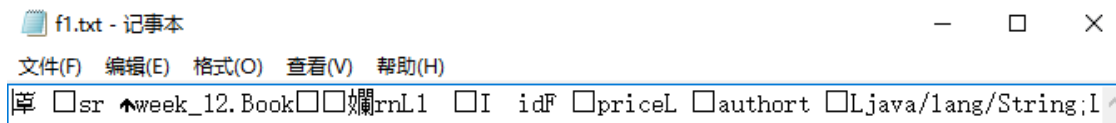
(1)

实验结果:

控制台输出：

```
ID is:100032
name is:Java Programming Skills
author is:Wang Sir
price is:30.0
```

f1 文件存入了数据:



实验分析：

`ObjectOutputStream` 将 Java 对象的基本数据类型和图形写入 `OutputStream`。可以使用 `ObjectInputStream` 读取（重构）对象。通过在流中使用文件可以实现对象的持久存储。如果流是网络套接字流，则可以在另一台主机上或另一个进程中重构对象。

只能支持 `java.io.Serializable` 接口的对象写入流中。每个 `serializable` 对象的类都被编码，编码内容包括类名和类签名、对象的字段值和数组值，以及从初始对象中引用的其他所有对象的闭包，所以打开文件会出现乱码，因为被编码了。本程序先调用 `ObjectOutputStream` 写入数据，然后再用 `ObjectInputStream` 读出数据。

(2)

实验代码:

```
import java.io.*;
class Book implements Serializable {
    int id;
    String name;
}
```

```
String author;
float price;
public Book(int id,String name,String author,float price) {
    this.id=id;
    this.name=name;
    this.author=author;
    this.price=price;
}
}

public class TestJava
{
    public static void main(String[] args) throws
IOException,ClassNotFoundException
    {
        Student st=new Student(2015111948,"王秋锋",21,"计算机科学与技术");
//创建可串行化的 st对象
        FileOutputStream fout=new FileOutputStream("G:\\test\\f1.txt");
        ObjectOutputStream sout=new ObjectOutputStream(fout);
        try{ //输出流将对象状态存入文件"data.ser"
            sout.writeObject(st);
            sout.close();
        }catch(IOException e){}

        st=null; // st对象与ObjectInputStream联系起来
        FileInputStream fin=new FileInputStream("G:\\test\\f1.txt");
        ObjectInputStream sin=new ObjectInputStream(fin);
        try{ //输入流将对象状态从文件"data.ser"读出
            st=(Student)sin.readObject();
            sin.close();
        }catch(IOException e){}

        System.out.println("学生信息:"); //输出验证
        System.out.print("学号:" + st.id + " 姓名:" + st.name+ " 年龄:" +
st.age);
        System.out.println(" 院系:" + st.department);
    }
}

class Student implements Serializable
{
    int id;
    String name;
    int age;
    String department;
    public Student (int id, String name, int age, String department)
    {
```

```
this.id=id;
this.name=name;
this.age=age;
this.department=department;
}
//一般可不重写，定制串行化时需要重写writeObject()方法
private void writeObject(ObjectOutputStream out)throws IOException
{
    out.writeInt(id);
    out.writeInt(age);
    out.writeUTF(name);
    out.writeUTF(department);
}
//一般可不重写，定制串行化时需要重写readObject()方法
private void readObject(ObjectInputStream in)throws IOException
{
    id=in.readInt();
    age=in.readInt();
    name=in.readUTF();
    department=in.readUTF();
}
}
```

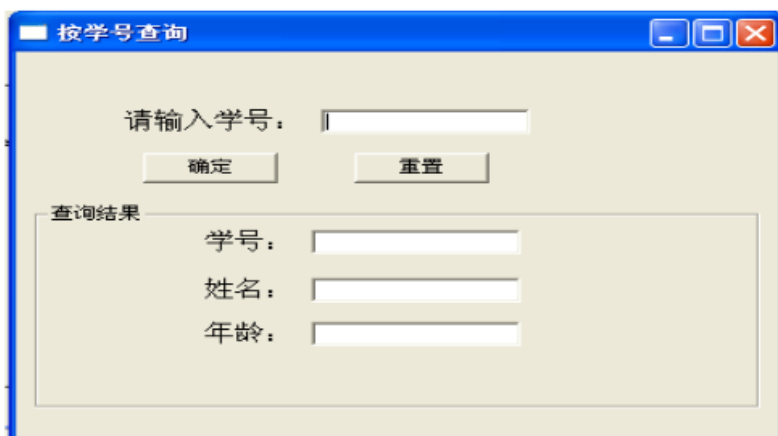
实验结果：

学生信息：
学号：2015111948 姓名：王秋锋 年龄：21 院系：计算机科学与技术

7、编写程序

定义串行化学生类，创建多个学生对象，并通过对象流写入文件来保存数据，设计如下 GUI 实现从保存学生对象流文件中查询，并把查询结果显示出来。

要求：给出源码，给出程序运行结果。



【实验结果与分析】

实验代码:

```
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JLabel;
import javax.swing.JButton;
import javax.swing.JTextField;
import javax.swing.BorderFactory;

class Student implements Serializable {
    int id;
    String name;
    int age;
    public Student(int id, String name, int age) {
        this.id = id;
        this.name = name;
        this.age = age;
    }
    //一般可不重写，定制串行化时需要重写writeObject()方法
    private void writeObject(ObjectOutputStream out) throws IOException {
        out.writeInt(id);
        out.writeInt(age);
        out.writeUTF(name);
    }
    //一般可不重写，定制串行化时需要重写readObject()方法
    private void readObject(ObjectInputStream in) throws IOException {
        id = in.readInt();
        age = in.readInt();
        name = in.readUTF();
    }
}

public class StudentSerializable extends WindowAdapter implements
ActionListener
{
    JFrame f;
    JPanel p1, p2;
    JButton b1, b2;
    JTextField t1, t2, t3, t4;
    StudentSerializable() {
        Student st1 = new Student(2015111948, "王秋锋", 21);
        Student st2 = new Student(2015111111, "Alice", 19);
    }
}
```

```
Student st3 = new Student(2017111666, "Messi", 23);

f = new JFrame("按学号查询"); f.addWindowListener(this);
p1 = new JPanel(); p1.setLayout(new FlowLayout());
p1.add(new JLabel("请输入学号: "));
t1 = new JTextField(18); p1.add(t1);
b1 = new JButton("确定"); b1.addActionListener(this);
b2 = new JButton("重置"); b2.addActionListener(this);
p1.add(b1); p1.add(b2);

p2 = new JPanel();
p2.setLayout(new FlowLayout());
p2.setBorder(BorderFactory.createTitledBorder("查询结果"));
p2.add(new JLabel("学号:"));
t2 = new JTextField(20); p2.add(t2);
p2.add(new JLabel("姓名:"));
t3 = new JTextField(20); p2.add(t3);
p2.add(new JLabel("年龄:"));
t4 = new JTextField(20); p2.add(t4);

f.setLayout(new GridLayout(2, 1));
f.add(p1); f.add(p2);
f.pack();
f.setVisible(true);

try {
    FileOutputStream fout = new FileOutputStream("G:\\test\\f1.txt");
    ObjectOutputStream sout = new ObjectOutputStream(fout);
    sout.writeObject(st1);
    sout.writeObject(st2);
    sout.writeObject(st3);
    sout.close();
} catch (IOException e){}
}

public static void main(String []args) {
    new StudentSerializable();
}

public void windowsClosing(WindowEvent e) {
    System.exit(0);
}

public void actionPerformed(ActionEvent act){
    if (act.getSource() == b2) {
        t1.setText("");
    }
    else {
        try{
```

```
Student st=null;
FileInputStream fin = new FileInputStream("G:\\test\\f1.txt");
ObjectInputStream sin = new ObjectInputStream(fin);
while ((st = (Student)sin.readObject()) != null) {
    if (String.valueOf(st.id).equals(t1.getText())) {
        t2.setText(String.valueOf(st.id));
        t3.setText(st.name);
        t4.setText(String.valueOf(st.age));
    }
}
sin.close();
}catch(IOException e){}
catch (ClassNotFoundException e) {}
}
}
```

实验结果:

输入学号 2015111948, 按下确定

按学号查询

请输入学号: 2015111948

确定 重置

查询结果

学号: 2015111948

姓名: 王秋锋

年龄: 21

按下重置

按学号查询

请输入学号:

确定 重置

再输入学号 2015111111, 按下确定

按学号查询

请输入学号: 2015111111

确定 重置

查询结果

学号: 2015111111

姓名: Alice

年龄: 19

