# 输入/输出流和文件操作

14111204056 王昊

## 实验目的

1. 理解流的概念，流的划分
2. 掌握文件描述，顺序处理，随机访问处理的方法;
3. 能够熟练的使用过滤流；掌握字符流处理的方法;
4. 理解对象串行化的概念和方法。

## 实验内容

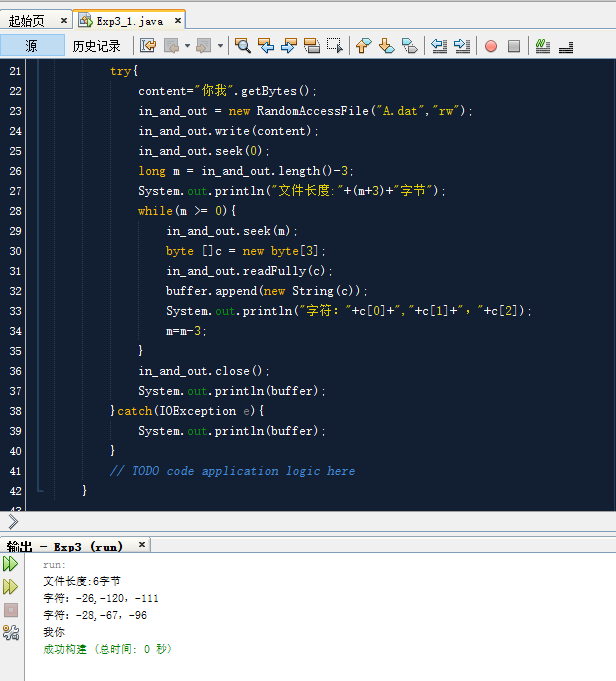
#### 仔细读下面的JAVA语言源程序，自己给出程序的运行结果。再试着上机运行，请说明该程序的功能是什么。要求：截取运行结果图到实验报告中。

编程分析：

将“你我”写入A.dat文件，显示文件长度，并把他倒序转换成byte数组输出。

源代码：（略）

截图：



调试情况分析：

A.dat文件被创建在项目根目录

仔细读下面的JAVA语言源程序，并完成程序填空。要求：截取运行结果图到实验报告中。该程序是将指定的文本串输入到指定的文件中，再从该文件中将内容倒置读出。

注：请查看项目属性中编码是否为“UTF-8”，若不是，请调整。请思考：如果编码为“GBK”，请问程序如何修改。

编程分析：

源代码：

package exp3\_2;

import java.io.\*;

public class Exp3\_2 {

public static void main(String[] args) {

RandomAccessFile in\_and\_out = null;

byte content[];

StringBuffer buffer = new StringBuffer();

try {

content = "ABC你我他".getBytes();

in\_and\_out = new RandomAccessFile("B.dat", "rw"); **//读写**

in\_and\_out.write(content);

in\_and\_out.seek(0);

long m = in\_and\_out.length(); **//获取文件流的长度**

System.out.println("文件长度:" + m + "字节");

while (m >= 0) {

m = m - 1;

in\_and\_out.seek(m); **//定位**

byte a = in\_and\_out.readByte();

if (a >= 0 && a <= 255) {

buffer.append(((char) a)); **//向buffer中加入刚获取的a**

System.out.println("字符：" + ((char) a));

} else {

m = m - 2; **//如果不是英文字符就再往前走两位**

in\_and\_out.seek(m); **//定位**

byte[] c = new byte[3];

in\_and\_out.readFully(c);

buffer.append(new String(c)); **//加入刚获取的字符**

System.out.println("字符：" + c[0] + "," + c[1] + "，" + c[2]);

}

}

in\_and\_out.close(); **//关闭文件流**

System.out.println(buffer);

} catch (IOException e) {

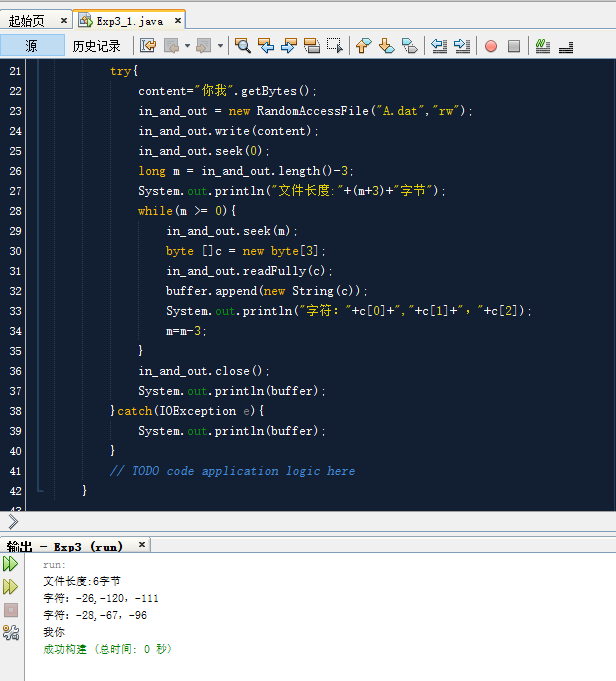
System.out.println(buffer);

}

}

}

#### 运行截图：



调试分析：

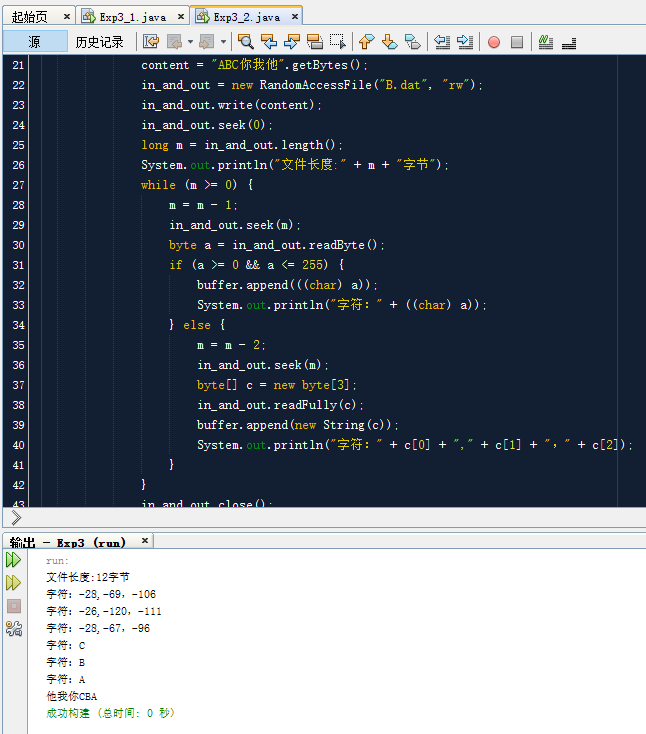
如果编码为GBK那就把content = "ABC你我他".getBytes();里面加上参数"GBK"

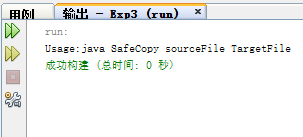
## 仔细读下面的JAVA语言源程序，自己给出程序的运行结果。再试着上机运行，请说明该程序的功能是什么。要求：截取运行结果图到实验报告中。

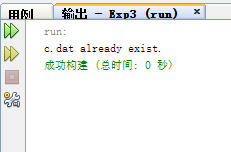
#### 编程分析：

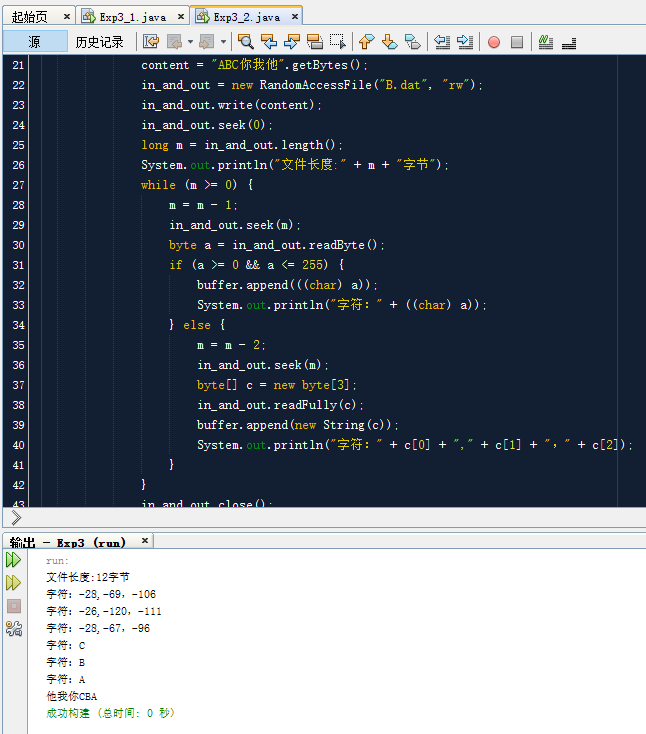
运行程序时如果获得的参数数不是2，那么报错（提示用法）。如果是两个参数，第一个参数为源文件，第二个为目标文件。将源文件复制为目标文件。如果源文件不存在，报错。如果目标文件已经存在，报错。发生其他错误的话，显示为Unknown Error。

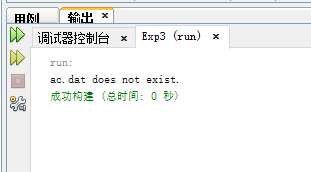
运行截图：

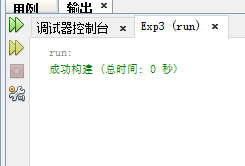


参数不正确



目标文件存在

原文件不存在

一切正常

设计一个简单的文本编辑器，要求能新建、打开和保存文本文件。

编程分析：源代码部分仅写出来题目中没有的函数……

源代码:

public void readFile(){

**//仿照writeFile部分写的**

try(FileInputStream fis = new FileInputStream(f)){

int n = -1;

byte[] a = new byte[256];

String content="";

while((n = fis.read(a,0,256))!=-1){

String append = new String(a,0,n);

content += append;

}

text.setText(content);

fis.close();

}catch(IOException e){

JOptionPane.showMessageDialog(null, "打开文件失败："+e,"错误",JOptionPane.OK\_OPTION);

}

}

**//xg就是修改，当文档内容发生了变动应该将其设置为true**

jTextArea1.getDocument().addDocumentListener(new DocumentListener() {

@Override

public void insertUpdate(DocumentEvent e) {

xg = true;

}

@Override

public void removeUpdate(DocumentEvent e) {

xg = true;

}

@Override

public void changedUpdate(DocumentEvent e) {

xg = true;

}

});

private void jMenuItem\_newActionPerformed(java.awt.event.ActionEvent evt) {

**// 如果文档内容有变动询问是否保存，然后要保存的话保存，不要的话将xg设为false，清空textArea内容**

if (xg) {

int select = JOptionPane.showConfirmDialog(null, "是否保存当前文档", "提示", JOptionPane.OK\_CANCEL\_OPTION);

if (select == 0) {

jMenuItem\_saveActionPerformed(evt);

}

}

xg = false;

jTextArea1.setText("");

}

private void jMenuItem\_saveActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

if (xg) {

int select = JOptionPane.showConfirmDialog(null, "是否保存当前文档", "提示", JOptionPane.OK\_CANCEL\_OPTION);

if (select == 0) {

jMenuItem\_saveActionPerformed(evt);

}

}

JFileChooser jfc = new JFileChooser();

jfc.showSaveDialog(null);

File f = jfc.getSelectedFile();

String s = f.getAbsolutePath();

fedit.init(s);

fedit.writeFile();

xg = false;

}

private void jMenuItem\_openActionPerformed(java.awt.event.ActionEvent evt) {

if (xg) {

int select = JOptionPane.showConfirmDialog(null, "是否保存当前文档", "提示", JOptionPane.OK\_CANCEL\_OPTION);

if (select == 0) {

jMenuItem\_saveActionPerformed(evt);

}

}

JFileChooser jfc = new JFileChooser();

jfc.showOpenDialog(null);

File f = jfc.getSelectedFile();

String s = f.getAbsolutePath();

fedit.init(s);

fedit.readFile();

xg = false;

}

private void jMenuItem\_exitActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

if (xg) {

int select = JOptionPane.showConfirmDialog(null, "是否保存当前文档", "提示", JOptionPane.OK\_CANCEL\_OPTION);

if (select == 0) {

jMenuItem\_saveActionPerformed(evt);

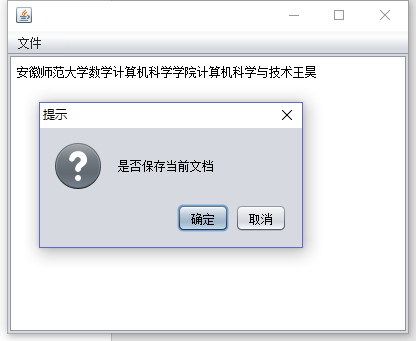
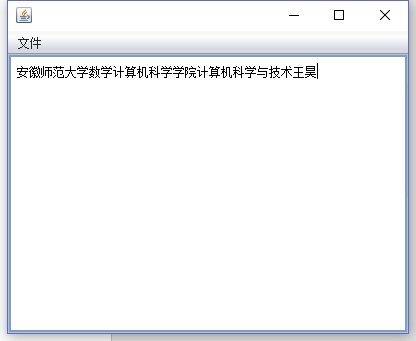
}

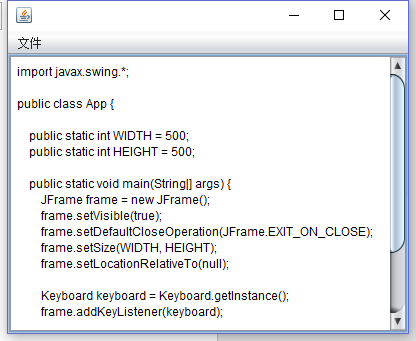
}

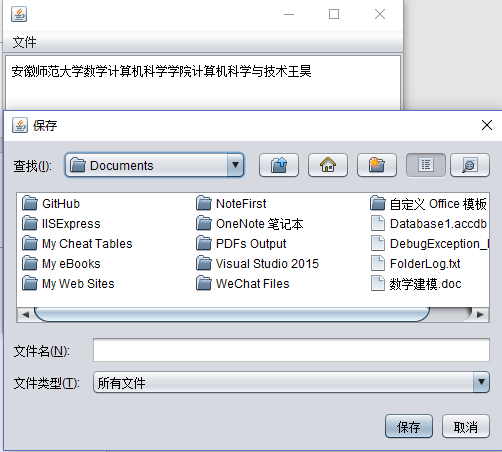
System.exit(0);

}

运行截图：



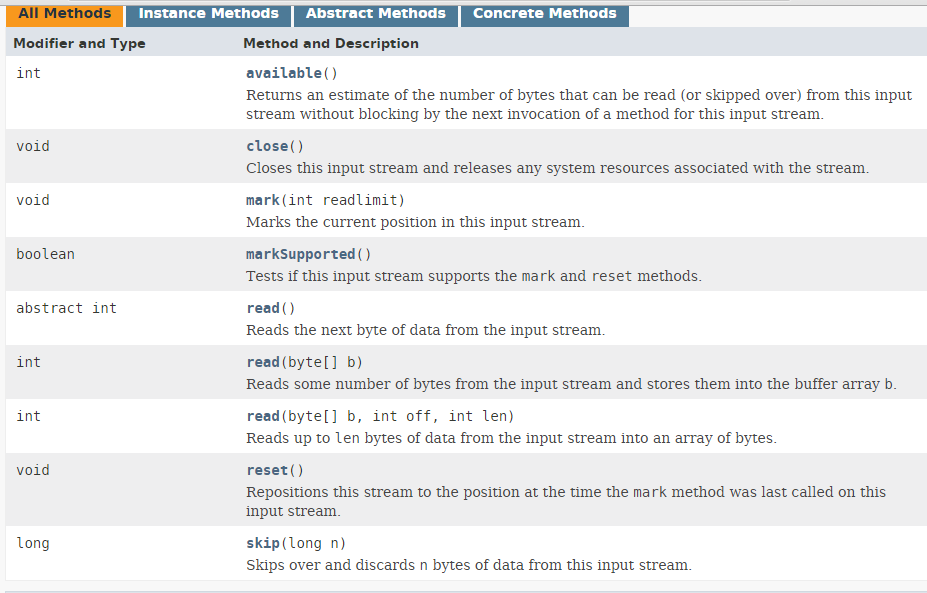
打开

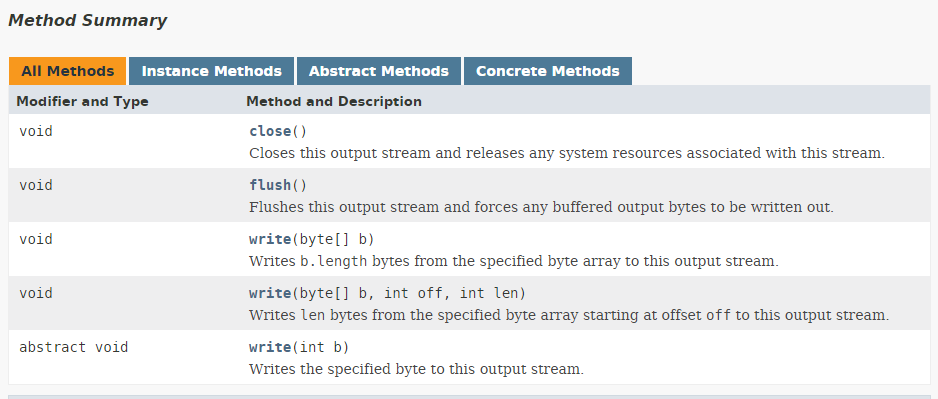


## 思考题

1. InputStream和OutputStream是抽象类吗？它们有哪些基本方法？

是。

（InputStream类）

（OutputStream类）

1. DataOutputStream类的常用方法。

|  |  |
| --- | --- |
| **Modifier and Type** | **Method and Description** |
| void | [**flush**](file:///F:\ACP134\Documents\docs\api\java\io\DataOutputStream.html#flush--)()  Flushes this data output stream. |
| int | [**size**](file:///F:\ACP134\Documents\docs\api\java\io\DataOutputStream.html#size--)()  Returns the current value of the counter written, the number of bytes written to this data output stream so far. |
| void | [**write**](file:///F:\ACP134\Documents\docs\api\java\io\DataOutputStream.html#write-byte:A-int-int-)(byte[] b, int off, int len)  Writes len bytes from the specified byte array starting at offset off to the underlying output stream. |
| void | [**write**](file:///F:\ACP134\Documents\docs\api\java\io\DataOutputStream.html#write-int-)(int b)  Writes the specified byte (the low eight bits of the argument b) to the underlying output stream. |
| void | [**writeBoolean**](file:///F:\ACP134\Documents\docs\api\java\io\DataOutputStream.html#writeBoolean-boolean-)(boolean v)  Writes a boolean to the underlying output stream as a 1-byte value. |
| void | [**writeByte**](file:///F:\ACP134\Documents\docs\api\java\io\DataOutputStream.html#writeByte-int-)(int v)  Writes out a byte to the underlying output stream as a 1-byte value. |
| void | [**writeBytes**](file:///F:\ACP134\Documents\docs\api\java\io\DataOutputStream.html#writeBytes-java.lang.String-)([**String**](file:///F:\ACP134\Documents\docs\api\java\lang\String.html) s)  Writes out the string to the underlying output stream as a sequence of bytes. |
| void | [**writeChar**](file:///F:\ACP134\Documents\docs\api\java\io\DataOutputStream.html#writeChar-int-)(int v)  Writes a char to the underlying output stream as a 2-byte value, high byte first. |
| void | [**writeChars**](file:///F:\ACP134\Documents\docs\api\java\io\DataOutputStream.html#writeChars-java.lang.String-)([**String**](file:///F:\ACP134\Documents\docs\api\java\lang\String.html) s)  Writes a string to the underlying output stream as a sequence of characters. |
| void | [**writeDouble**](file:///F:\ACP134\Documents\docs\api\java\io\DataOutputStream.html#writeDouble-double-)(double v)  Converts the double argument to a long using the doubleToLongBits method in class Double, and then writes that long value to the underlying output stream as an 8-byte quantity, high byte first. |
| void | [**writeFloat**](file:///F:\ACP134\Documents\docs\api\java\io\DataOutputStream.html#writeFloat-float-)(float v)  Converts the float argument to an int using the floatToIntBits method in class Float, and then writes that int value to the underlying output stream as a 4-byte quantity, high byte first. |
| void | [**writeInt**](file:///F:\ACP134\Documents\docs\api\java\io\DataOutputStream.html#writeInt-int-)(int v)  Writes an int to the underlying output stream as four bytes, high byte first. |
| void | [**writeLong**](file:///F:\ACP134\Documents\docs\api\java\io\DataOutputStream.html#writeLong-long-)(long v)  Writes a long to the underlying output stream as eight bytes, high byte first. |
| void | [**writeShort**](file:///F:\ACP134\Documents\docs\api\java\io\DataOutputStream.html#writeShort-int-)(int v)  Writes a short to the underlying output stream as two bytes, high byte first. |
| void | [**writeUTF**](file:///F:\ACP134\Documents\docs\api\java\io\DataOutputStream.html#writeUTF-java.lang.String-)([**String**](file:///F:\ACP134\Documents\docs\api\java\lang\String.html) str)  Writes a string to the underlying output stream using [**modified UTF-8**](file:///F:\ACP134\Documents\docs\api\java\io\DataInput.html#modified-utf-8) encoding in a machine-independent manner. |

3、字符流和字节流的区别是什么？

字符流的底层是字节流。字符流的读取单位是字符，字节流读取的单位是字节。

4、对文件进行读写操作的过程是什么？

打开文件 -- 产生句柄 -- 权限 -- 转换成流 -- 读取 --- 关闭句柄 -- 关闭文件  
  
打开文件 -- 产生句柄 -- 权限 -- 加锁 --- 是否追加写入 -- 写流 --- 关闭句柄 -- 关闭文件