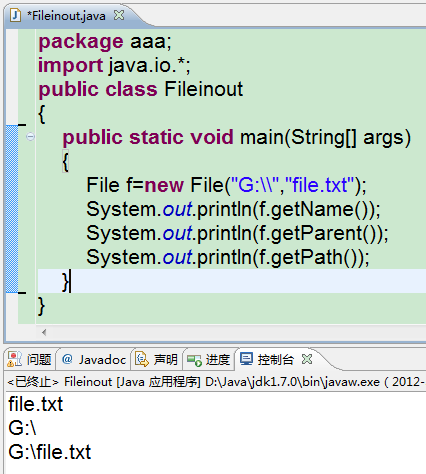
首先要引入java.io

1. 要使用到的类：File\InputStream\OutputStream\FilterStream\Writer\Reader



创建File对象的方法;

File f=**new** File("G:\\","file.txt");

File f1=**new** File("G:\\file.txt");

f.createNewFile();//真正创建文件

1. File类的主要方法：

|  |  |
| --- | --- |
| 名称 | 说明 |
| Boolean canread() | 测试文件是否可读 |
| Boolean canwrite() | 测试文件是否可写 |
| Boolean createNewFile() | 看这个文件或目录是否存在 |
| Boolean delete() | 删除文件，成功返回true,否则返回false |
| Boolean exists() | 测试当前对象所指文件是否存在true/false |
| String getName() | 获取文件或目录的名称,不包含路径 |
| String getPatent() | 获取文件所在目录的名称 |
| String getPath() | 获取文件或目录的名称(包含路径) |
| Boolean isDirectory() | 测试当前对象是否为目录 |
| Boolean isFile() | 测试当前对象是否为文件 |
| Boolean isHidden() | 测试当前对象是否为隐藏文件 |
| Long lastModified() | 返回文件的最后修改日期 |
| Long length() | 返回文件大小 |
| Boolean mkdir() | 在当前对象指定的路径上创建一个目录 |
| Boolean renameTo(File dest) | 将当前所指的文件名改为dest |
| Boolean setReadOnly() | 将当前文件设为只读 |

**package** aaa;

**import** java.io.\*;

**public** **class** Fileinout

{

**public** **static** **void** main(String[] args) **throws** IOException

{

File f=**new** File("G:\\","file.txt");

f.createNewFile();

System.*out*.println(f.getName());

System.*out*.println(f.getParent());

System.*out*.println(f.getPath());

System.*out*.println(f.lastModified());

System.*out*.println(f.length());

**if**(f.isDirectory())

System.*out*.println("是目录");

**if**(f.isFile())

System.*out*.println("是文件");

**if**(f.exists())

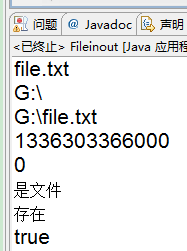
System.*out*.println("存在");

**boolean** flag=f.delete();

System.*out*.println(flag);

}

}



1. 文件创立好后，就要进行读写操作，通过流来完成。

inputStream类的主要方法

|  |  |
| --- | --- |
| 方法名称 | 说明 |
| Int read() | 自输入流读取并返回0-255之间的int整数，若文件读完了或者再无可读数据返回-1 |
| Int read(byte b[]) | 从输入流中读取数据并存在字节数组b中，它的返回值是所读取的byte数。 |
| Int read(byte[] b,int off,int len) | 从输入流中读取len个字节的数据，并存入字节数组b中，并从字节数组的第off位开始存储，它的返回值是所读取的字节数 |
| Void close() | 关闭输入流，并释放资源 |

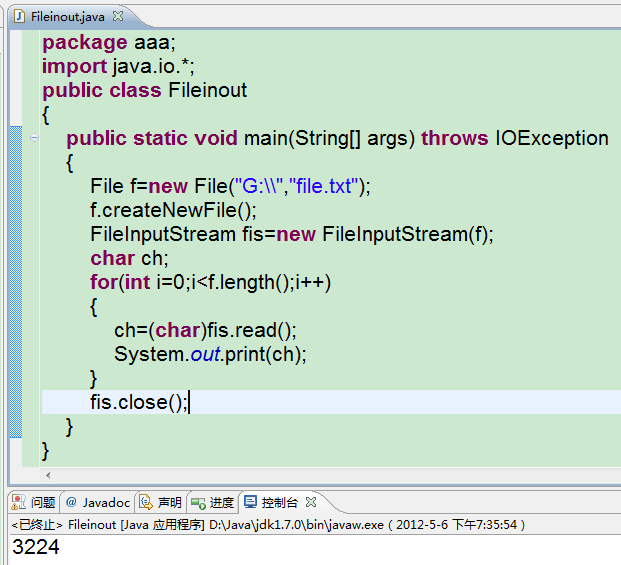
outputStream类的主要方法

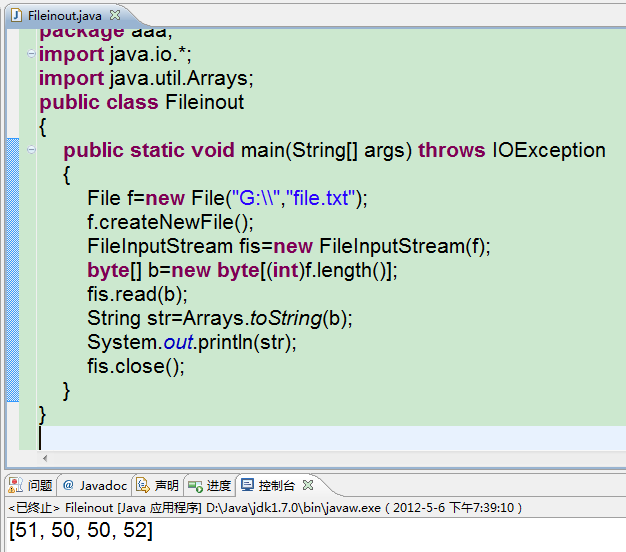
|  |  |
| --- | --- |
| 方法名称 | 说明 |
| Int write(int b) | 将b转换成字节，然后写到输出流中，每次写一个字节 |
| Int write(byte b[]) | 将字节数组中的数据写到输出流中 |
| Int write(byte b[],int off,int len) | 将字节数组中从第off位置开始长度为len个字节的数据写入到输出流中 |
| Void close() | 关闭输出流，并释放资源 |

inputStrean和outputStream是抽象类，而FileInputStream和FileOutputStream是他们的子类，不是抽象类。

FileInputStream fis=**new** FileInputStream(f);

//创建文件输入流，就是读数据

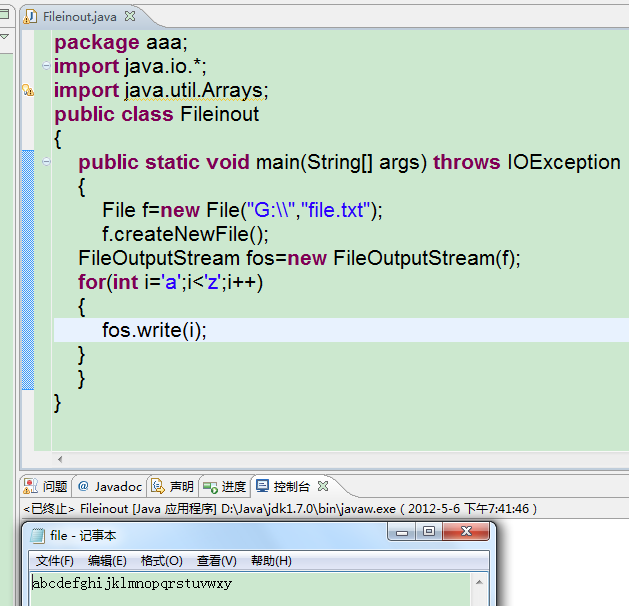




50代表2

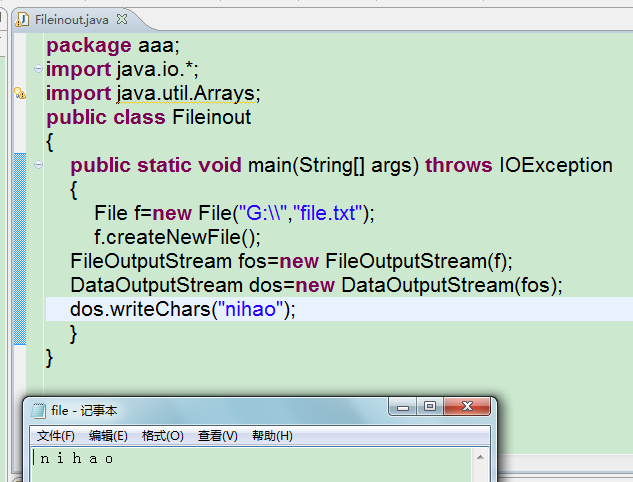
数据输出流：

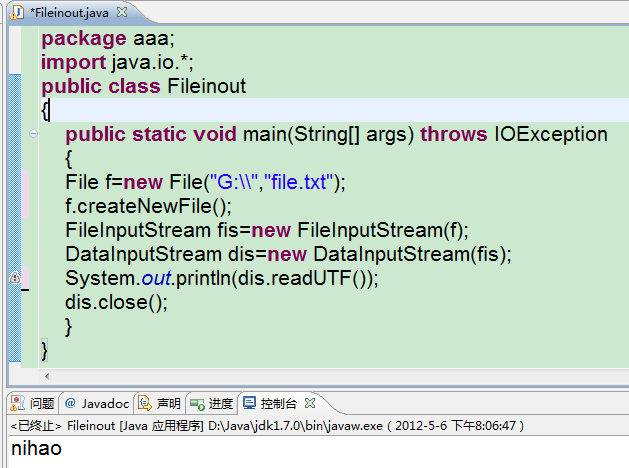
FileOutputStream fos=**new** FileOutputStream(f);



上面都是单字节操作。

1. 处理整型、字符串型数据----Data





上面都是按字节进行操作的

下面是按字符=2个字节进行操作。

Reader类和Writer类

Reader类的常用方法：

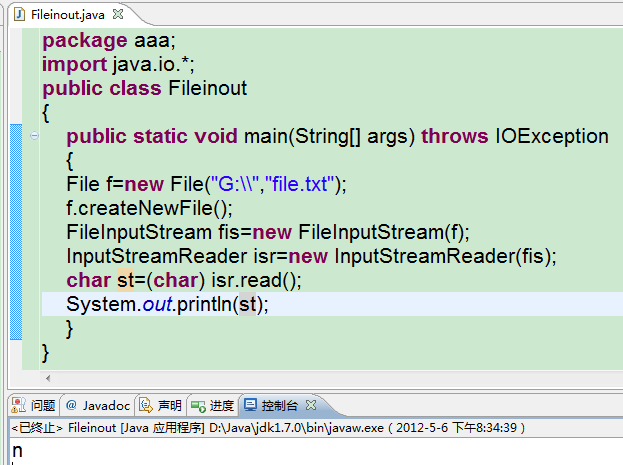
|  |  |
| --- | --- |
| 方法名 | 说明 |
| Int read() | 自输入流读取一个字符，如果无数据，返回-1 |
| Int read(char cbuf[]) | 自输入流读取cbuf这个数组长度的字符，并存在数组中，如果无数据，返回-1 |

Writer类的方法：

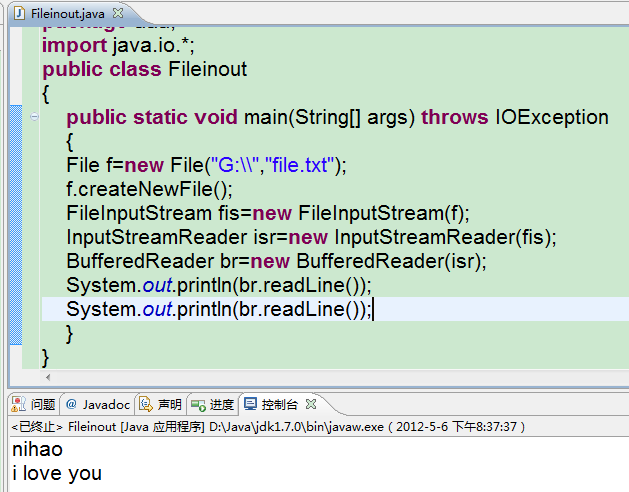
|  |  |
| --- | --- |
| 方法名 | 说明 |
| Int write(int c) | 将字符c写入到输出流 |
| Int write(char cbuf[]) | 将字符数组中的字符写入输出流 |
| Void write(String str) | 将字符串str写入输出流 |

Reader和Writer都是抽象类。

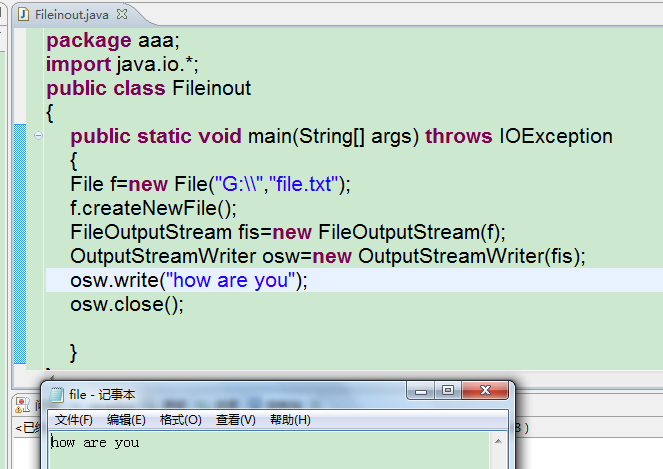
Reader可以形成两个类：InputStreamReader和BufferedReader



BufferedReader 常用一个方法：readLine();



Writer为抽象类，可以形成两个类：OutputStreamWriter和BufferedWriter

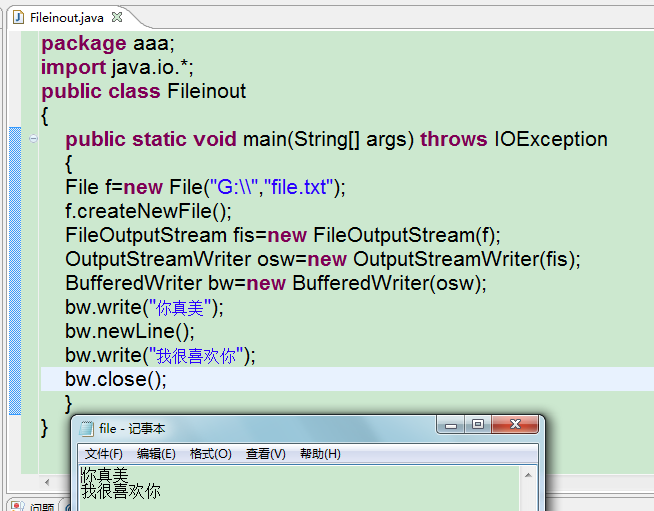


BufferedWriter类的三个方法;

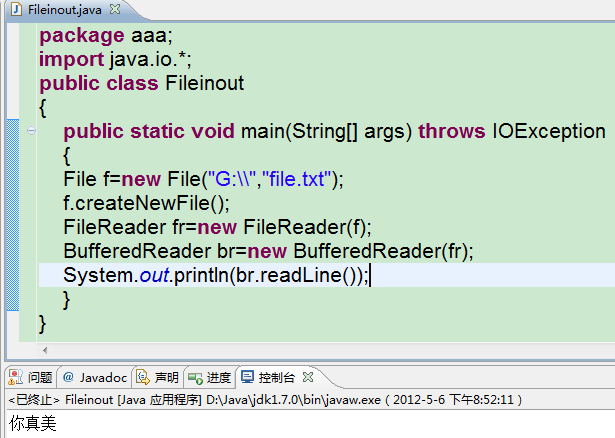
Public void write(String str):将字符串写入输出流

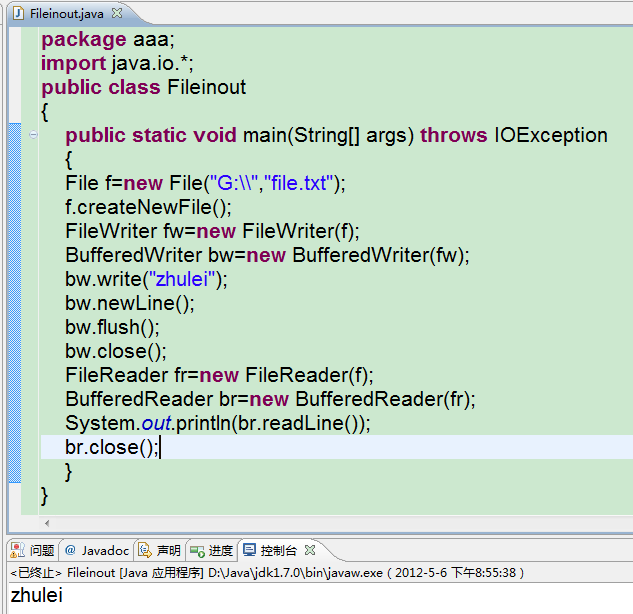
Public void flush():将缓冲区的数据强制写入输出流

Public void newline():想输出流中写入一个行结束标记

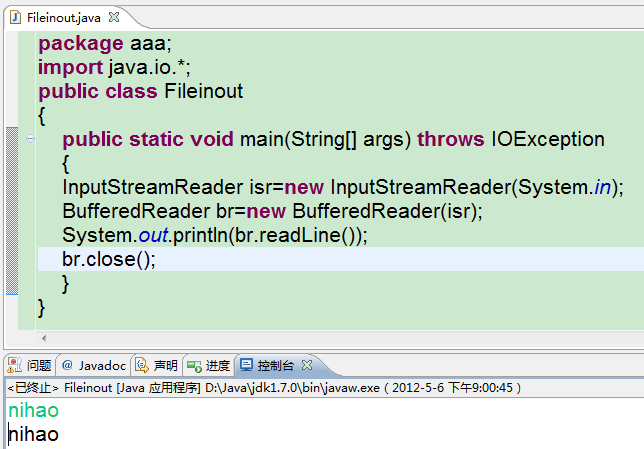


FileReader类和FileWriter类分别是InputStreamReader和OutputStreamWriter类的子类，他们提供将字符数据直接写入文件，或从文件中直接读出字符数据的简便方法。





从键盘输入数据：



其中：绿色的nihao是系统等待键盘输入，而黑色的nihao是输出的。

1. 打印输入流PrintWriter类和PrintStream类

PrintWriter类和PrintStream类都是打印输出流

PrintStream会调用println()方法，其输出包含换行符。但PrintWriter只有在调用Println方法时自动刷新，其实两者的方法很类似。

8. 对象序列化：ObjectInput接口与ObjectOutput接口分别继承了DateInput和DataOutput接口，提供用于读写基本数据和对象数据的方法。ObjectOutput提供了writeObject()方法，用于将对象写入流中。ObjectInput提供了readObject()方法，此方法用于将对象从流中读出。

**package** aaa;

**import** java.io.\*;

**public** **class** Fileinout

{

**public** **static** **void** main(String[] args) **throws** IOException

{

File f=**new** File("G://file.txt");

f.createNewFile();

FileOutputStream fos=**new** FileOutputStream(f);

ObjectOutputStream oos=**new** ObjectOutputStream(fos);

student stu=**new** student();

stu.name="zhulei";

stu.code="9876";

stu.address="上海市";

stu.age="22";

stu.sexy="male";

stu.school="tongji";

stu.grade="yanyi";

oos.writeObject(stu);

oos.close();

System.*out*.println("OK");

}

}

**class** student **implements** Serializable

{

String name;

String age;

String sexy;

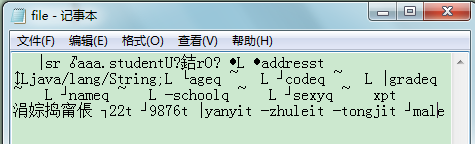
String school;

String grade;

String address;

String code;

}



读进来数据：

**package** aaa;

**import** java.io.\*;

**public** **class** Filein

{

**public** **static** **void** main(String[] args) **throws** IOException, ClassNotFoundException

{

File f=**new** File("G://file.txt");

FileInputStream fis=**new** FileInputStream(f);

ObjectInputStream ois=**new** ObjectInputStream(fis);

student st=**new** student();

st=(student)ois.readObject();

ois.close();

System.*out*.println(st.name);

System.*out*.println(st.address);

System.*out*.println(st.age);

System.*out*.println(st.school);

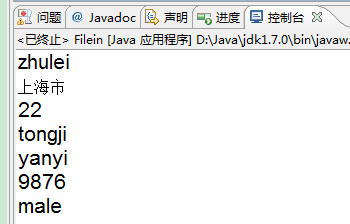
System.*out*.println(st.grade);

System.*out*.println(st.code);

System.*out*.println(st.sexy);

}

}



上面表明，进行了编码，虽然是乱的，但是读出来就是非常好的。