**FileOutputStream**

该类用来创建一个文件并向文件中写数据。

如果该流在打开文件进行输出前，目标文件不存在，那么该流会创建该文件。

有两个构造方法可以用来创建 FileOutputStream 对象。

使用字符串类型的文件名来创建一个输出流对象：

OutputStream f = new FileOutputStream("C:/java/hello")

也可以使用一个文件对象来创建一个输出流来写文件。我们首先得使用File()方法来创建一个文件对象：

File f = new File("C:/java/hello");

OutputStream fOut = new FileOutputStream(f);

创建OutputStream 对象完成后，就可以使用下面的方法来写入流或者进行其他的流操作。

序号 方法及描述

1 public void close() throws IOException{}

关闭此文件输入流并释放与此流有关的所有系统资源。抛出IOException异常。

2 protected void finalize()throws IOException {}

这个方法清除与该文件的连接。确保在不再引用文件输入流时调用其 close 方法。抛出IOException异常。

3 public void write(int w)throws IOException{}

这个方法把指定的字节写到输出流中。

4 public void write(byte[] w)

把指定数组中w.length长度的字节写到OutputStream中。

实例

下面是一个演示 InputStream 和 OutputStream 用法的例子：

fileStreamTest.java 文件代码：

import java.io.\*;

public class fileStreamTest {

public static void main(String[] args) {

try {

byte bWrite[] = { 11, 21, 3, 40, 5 };

OutputStream os = new FileOutputStream("test.txt");

for (int x = 0; x < bWrite.length; x++) {

os.write(bWrite[x]); // writes the bytes

}

os.close();

InputStream is = new FileInputStream("test.txt");

int size = is.available();

for (int i = 0; i < size; i++) {

System.out.print((char) is.read() + " ");

}

is.close();

} catch (IOException e) {

System.out.print("Exception");

}

}

}

上面的程序首先创建文件test.txt，并把给定的数字以二进制形式写进该文件，同时输出到控制台上。

以上代码由于是二进制写入，可能存在乱码，你可以使用以下代码实例来解决乱码问题：

fileStreamTest2.java 文件代码：

//文件名 :fileStreamTest2.java

import java.io.\*;

public class fileStreamTest2 {

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

File f = new File("a.txt");

FileOutputStream fop = new FileOutputStream(f);

// 构建FileOutputStream对象,文件不存在会自动新建

OutputStreamWriter writer = new OutputStreamWriter(fop, "UTF-8");

// 构建OutputStreamWriter对象,参数可以指定编码,默认为操作系统默认编码,windows上是gbk

writer.append("中文输入");

// 写入到缓冲区

writer.append("\r\n");

// 换行

writer.append("English");

// 刷新缓存冲,写入到文件,如果下面已经没有写入的内容了,直接close也会写入

writer.close();

// 关闭写入流,同时会把缓冲区内容写入文件,所以上面的注释掉

fop.close();

// 关闭输出流,释放系统资源

FileInputStream fip = new FileInputStream(f);

// 构建FileInputStream对象

InputStreamReader reader = new InputStreamReader(fip, "UTF-8");

// 构建InputStreamReader对象,编码与写入相同

StringBuffer sb = new StringBuffer();

while (reader.ready()) {

sb.append((char) reader.read());

// 转成char加到StringBuffer对象中

}

System.out.println(sb.toString());

reader.close();

// 关闭读取流

fip.close();

// 关闭输入流,释放系统资源

}

}