JAVA-IO详解

一． Input和Output

1. stream代表的是任何有能力产出数据的数据源，或是任何有能力接收数据的接收源。在Java的IO中，所有的stream（包括Input和Out stream）都包括两种类型：

1.1 以字节为导向的stream

以字节为导向的stream，表示以字节为单位从stream中读取或往stream中写入信息。以字节为导向的stream 包括下面几种类型：

1) input　stream：

1) ByteArrayInputStream：把内存中的一个缓冲区作为InputStream使用

2) StringBufferInputStream：把一个String对象作为InputStream

3) FileInputStream：把一个文件作为InputStream，实现对文件的读取操作

4) PipedInputStream：实现了pipe的概念，主要在线程中使用

5) SequenceInputStream：把多个InputStream合并为一个InputStream

2) Out　stream

1) ByteArrayOutputStream：把信息存入内存中的一个缓冲区中

2) FileOutputStream：把信息存入文件中

3) PipedOutputStream：实现了pipe的概念，主要在线程中使用

4) SequenceOutputStream：把多个OutStream合并为一个OutStream

1.2 以Unicode字符为导向的stream

以Unicode字符为导向的stream，表示以Unicode字符为单位从stream中读取或往stream中写入信息。以 Unicode字符为导向的stream包括下面几种类型：

1) Input　Stream

1) CharArrayReader：与ByteArrayInputStream对应

2) StringReader：与StringBufferInputStream对应

3) FileReader：与FileInputStream对应

4) PipedReader：与PipedInputStream对应

2) Out　Stream

1) CharArrayWrite：与ByteArrayOutputStream对应

2) StringWrite：无与之对应的以字节为导向的stream

3) FileWrite：与FileOutputStream对应

4) PipedWrite：与PipedOutputStream对应

以字符为导向的stream基本上对有与之相对应的以字节为导向的stream。两个对应类实现的功能相同，字是在操作时的导向不同。如CharArrayReader：和ByteArrayInputStream的作用都是把内存中的一个缓冲区作为InputStream使用，所不同的是前者每次从内存中读取一个字节的信息，而后者每次从内存中读取一个字符。

1.3 两种不现导向的stream之间的转换

InputStreamReader和OutputStreamReader：把一个以字节为导向的stream转换成一个以字符为导向的stream。

2. stream添加属性

2.1 “为stream添加属性”的作用

运用上面介绍的Java中操作IO的API，我们就可完成我们想完成的任何操作了。但通过FilterInputStream和FilterOutStream的子类，我们可以为stream添加属性。下面以一个例子来说明这种功能的作用。

如果我们要往一个文件中写入数据，我们可以这样操作：

FileOutStream fs = new FileOutStream(“test.txt”);

然后就可以通过产生的fs对象调用write()函数来往test.txt文件中写入数据了。但是，如果我们想实现“先把要写入文件的数据先缓存到内存中，再把缓存中的数据写入文件中”的功能时，上面的API就没有一个能满足我们的需求了。但是通过FilterInputStream和FilterOutStream的子类，为FileOutStream添加我们所需要的功能。

2.2 FilterInputStream的各种类型

\*\*\*

\*\*\*

2.2.1 用于封装以字节为导向的InputStream

1) DataInputStream：从stream中读取基本类型（int、char等）数据。

2) BufferedInputStream：使用缓冲区

3) LineNumberInputStream：会记录input stream内的行数，然后可以调用getLineNumber()和setLineNumber(int)

4) PushbackInputStream：很少用到，一般用于编译器开发

2.2.2 用于封装以字符为导向的InputStream

1) 没有与DataInputStream对应的类。除非在要使用readLine()时改用BufferedReader，否则使用DataInputStream

作者： 222.67.6.\* 2007-2-7 07:49 　 回复此发言

--------------------------------------------------------------------------------

2 java io

2) BufferedReader：与BufferedInputStream对应

3) LineNumberReader：与LineNumberInputStream对应

4) PushBackReader：与PushbackInputStream对应

2.3 FilterOutStream的各种类型

2.2.3 用于封装以字节为导向的OutputStream

1) DataIOutStream：往stream中输出基本类型（int、char等）数据。

2) BufferedOutStream：使用缓冲区

3) PrintStream：产生格式化输出

2.2.4 用于封装以字符为导向的OutputStream

1) BufferedWrite：与对应

2) PrintWrite：与对应

3. RandomAccessFile

1) 可通过RandomAccessFile对象完成对文件的读写操作

2) 在产生一个对象时，可指明要打开的文件的性质：r，只读；w，只写；rw可读写

3) 可以直接跳到文件中指定的位置

什么是IO

--------------------------------------------------------------------------------

　　IO（Input/Output）是计算机输出/输出的接口。Java的核心库java.io提供了全面的IO接口，包括：文件读写，标准设备输出等等。Java中IO是以流为基础进行输入输出的，所有数据被串行化写入输出流，或者从输入流读入。此外，Java也对块传输提供支持，在核心库java.nio中采用的便是块IO。关于NIO我们在后面还有专门的讨论。

　　流IO的好处是简单易用，缺点是效率较低。块IO效率很高，但编程比较复杂。

Java IO模型

--------------------------------------------------------------------------------

　　Java的IO模型设计非常优秀，它使用Decorator模式，按功能划分Stream，您可以动态装配这些Stream，以便获得您需要的功能。例如，您需要一个具有缓冲的文件输入流，则应当组合使用FileInputStream和BufferedInputStream。

　　Java的IO体系分Input/Output和Reader/Writer两类，区别在于Reader/Writer在读写文本时能自动转换内码。基本上，所有的IO类都是配对的，即有XxxInput就有一个对应的XxxOutput。

Java IO教程

--------------------------------------------------------------------------------

　　如果您对Decorator模式非常了解，就能轻易地看出Java的IO类结构：根接口是InputStream/OutputStream，充当数据源的IO类有FileInputStream/FileOutputStream，ByteArrayInputStream/ByteArrayOutputStream等，充当装饰功能的IO类有BufferedInputStream/BufferedOutputStream，DataInputStream/DataOutputStream等，它们都是继承装饰接口FilterInputStream/FilterOutputStream。使用IO时，首先创建一个数据源IO，然后根据需要的功能创建装饰类IO，其构造函数的参数为已创建的数据源IO。我们以创建一个具有缓冲的文件输入流为例，假定需要从磁盘读取文件“C:\log.txt”：

// 创建一个FileInputStream:

FileInputStream fileInput = new FileInputStream("C:\\log.txt");

// 创建一个BufferedInputStream:

BufferedInputStream bufferedInput = new BufferedInputStream(fileInput);

// 现在得到的bufferedInput即是具有缓冲的文件输入流

　　或者进一步简写如下：

InputStream input = new BufferedInputStream(

new FileInputStream("C:\\log.txt"));

// 现在得到的input即是具有缓冲的文件输入流

　　在您对Java的IO有了总体了解后，我们推荐您看看教程Introduction to Java I/O和I/O: Reading and Writing。

作者： 222.67.6.\* 2007-2-7 07:49 　 回复此发言

--------------------------------------------------------------------------------

3 回复：java io

java io 总结

java中的io系统总结.

java中的io中的（input/output）stream无非就是包括基于字符的stream、基于字节的stream和把字节导向的stream转换

字符为导向的stream的stream。（很难理解么？）

以字节为导向的stream------InputStream/OutputStream

InputStream 和 OutputStream是两个abstact类，对于字节为导向的stream都扩展这两个鸡肋（基类^\_^）;

--InputStream

ByteArrayInputStream -- 把内存中的一个缓冲区作为InputStream使用.

construct---ByteArrayInputStream(byte[])创建一个新字节数组输入流，它从指定字节数组中读取数据。

---ByteArrayInputStream(byte[], int, int) 创建一个新字节数组输入流，它从指定字节数组中读取数据。

---mark::该字节数组未被复制。

StringBufferInputStream -- 把一个String对象作为InputStream .

注释：不推荐使用 StringBufferInputStream 方法。 此类不能将字符正确的转换为字节。

同 JDK 1.1 版中的类似，从一个串创建一个流的最佳方法是采用 StringReader 类。

construct---StringBufferInputStream(String) 据指定串创建一个读取数据的输入流串。

FileInputStream -- 把一个文件作为InputStream，实现对文件的读取操作

construct---FileInputStream(File) 创建一个输入文件流，从指定的 File 对象读取数据。

---FileInputStream(FileDescriptor) 创建一个输入文件流，从指定的文件描述器读取数据。

---FileInputStream(String) 创建一个输入文件流，从指定名称的文件读取数据。

method ---- read() 从当前输入流中读取一字节数据。

read(byte[]) 将当前输入流中 b.length 个字节数据读到一个字节数组中。

read(byte[], int, int) 将输入流中 len 个字节数据读入一个字节数组中。

PipedInputStream：实现了pipe的概念，主要在线程中使用. 管道输入流是指一个通讯管道的接收端。

一个线程通过管道输出流发送数据，而另一个线程通过管道输入流读取数据，

这样可实现两个线程间的通讯。

PipedInputStream() 创建一个管道输入流，它还未与一个管道输出流连接。

PipedInputStream(PipedOutputStream) 创建一个管道输入流, 它已连接到一个管道输出流。

SequenceInputStream：把多个InputStream合并为一个InputStream .“序列输入流”类允许应用程序把几个输入流连续地合并起来，

并且使它们像单个输入流一样出现。每个输入流依次被读取，直到到达该流的末尾。

然后“序列输入流”类关闭这个流并自动地切换到下一个输入流。

SequenceInputStream(Enumeration) 创建一个新的序列输入流，并用指定的输入流的枚举值初始化它。

SequenceInputStream(InputStream, InputStream) 创建一个新的序列输入流，初始化为首先 读输入流 s1, 然后读输入流 s2。

--OutputSteam

ByteArrayOutputStream：把信息存入内存中的一个缓冲区中.该类实现一个以字节数组形式写入数据的输出流。

当数据写入缓冲区时，它自动扩大。用 toByteArray() 和 toString() 能检索数据。

construct --- ByteArrayOutputStream() 创建一个新的字节数组输出流。

--- ByteArrayOutputStream() 创建一个新的字节数组输出流。

--- ByteArrayOutputStream(int) 创建一个新的字节数组输出流，并带有指定大小字节的缓冲区容量。

toString(String) 根据指定字符编码将缓冲区内容转换为字符串，并将字节转换为字符。

write(byte[], int, int) 将指定字节数组中从偏移量 off 开始的 len 个字节写入该字节数组输出流。

write(int) 将指定字节写入该字节数组输出流。

writeTo(OutputStream) 用 out.write(buf, 0, count) 调用输出流的写方法将该字节数组输出流的全部内容写入指定的输出流参数。

FileOutputStream:文件输出流是向 File 或 FileDescriptor 输出数据的一个输出流。

作者： 222.67.6.\* 2007-2-8 07:40 　 回复此发言

--------------------------------------------------------------------------------

4 回复：java io

FileOutputStream(File) 创建一个文件输出流，向指定的 File 对象输出数据。

FileOutputStream(FileDescriptor) 创建一个文件输出流，向指定的文件描述器输出数据。

FileOutputStream(String) 创建一个文件输出流，向指定名称的文件输出数据。

FileOutputStream(String, boolean) 用指定系统的文件名，创建一个输出文件。

PipedOutputStream:管道输出流是指一个通讯管道的发送端。 一个线程通过管道输出流发送数据，

而另一个线程通过管道输入流读取数据，这样可实现两个线程间的通讯。

PipedOutputStream() 创建一个管道输出流，它还未与一个管道输入流连接。

PipedOutputStream(PipedInputStream) 创建一个管道输出流，它已连接到一个管道输入流。

以字符为导向的stream Reader/Writer

以Unicode字符为导向的stream，表示以Unicode字符为单位从stream中读取或往stream 中写入信息。

Reader/Writer 为abstact类

以Unicode字符为导向的stream包括下面几种类型：

－－ Reader

1) CharArrayReader：与ByteArrayInputStream对应

CharArrayReader(char[]) 用指定字符数组创建一个 CharArrayReader。

CharArrayReader(char[], int, int) 用指定字符数组创建一个 CharArrayReader。

2) StringReader：与StringBufferInputStream对应

StringReader(String) 创建一新的串读取者。

3) FileReader：与FileInputStream对应

4) PipedReader：与PipedInputStream对应

－－ Writer

1) CharArrayWrite：与ByteArrayOutputStream对应

2) StringWrite：无与之对应的以字节为导向的stream

3) FileWrite：与FileOutputStream对应

4) PipedWrite：与PipedOutputStream对应

两种不现导向的stream之间的转换

InputStreamReader和OutputStreamReader：把一个以字节为导向的stream转换成一个以字符为导向的stream。

一个 InputStreamReader 类是从字节流到字符流的桥梁：它读入字节，并根据指定的编码方式，将之转换为字符流。

使用的编码方式可能由名称指定，或平台可接受的缺省编码方式。

InputStreamReader 的 read() 方法之一的每次调用，可能促使从基本字节输入流中读取一个或多个字节。

为了达到更高效率，考虑用 BufferedReader 封装 InputStreamReader，

BufferedReader in = new BufferedReader(new InputStreamReader(System.in));

InputStreamReader(InputStream) 用缺省的字符编码方式，创建一个 InputStreamReader。

InputStreamReader(InputStream, String) 用已命名的字符编码方式，创建一个 InputStreamReader。

OutputStreamWriter 将多个字符写入到一个输出流，根据指定的字符编码将多个字符转换为字节。

每个 OutputStreamWriter 合并它自己的 CharToByteConverter, 因而是从字符流到字节流的桥梁。

FilterInputStream、RandomAccessFile 见例子。

ObjectInputStream 、 ObjectOutputStream见另外blog。

Java IO的一般使用原则：

一、按数据来源（去向）分类：

1、是文件： FileInputStream, FileOutputStream, FileReader, FileWriter

2、是byte[]：ByteArrayInputStream, ByteArrayOutputStream

3、是Char[]: CharArrayReader, CharArrayWriter

4、是String: StringBufferInputStream, StringReader, StringWriter

5、网络数据流：InputStream, OutputStream, Reader, Writer

二、按是否格式化输出分：

1、要格式化输出：PrintStream, PrintWriter

三、按是否要缓冲分：

1、要缓冲：BufferedInputStream, BufferedOutputStream, BufferedReader, BufferedWriter

四、按数据格式分：

1、二进制格式（只要不能确定是纯文本的）: InputStream, OutputStream及其所有带Stream结束的子类

2、纯文本格式（含纯英文与汉字或其他编码方式）；Reader, Writer及其所有带Reader, Writer的子类

五、按输入输出分：

1、输入：Reader, InputStream类型的子类

2、输出：Writer, OutputStream类型的子类

六、特殊需要：

1、从Stream到Reader,Writer的转换类：InputStreamReader, OutputStreamWriter

2、对象输入输出：ObjectInputStream, ObjectOutputStream

3、进程间通信：PipeInputStream, PipeOutputStream, PipeReader, PipeWriter

4、合并输入：SequenceInputStream

5、更特殊的需要：PushbackInputStream, PushbackReader, LineNumberInputStream, LineNumberReader

决定使用哪个类以及它的构造进程的一般准则如下（不考虑特殊需要）：

首先，考虑最原始的数据格式是什么： 原则四

第二，是输入还是输出：原则五

第三，是否需要转换流：原则六第1点

第四，数据来源（去向）是什么：原则一

第五，是否要缓冲：原则三 （特别注明：一定要注意的是readLine()是否有定义，有什么比read, write更特殊的输入或输出方法）

第六，是否要格式化输出：原则二

作者： 222.67.6.\* 2007-2-8 07:40 　 回复此发言

--------------------------------------------------------------------------------

5 回复：java io

使用Java 输出/输出流读写数据

Java 输入/输出（I/O）机制提供了一套简单的，标准化的API以便从不同的数据源读取和写入字符和字节数据。在“面向对象编程：Java collection更有效管理elements”一文中，我们讨论了Java 集合类架构中的类和功能并介绍了它的排序功能。在本文中，我们将学习Java 平台提供的这些I/O类，接口和操作。让我们先从了解Java 数据流开始。

数据流

Java所有的I/O机制都是基于数据流的，这些数据流表示了字符或者字节数据的流动序列。Java的I/O流提供了读写数据的标准方法。任何Java中表示数据源的对象都会提供以数据流的方式读写它的数据的方法。

Java.io是大多数面向数据流的输入/输出类的主要软件包。这个软件包包含了两个抽象类，InputStream和OutputStream。所有其它面象数据流的输入/输出类都要扩展这两个基类。

java.io软件包提供了一些类和接口，它们在由InputStream和OuputStream类提供的读写操作的顶端定义了一些有用的抽象。例如，ObjectInputStream类提供了让你把输入/输出流中的数据当成对象来读取的方法，而ObjectOutputStream类提供了让你能够把Java对象写入数据流中的方法。

优化读写过程

JDK 1.1 增加了一套读写类，它们提供了比现有数据流类更有用的抽象和更好的输入/输出性能。例如，BufferedReader和BufferedWriter 类被用来从基于字符的输入和输出流中读取和写入文本。BufferdReader 类缓存字符以更高效的读取字符串，数组和文本行。BufferedWriter类缓存字符以更高效的写入字符串，数组和文本行。BufferedReader和BufferedWriter 类可以按需求进行设置。

Java输入/输出架构提供的读取器和写入器类包括 LineNumberReader 类，CharArrayReader类，FileReader类，FilterReader类，PushbackReader类，PipedReader类，StringReader类以及其它一些类。这些类是在InputStream和OuputStream类顶部的包裹类因此提供了与InputStream和OuputStream类相似的方法。但是，这些类为读写特定的对象，比方文件，字符数组和字符串等等提供了更高效而有用的抽象。

读取数据

当你从一个相应的数据源对象里提取输入流或者是创建一个读取器对象的时候就会自动打开一个输入流。例如，要为一个文件打开输入流，我们只需要以下面的方式把文件名传递给Java.io.FileReader对象的构造函数：

java.io.FileReader fileReader = new java.io.FileReader("/home/me/myfile.txt");

要按顺序读取FileReader底层的输入流中的一个字节数据，只需要使用不带参数的read方法。表A中的代码段从一个文件读取文本数据，一次一个字符，然后把它写入System.out里。

要从输入流读取指定数目的字节数据到char数组里，只需要使用带一个char[]参数的read方法。数组的长度被用来确定应该读取的字符的个数。表B演示了这个技术。

要关闭一个输入流以及这个流使用的所有系统资源，你只需要以下面的方式调用close方法：

fileReader.close();

写入数据

象一个输入流一样，输出流通常在你从相应的数据源提取它或者是在你创建一个写入对象的时候被自动的打开。例如，要为一个文件打开输出流，我们把文件的名字传递给java.io.FileWriter对象的构造函数，如下所示：

java.io.FileWriter fileWriter = new

java.io.FileWriter("/home/me/out.txt");

要将一个特定的字符写入到输出流中，可以使用带一个int参数的write方法，int参数代表要定入的字符。

int aChar = (int)'X';

fileWriter.write(aChar);

要在输出流给定的偏移地址写入一个char数组中特定数目的字符，你可以使用带一个char[]参数，一个int 偏移量参数和一个int长度参数的write方法，如下面的例子所示：

fileWriter.write(buffer, 0, byteCount);

要关闭一个输出流并释放所有与之相关的系统资源，可以使用close方法，就象这样：

fileWriter.close();

要强迫写出一个输出流中的所有数据，可以使用下面的flush方法：

fileWriter.flush();

把它们全部综合起来

我们可以使用我们学习过的这些函数从一个文件中读取数据并同时写到另一个文件中去，如表C所示。

总结

Java的输入/输出机制为从不同的数据源读取和写入字符增加了一套简单而标准化的API。你对一种数据源使用Java流的经验能够让你容易的使用其它由Java提供的数据源类型。

在我们下一篇文章中，我们将会开始学习Java平台的联网和远程通讯架构。我们将会把我们对Java流的讨论扩展到这些环境并演示如何打开远程数据源，并象操作本地数据源，比方文件一样，写入数据和读取数据

作者： 222.67.6.\* 2007-2-8 07:42 　 回复此发言

--------------------------------------------------------------------------------

6 回复：java io

字符流的处理

　　java中提供了处理以16位的Unicode码表示的字符流的类，即以Reader和Writer 为基类派生出的一系列类。

Reader和Writer

　 这两个类是抽象类，只是提供了一系列用于字符流处理的接口，不能生成这两个类的实例，只能通过使用由它们派生出来的子类对象来处理字符流。

　1．Reader类是处理所有字符流输入类的父类。

　　◇ 读取字符

　　public int read() throws IOException; //读取一个字符，返回值为读取的字符

　　public int read(char cbuf[]) throws IOException; /\*读取一系列字符到数组cbuf[]中，返回值为实际读取的字符的数量\*/

　　public abstract int read(char cbuf[],int off,int len) throws IOException;

　　/\*读取len个字符，从数组cbuf[]的下标off处开始存放，返回值为实际读取的字符数量，该方法必须由子类实现\*/

　　◇ 标记流

　　public boolean markSupported(); //判断当前流是否支持做标记

　　public void mark(int readAheadLimit) throws IOException;

　　 //给当前流作标记，最多支持readAheadLimit个字符的回溯。

　　public void reset() throws IOException; //将当前流重置到做标记处

　　◇ 关闭流

　　public abstract void close() throws IOException;

　2． Writer类是处理所有字符流输出类的父类。

　　◇ 向输出流写入字符

　　public void write(int c) throws IOException；

　　//将整型值c的低16位写入输出流

　　public void write(char cbuf[]) throws IOException；

　　//将字符数组cbuf[]写入输出流

　　public abstract void write(char cbuf[],int off,int len) throws IOException；

　　//将字符数组cbuf[]中的从索引为off的位置处开始的len个字符写入输出流

　　public void write(String str) throws IOException；

　　//将字符串str中的字符写入输出流

　　public void write(String str,int off,int len) throws IOException；

　　//将字符串str 中从索引off开始处的len个字符写入输出流

　　◇ flush( )

　　刷空输出流，并输出所有被缓存的字节。

　　◇ 关闭流

　　public abstract void close() throws IOException

InputStreamReader和OutputStreamWriter

　　java.io包中用于处理字符流的最基本的类,用来在字节流和字符流之间作为中介。

　 ◇ 生成流对象

　　public InputStreamReader(InputStream in);

　　/\*in是字节流，而InputStreamReader是字符流，但是其来源是字节流in，

　　因此InputStreamReader就可以把字节流in转换成字符流处理。/\*

　　public InputStreamReader(InputStream in,String enc) throws UnsupportedEncodingException;

　　/\*enc是编码方式，就是从字节流到字符流进行转换时所采用的编码方式，

　　　例如 ISO8859-1，UTF-8，UTF-16等等\*/

　　public OutputStreamWriter(OutputStream out);

　　/\*out是字节流，而OutputStreamReader是字符流 \*/

　　public OutputStreamWriter(OutputStream out,String enc) throws UnsupportedEncodingException; //enc是编码方式

　　InputStreamReader和OutputStreamWriter的方法：

　　◇ 读入和写出字符

　　基本同Reader和Writer。

　　◇ 获取当前编码方式

　　public String getEncoding();

　　◇ 关闭流

　　public void close() throws IOException;

BufferedReader和BufferedWriter

　◇ 生成流对象

　　public BufferedReader(Reader in); //使用缺省的缓冲区大小

　　public BufferedReader(Reader in, int sz); //sz为缓冲区的大小

　　public BufferedWriter(Writer out);

　　public BufferedWriter(Writer out, int sz);

　◇ 读入/写出字符

　　除了Reader和Writer中提供的基本的读写方法外，增加对整行字符的处理。

　　public String readLine() throws IOException; //读一行字符

　　public void newLine() throws IOException; //写一行字符

import java.io.\*;

　　public class NumberInput{

　　　public static void main(String args[]){

　　　　try{

　　　　　　InputStreamReader ir;

　　　　　　BufferedReader in;

　　　　　　ir=new InputStreamReader(System.in);

　　　　　　//从键盘接收了一个字符串的输入，并创建了一个字符输入流的对象

　　　　　　in=new BufferedReader(ir);

　　　　　　String s=in.readLine();

　　　　　　//从输入流in中读入一行，并将读取的值赋值给字符串变量s

　　　　　　System.out.println("Input value is: "+s);

　　　　　　int i = Integer.parseInt(s);//转换成int型

　　　　　　i\*=2;

　　　　　　System.out.println("Input value changed after doubled: "+i);

　　　　}catch(IOException e)

　　　　{System.out.println(e);}

　　　}

　　}

　　　　运行结果

D:\>java NumberInput

123

Input value is 123

Input value changed after doubled： 246

　注意：在读取字符流时，如果不是来自于本地的，比如说来自于网络上某处的与本地编码方式不同的机器，那么我们在构造输入流时就不能简单地使用本地缺省的编码方式，否则读出的字符就不正确；为了正确地读出异种机上的字符，我们应该使用下述方式构造输入流对象：

　　　　ir = new InputStreamReader(is, "8859\_1");

　　采用ISO 8859\_1编码方式，这是一种映射到ASCII码的编码方式，可以在不同平台之间正确转换字符。

作者： 222.67.6.\* 2007-2-8 07:44 　 回复此发言

--------------------------------------------------------------------------------

7 回复：java io

JAVA IO

流一般分为输入流（Input Stream）和输出流（Output Stream）两类，但这种划分并不是绝对的。比如一个文件，当向其中写数据时，它就是一个输出流；当从其中读取数据时，它就是一个输入流.

字节流：

　　从InputStream和OutputStream派生出来的一系列类。这类流以字节(byte)为基本处理单位。

字符流：

　　从Reader和Writer派生出的一系列类，这类流以16位的Unicode码表示的字符为基本处理单位.

Reader类和Writer类

前面说过，在JDK1.1之前，java.io包中的流只有普通的字节流（以byte为基本处理单位的流），这种流对于以16位的Unicode码表示的字符流处理很不方便。从JDK1.1开始， java.io包中加入了专门用于字符流处理的类，它们是以Reader和Writer为基础派生的一系列类

同类InputStream和OutputStream一样，Reader和Writer也是抽象类，只提供了一系列用于字符流处理的接口。它们的方法与类InputStream和OutputStream类似，只不过其中的参数换成字符或字符数组

作者： 222.67.6.\* 2007-2-8 07:44 　 回复此发言

--------------------------------------------------------------------------------

8 回复：java io

java class diagram

贴子相关图片:

作者： 222.67.6.\* 2007-2-8 07:54 　 回复此发言

--------------------------------------------------------------------------------

9 回复：java io

java将读取数据对象成为输入流，能向其写入的对象叫输出流。结构图如下：输入流：

贴子相关图片:

作者： 222.67.6.\* 2007-2-8 07:55 　 回复此发言

--------------------------------------------------------------------------------

10 回复：java io

Reader的体系结构

贴子相关图片:

作者： 222.67.6.\* 2007-2-8 07:56 　 回复此发言

--------------------------------------------------------------------------------

11 回复：java io

Writer类体系结构

贴子相关图片:

作者： 222.67.6.\* 2007-2-8 07:56 　 回复此发言

--------------------------------------------------------------------------------

12 回复：java io

Introducing I/O

Input and output, I/O for short, are fundamental to any computer operating system or programming language. Only theorists find it interesting to write programs that don't require input or produce output. At the same time, I/O hardly qualifies as one of the more "thrilling" topics in computer science. It's something in the background, something you use every daybut for most developers, it's not a topic with much sex appeal.

But in fact, there are plenty of reasons Java programmers should find I/O interesting. Java includes a particularly rich set of I/O classes in the core API, mostly in the java.io and java.nio packages. These packages support several different styles of I/O. One distinction is between byte-oriented I/O, which is handled by input and output streams, and character-I/O, which is handled by readers and writers. Another distinction is between the old-style stream-based I/O and the new-style channel- and buffer-based I/O. These all have their place and are appropriate for different needs and use cases. None of them should be ignored.

Java's I/O libraries are designed in an abstract way that enables you to read from external data sources and write to external targets, regardless of the kind of thing you're writing to or reading from. You use the same methods to read from a file that you do to read from the console or from a network connection. You use the same methods to write to a file that you do to write to a byte array or a serial port device.

Reading and writing without caring where your data is coming from or where it's going is a very powerful abstraction. Among other things, this enables you to define I/O streams that automatically compress, encrypt, and filter from one data format to another. Once you have these tools, programs can send encrypted data or write zip files with almost no knowledge of what they're doing. Cryptography or compression can be isolated in a few lines of code that say, "Oh yes, make this a compressed, encrypted output stream."

In this book, I'll take a thorough look at all parts of Java's I/O facilities. This includes all the different kinds of streams you can use and the channels and buffers that offer high-performance, high-throughput, nonblocking operations on servers. We're also going to investigate Java's support for Unicode. We'll look at Java's powerful facilities for formatting I/O. Finally, we'll look at the various APIs Java provides for low-level I/O through various devices including serial ports, parallel ports, USB, Bluetooth, and other hardware you'll find in devices that don't necessarily look like a traditional desktop computer or server.

I won't go so far as to say, "If you've always found I/O boring, this is the book for you!" I will say that if you do find I/O uninteresting, you probably don't know as much about it as you should. I/O is the means for communication between software and the outside world. Java provides a powerful and flexible set of tools for doing this crucial part of the job. Having said that, let's start with the basics.

作者： 222.67.6.\* 2007-2-8 07:57 　 回复此发言

--------------------------------------------------------------------------------

13 回复：java io

java io 简介

stream代表的是任何有能力产出数据的数据源，或是任何有能力接收数据的接收源。在Java的IO中，所有的stream（包括Inputstream和Out stream）都包括两种类型：

(1)字节流

表示以字节为单位从stream中读取或往stream中写入信息，即io包中的inputstream类和outputstream类的派生类。通常用来读取二进制数据，如图象和声音。

（2）字符流

以Unicode字符为导向的stream，表示以Unicode字符为单位从stream中读取或往stream中写入信息。

区别：

Reader和Writer要解决的，最主要的问题就是国际化。原先的I/O类库只支持8位的字节流，因此不可能很好地处理16位的Unicode字符流。Unicode是国际化的字符集(更何况Java内置的char就是16位的Unicode字符)，这样加了Reader和Writer之后，所有的I/O就都支持Unicode了。此外新类库的性能也比旧的好。

但是，Read和Write并不是取代InputStream和OutputStream，有时，你还必须同时使用"基于byte的类"和"基于字符的类"。为此，它还提供了两个"适配器(adapter)"类。InputStreamReader负责将InputStream转化成Reader，而OutputStreamWriter则将OutputStream转化成Writer。

一．流的层次结构

定义：

（1）java将读取数据对象成为输入流，能向其写入的对象叫输出流。结构图如下：输入流：

输出流

二．InputStream类

inputstream类和outputstream类都为抽象类，不能创建对象，可以通过子类来实例化。

InputStream是输入字节数据用的类，所以InputStream类提供了3种重载的read方法.Inputstream类中的常用方法：

（1）public abstract int read( )：读取一个byte的数据，返回值是高位补0的int类型值。

（2）public int read(byte b[ ])：读取b.length个字节的数据放到b数组中。返回值是读取的字节数。该方法实际上是调用下一个方法实现的

（3）public int read(byte b[ ], int off, int len)：从输入流中最多读取len个字节的数据，存放到偏移量为off的b数组中。

（4） public int available( )：返回输入流中可以读取的字节数。注意：若输入阻塞，当前线程将被挂起，如果InputStream对象调用这个方法的话，它只会返回0，这个方法必须由继承InputStream类的子类对象调用才有用，

（5） public long skip(long n)：忽略输入流中的n个字节，返回值是实际忽略的字节数, 跳过一些字节来读取

（6） public int close( ) ：我们在使用完后，必须对我们打开的流进行关闭.

三．OutputStream类

OutputStream提供了3个write方法来做数据的输出，这个是和InputStream是相对应的。

1. public void write(byte b[ ])：将参数b中的字节写到输出流。

2. public void write(byte b[ ], int off, int len) ：将参数b的从偏移量off开始的len个字节写到输出流。

3. public abstract void write(int b) ：先将int转换为byte类型，把低字节写入到输出流中。

4. public void flush( ) : 将数据缓冲区中数据全部输出，并清空缓冲区。

5. public void close( ) : 关闭输出流并释放与流相关的系统资源。

注意：

1. 上述各方法都有可能引起异常。

2. InputStream和OutputStream都是抽象类，不能创建这种类型的对象。

四．FileInputStream类

FileInputStream类是InputStream类的子类，用来处理以文件作为数据输入源的数据流。使用方法：

方式1：

File fin=new File("d:/abc.txt");

FileInputStream in=new FileInputStream(fin);

方式2：

FileInputStream in=new

FileInputStream("d: /abc.txt");

方式3：

构造函数将 FileDescriptor()对象作为其参数。

FileDescriptor() fd=new FileDescriptor();

作者： 222.67.6.\* 2007-2-8 10:45 　 回复此发言

--------------------------------------------------------------------------------

14 回复：java io

FileInputStream f2=new FileInputStream(fd);

五．FileOutputStream类

FileOutputStream类用来处理以文件作为数据输出目的数据流；一个表示文件名的字符串，也可以是File或FileDescriptor对象。

创建一个文件流对象有两种方法：

方式1：

File f=new File("d:/abc.txt");

FileOutputStream out=new FileOutputStream (f);

方式2：

FileOutputStream out=new

FileOutputStream("d:/abc.txt");

方式3：构造函数将 FileDescriptor()对象作为其参数。

FileDescriptor() fd=new FileDescriptor();

FileOutputStream f2=new FileOutputStream(fd);

方式4：构造函数将文件名作为其第一参数，将布尔值作为第二参数。

FileOutputStream f=new FileOutputStream("d:/abc.txt",true);

注意：

（1）文件中写数据时，若文件已经存在，则覆盖存在的文件；（2）的读/写操作结束时，应调用close方法关闭流。

举例：2-1

六．File类

File类与InputStream / OutputStream类同属于一个包，它不允许访问文件内容。

File类主要用于命名文件、查询文件属性和处理文件目录。

举例：2-2

七．从一个流构造另一个流

java的流类提供了结构化方法，如，底层流和高层过滤流。

而高层流不是从输入设备读取，而是从其他流读取。同样高层输出流也不是写入输出设备，而是写入其他流。

使用"分层对象(layered objects)"，为单个对象动态地，透明地添加功能的做法，被称为Decorator Pattern。Decorator模式要求所有包覆在原始对象之外的对象，都必须具有与之完全相同的接口。这使得decorator的用法变得非常的透明--无论对象是否被decorate过，传给它的消息总是相同的。这也是Java I/O类库要有"filter(过滤器)"类的原因：抽象的"filter"类是所有decorator的基类。Decorator模式常用于如下的情形：如果用继承来解决各种需求的话，类的数量会多到不切实际的地步。Java的I/O类库需要提供很多功能的组合，于是decorator模式就有了用武之地。

为InputStream和OutputStream定义decorator类接口的类，分别是FilterInputStream和FilterOutputStream。

7.1 FilterInputStream

FilterInputStream的种类

类

功能

构造函数的参数

用法

DataInputStream

与DataOutputStream配合使用，这样你就能以一种"可携带的方式(portable fashion)"从流里读取primitives了（int，char，long等）

InputStream

包含了一整套读取primitive数据的接口。

BufferedInputStream

用这个类来解决"每次要用数据的时候都要进行物理读取"的问题。你的意思是"用缓冲区。"

InputStream，以及可选的缓冲区的容量

它本身并不提供接口，只是提供一个缓冲区。需要连到一个"有接口的对象(interface object)"。

LineNumberInputStream

跟踪输入流的行号；有getLineNumber( )和setLineNumber(int)方法

InputStream

只是加一个行号，所以还得连一个"有接口的对象"。

PushbackInputStream

有一个"弹压单字节"的缓冲区(has a one byte push-back buffer)，这样你就能把最后读到的那个字节再压回去了。

InputStream

主要用于编译器的扫描程序。可能是为支持Java的编译器而设计的。用的机会不多。

作者： 222.67.6.\* 2007-2-8 10:45 　 回复此发言

--------------------------------------------------------------------------------

15 回复：java io

FilterOutputStream的种类

类

功能

构造函数的参数

用法

DataOutputStream

与DataInputStream配合使用，这样你就可以用一种"可携带的方式(portable fashion)"往流里写primitive了(int, char, long,等)

OutputStream

包括写入primitive数据的全套接口。

PrintStream

负责生成带格式的输出(formatted output)。DataOutputStrem负责数据的存储，而PrintStream负责数据的显示。

一个OutputStream以及一个可选的boolean值。这个boolean值表示，要不要清空换行符后面的缓冲区。

应该是OutputStream对象的最终包覆层。用的机会很多。

BufferedOutputStream

用 这个类解决"每次往流里写数据，都要进行物理操作"的问题。也就是说"用缓冲区"。用flush( )清空缓冲区。

OutputStream， 以及一个可选的缓冲区大小

本身并不提供接口，只是加了一个缓冲区。需要链接一个有接口的对象。

DataInputStream类对象可以读取各种类型的数据。

DataOutputStream类对象可以写各种类型的数据；

创建这两类对象时，必须使新建立的对象指向构造函数中的参数对象。例如：

FileInputStream in=new FileInputStream("d:/abc.txt");

DataInputStream din=new DataInputStream(in);

7.2BufferInputStream和bufferOutputStream

允许程序在不降低系统性能的情况下一次一个字节的从流中读取数据。

BufferInputstream定义了两种构造函数

（1） BufferInputStream b= new BufferInputstream(in);

（2） BufferInputStream b=new BufferInputStream(in,size)

第二个参数表示指定缓冲器的大小。

同样BufferOutputStream也有两种构造函数。一次一个字节的向流中写数据。

7.3printstream

用于写入文本或基本类型

两种构造函数方法：

PrintStream ps=new PrintStream(out);

PrintStream ps=new PrintStream(out, autoflush)

第二个参数为布尔值，控制每次输出换行符时java是否刷新输出流。

八．字符流的读取和写入

java.io.Reader 和 java.io.InputStream 组成了 Java 输入类。Reader 用于读入16位字符，也就是 Unicode 编码的字符；而 InputStream 用于读入 ASCII 字符和二进制数据。

Reader的体系结构

（1） FileReader

FileReader主要用来读取字符文件，使用缺省的字符编码，有三种构造函数：

--将文件名作为字符串

FileReader f=new FileReader(“c:/temp.txt”);

--构造函数将File对象作为其参数。

File f=new file(“c:/temp.txt”);

FileReader f1=new FileReader(f);

--构造函数将FileDescriptor对象作为参数

FileDescriptor() fd=new FileDescriptor()

FileReader f2=new FileReader(fd);

(2) charArrayReader

将字符数组作为输入流,构造函数为：

public CharArrayReader(char[] ch);

(3) StringReader

读取字符串，构造函数如下：

public StringReader(String s);

(4) InputStreamReader

从输入流读取字节，在将它们转换成字符。

Public inputstreamReader(inputstream is);

作者： 222.67.6.\* 2007-2-8 10:45 　 回复此发言

--------------------------------------------------------------------------------

16 回复：java io

(5) FilterReader

允许过滤字符流

protected filterReader(Reader r);

(6) BufferReader

接受Reader对象作为参数，并对其添加字符缓冲器，使用readline()方法可以读取一行。

Public BufferReader(Reader r);

Writer类体系结构

（1） FileWrite

将字符类型数据写入文件，使用缺省字符编码和缓冲器大小。

Public FileWrite(file f);

(2)chararrayWrite()

将字符缓冲器用作输出。

Public CharArrayWrite();

(3) PrintWrite

生成格式化输出

public PrintWriter(outputstream os);

(4) filterWriter

用于写入过滤字符流

protected FilterWriter(Writer w);

FilterOutputStream的种类

类

功能

构造函数的参数

用法

DataOutputStream

与DataInputStream配合使用，这样你就可以用一种"可携带的方式(portable fashion)"往流里写primitive了(int, char, long,等)

OutputStream

包括写入primitive数据的全套接口。

PrintStream

负责生成带格式的输出(formatted output)。DataOutputStrem负责数据的存储，而PrintStream负责数据的显示。

一个OutputStream以及一个可选的boolean值。这个boolean值表示，要不要清空换行符后面的缓冲区。

应该是OutputStream对象的最终包覆层。用的机会很多。

BufferedOutputStream

用 这个类解决"每次往流里写数据，都要进行物理操作"的问题。也就是说"用缓冲区"。用flush( )清空缓冲区。

OutputStream， 以及一个可选的缓冲区大小

本身并不提供接口，只是加了一个缓冲区。需要链接一个有接口的对象。

DataInputStream类对象可以读取各种类型的数据。

DataOutputStream类对象可以写各种类型的数据；

创建这两类对象时，必须使新建立的对象指向构造函数中的参数对象。例如：

FileInputStream in=new FileInputStream("d:/abc.txt");

DataInputStream din=new DataInputStream(in);

7.2BufferInputStream和bufferOutputStream

允许程序在不降低系统性能的情况下一次一个字节的从流中读取数据。

BufferInputstream定义了两种构造函数

（1） BufferInputStream b= new BufferInputstream(in);

（2） BufferInputStream b=new BufferInputStream(in,size)

第二个参数表示指定缓冲器的大小。

同样BufferOutputStream也有两种构造函数。一次一个字节的向流中写数据。

7.3printstream

用于写入文本或基本类型

两种构造函数方法：

PrintStream ps=new PrintStream(out);

PrintStream ps=new PrintStream(out, autoflush)

第二个参数为布尔值，控制每次输出换行符时java是否刷新输出流。

八．字符流的读取和写入

java.io.Reader 和 java.io.InputStream 组成了 Java 输入类。Reader 用于读入16位字符，也就是 Unicode 编码的字符；而 InputStream 用于读入 ASCII 字符和二进制数据。

Reader的体系结构

（1） FileReader

FileReader主要用来读取字符文件，使用缺省的字符编码，有三种构造函数：

--将文件名作为字符串

FileReader f=new FileReader(“c:/temp.txt”);

--构造函数将File对象作为其参数。

File f=new file(“c:/temp.txt”);

FileReader f1=new FileReader(f);

--构造函数将FileDescriptor对象作为参数

FileDescriptor() fd=new FileDescriptor()

FileReader f2=new FileReader(fd);

(2) charArrayReader

将字符数组作为输入流,构造函数为：

public CharArrayReader(char[] ch);

(3) StringReader

读取字符串，构造函数如下：

public StringReader(String s);

(4) InputStreamReader

从输入流读取字节，在将它们转换成字符。

Public inputstreamReader(inputstream is);

(5) FilterReader

允许过滤字符流

protected filterReader(Reader r);

(6) BufferReader

接受Reader对象作为参数，并对其添加字符缓冲器，使用readline()方法可以读取一行。

Public BufferReader(Reader r);

Writer类体系结构

（1） FileWrite

将字符类型数据写入文件，使用缺省字符编码和缓冲器大小。

Public FileWrite(file f);

(2)chararrayWrite()

将字符缓冲器用作输出。

Public CharArrayWrite();

(3) PrintWrite

生成格式化输出

public PrintWriter(outputstream os);

(4) filterWriter

用于写入过滤字符流

protected FilterWriter(Writer w);

作者： 222.67.6.\* 2007-2-8 10:45 　 回复此发言

--------------------------------------------------------------------------------

17 java io simple code

13-1

import java.io.\*;

class FileCopy {

public static void main(String[] args) throw Exception {

File InputStream input = new FileInputStream("java13.txt");

//输入当前目录下的java13.txt文件

FileOutputStream output = new File OutputStream("java.bak");

//输出到当前目录下的java13.bak文件

int size = input.available(); //检查输入字节数

System.out.println("可输入字节数为：" + size);

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

output.write(input.read()); //拷贝到另一个文件

}

input.close();

output.close();

}

}

13-2

import java.io.\*;

class CopyFile {

public static void main(String[] args) throw Exception {

FileReader fr = new FileReader("java13.txt");

//输入当前目录下的java13.txt文件

FileWriter fw = new File FileWriter("java.bak");

//输出到当前目录下的java13.bak文件

int ch;

while((ch = fr.read())! = -1) {

fw.write(ch); //拷贝

}

fr.close();

fw.close();

}

}

13-3

import java.io.\*;

import java.util.\*;

class ByteArrayOutputStreamTest {

public static void main(String[] args) throw Exception {

ByteArrayOutputStream b = new ByteArrayOutputStream();

System.out.println("请输入七个字符，然后按回车")；

While(b.size() != 7) {

b.write(System.in.read()); //b.size()返回b里的存有的字节数

}

System.out.println(b.toString()); //把b变成字符串输出

byte arr[] = b.toByteArray(); //把b存入字节数组arr[]

for(int i = 0; i < arr.length; i++)

System.out.print((char)arr[i]);

System.out.println();

OutputStream f = new FileOutputStream("test.doc");

b.writeTo(f); //把b的内容拷贝到f里去

b.reset(); //reset()之后，再来输入就从b的开始写入

}

}

13-4

import java.io.\*;

public class ReaderWriter {

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

echo(new InputStreamReader(System.in),

new OutputStreamWriter(System.out));

}

public static void echo(InputStreamReader is, OutputStreamWriter os)

throws IOException {

try {

while(true) {

//在MS-DOS窗口中运行一个中文平台，输入中文

//CTRL-Z结束中文输入

int i = is.read(); //输入流结束，返回-1

if (i == -1) break;

char c = (char) i;

os.write©;

}

} catch(IOException e) {

System.err.println(e);

}

os.flush();

System.out.println("结束!");

}

}

13-5

import java.io.\*;

class FileTesting {

public static void main(String args[]) {

File f = new File("/temp/java12.txt");

if (f.exists() == true) {

System.out.println("这个文件存在。");

System.out.println("文件名是：" + f.getName());

System.out.println("文件路径是：" + f.getPath());

System.out.println("最后的修改日期是：" + f.lastModified());

System.out.println("文件长度是：" + f.length());

}

}

}

上一篇：JAVA规则 基本篇

下一篇：JAVA规则 中级篇 我也要收藏 [上一篇] [下一篇] (重金属男孩 的分类目录 [我的图书馆]) 相关文章java.io.Reader 和 java.io.InputStream的区别 - 周竞先...2008-09-05 luckystargq

彻底转变流，第 1 部分2006-08-09 starsiday

从Decorator，Adapter模式看Java/IO库（三）2008-09-02 木有银

从Decorator，Adapter模式看Java/IO库（一）2008-09-02 木有银

字节流与字符流2007-11-12 凤舞天煌

package 与 import2005-12-28 sowill

从Decorator，Adapter模式看Java/IO库（二）2008-09-02 木有银

在Java文件编码2008-10-01 凡尘一滴

查看更多文章>>

发表评论 游客请 登录 后再发表评论！您还没有 注册 ？30秒快速拥有您的“个人图书馆”!

发送评论时内容自动复制到剪切板