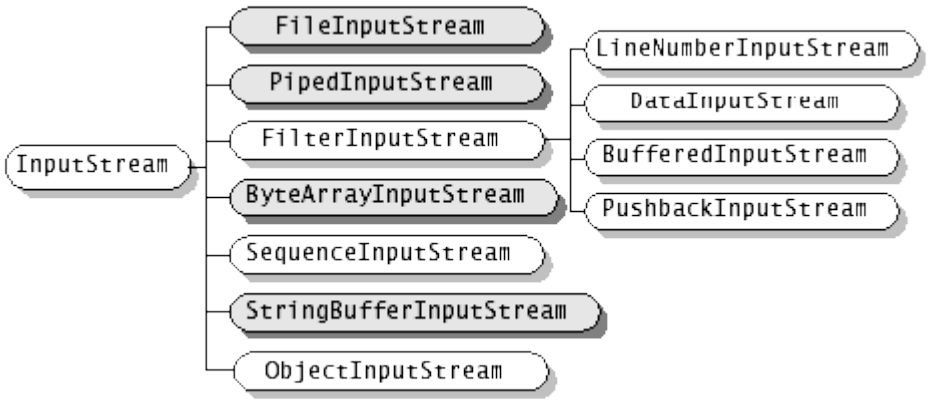
字符流与字节流的区别（转载）

Java 流在处理上分为字符流和字节流。字符流处理的单元为 2 个字节的 Unicode 字符，分别操作字符、字符数组或字符串，而字节流处理单元为 1 个字节，操作字节和字节数组。Java 内用 Unicode 编码存储字符，字符流处理类负责将外部的其他编码的字符流和 java 内 Unicode 字符流之间的转换。而类 InputStreamReader 和 OutputStreamWriter 处理字符流和字节流的转换。字符流（一次可以处理一个缓冲区）一次操作比字节流（一次一个字节）效率高。

(一)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 ^_^ ）；

1、 InputStream



1.1

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

construct---

(A)ByteArrayInputStream(byte[]) 创建一个新字节数组输入流（ ByteArrayInputStream ），它从指定字节数组中读取数据（ 使用 byte 作为其缓冲区数组）

(B)---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。

1.2

StringBufferInputStream -- 把一个 String 对象作为 InputStream .

construct---

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

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。

1.3

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

construct---

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

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

(C)-FileInputStream(String name) 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。

1.4

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。

construct---

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。

1.5

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。

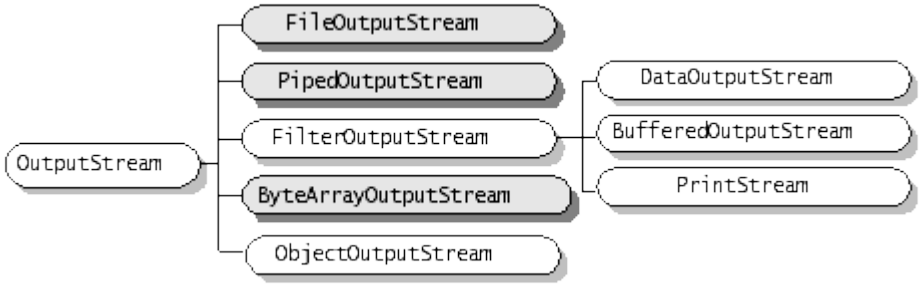
然后"序列输入流"类关闭这个流并自动地切换到下一个输入流。

construct---

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 。

2、 OutputSteam



2.1

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。

constructor

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

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

(C)--- 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。

2.2

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

constructor

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

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

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

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

2.3

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。

constructor

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

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

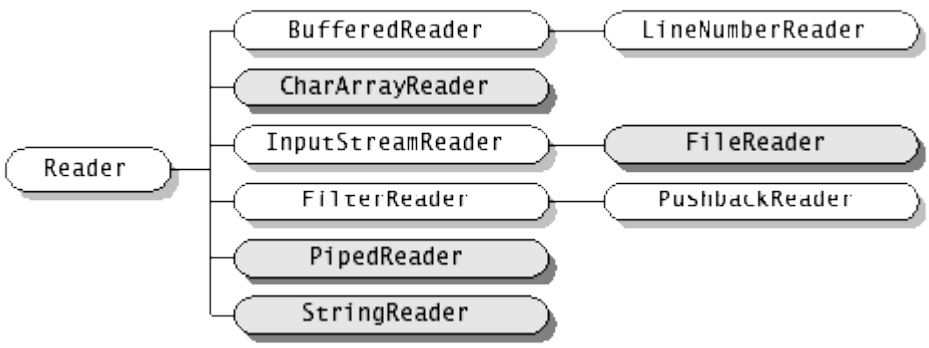
(二)以字符为导向的 stream Reader/Writer

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：

1. Reader



1.1

CharArrayReader ： 与 ByteArrayInputStream 对应此类实现一个可用作字符输入流的字符缓冲区

constructor

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

1.2

StringReader ： 与 StringBufferInputStream 对应其源为一个字符串的字符流。

StringReader(String) 创建一新的串读取者。

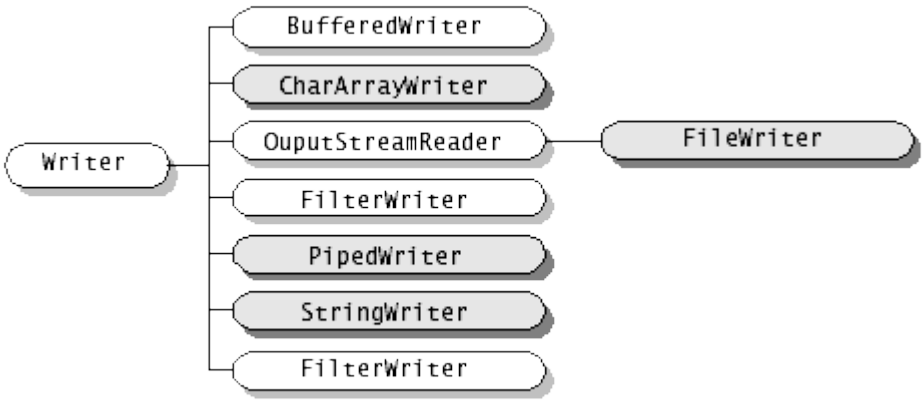
1.3

FileReader ： 与 FileInputStream 对应

1.4

PipedReader ： 与 PipedInputStream 对应

2. Writer



2.1 CharArrayWrite ： 与 ByteArrayOutputStream 对应

2.2 StringWrite ：无与之对应的以字节为导向的 stream

2.3 FileWrite ： 与 FileOutputStream 对应

2.4 PipedWrite ： 与 PipedOutputStream 对应

3、两种不同导向的 stream 之间的转换

3.1

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 。

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));

例如： // 实现从键盘输入一个整数

```
String s = null;
InputStreamReader re = new InputStreamReader(System.in);
                BufferedReader br = new BufferedReader(re);
               try {
                s = br.readLine();
                System.out.println("s= " + Integer.parseInt(s));
                br.close();
                }
               catch (IOException e)
                {
                e.printStackTrace();
                }
               catch (NumberFormatException e)// 当应用程序试图将字符串转换成一种数值类型，但该字符串不能转换为适当格式时，抛出该异常。
                {
                System.out.println(" 输入的不是数字 ");
                }
```

String s = null; InputStreamReader re = new InputStreamReader(System.in); BufferedReader br = new BufferedReader(re); try { s = br.readLine(); System.out.println("s= " + Integer.parseInt(s)); br.close(); } catch (IOException e) { e.printStackTrace(); } catch (NumberFormatException e)// 当应用程序试图将字符串转换成一种数值类型，但该字符串不能转换为适当格式时，抛出该异常。 { System.out.println(" 输入的不是数字 "); }

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。

（三）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, (字节流)FileReader, FileWriter(字符)

2 、是 byte[] ： ByteArrayInputStream, ByteArrayOutputStream(字节流)

3 、是 Char[]: CharArrayReader, CharArrayWriter(字符流)

4 、是 String: StringBufferInputStream, StringBufferOuputStream (字节流)StringReader, StringWriter(字符流)

5 、网络数据流： InputStream, OutputStream,(字节流) Reader, Writer(字符流)

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,(字节流) BufferedReader, BufferedWriter(字符流)

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