Scanner中hasNext()是针对扫描器而言的，执行一次就会扫描一下，并且一次只扫描一个分隔符之前的内容，扫描的内容String next()是分隔符之前的内容（不包括分隔符），扫描过后就会跳到下一个扫描位置，等待执行。（个人理解）

不管是硬盘上还是传输中，文件都是以字节存在的，字符只存在于内存中。

[ByteArrayInputStream](mk:@MSITStore:D:\Program%20Files\Java\JAVAAPI_CN\%5bJava参考文档%5d.JDK_API_1_6_zh_CN.CHM::/java/io/ByteArrayInputStream.html)构造函数 ，[ByteArrayInputStream](mk:@MSITStore:D:\Program%20Files\Java\JAVAAPI_CN\%5bJava参考文档%5d.JDK_API_1_6_zh_CN.CHM::/java/io/ByteArrayInputStream.html)相对于内存的读入，对程序来说就是输出。

|  |
| --- |
| [**ByteArrayInputStream**](mk:@MSITStore:D:\Program%20Files\Java\JAVAAPI_CN\%5bJava参考文档%5d.JDK_API_1_6_zh_CN.CHM::/java/io/ByteArrayInputStream.html#ByteArrayInputStream(byte[]))(byte[] buf)            创建一个 ByteArrayInputStream，使用 buf 作为其缓冲区数组。 |
| [**ByteArrayInputStream**](mk:@MSITStore:D:\Program%20Files\Java\JAVAAPI_CN\%5bJava参考文档%5d.JDK_API_1_6_zh_CN.CHM::/java/io/ByteArrayInputStream.html#ByteArrayInputStream(byte[], int, int))(byte[] buf, int offset, int length)            创建 ByteArrayInputStream，使用 buf 作为其缓冲区数组。 |

其中“buf”中可以认为已经包含了一个输入流输入的内容（InputStream输入的也是byte[]），可以将buf当做“文件”，只不过这个文件是块内存。其方法read()就是从这块内存中读取数据。

**ByteArrayOutputStream相对于内存的输出，对程序来说就是写入。**

此类实现了一个输出流，其中的数据被写入一个 byte 数组。缓冲区会随着数据的不断写入而自动增长。可使用 toByteArray() 和 toString() 获取数据。

ByteArrayOutputStream是用来缓存数据的（数据写入的目标（output stream原义）），向它的内部缓冲区写入数据，缓冲区自动增长，当写入完成时可以从中提取数据。由于这个原因，ByteArrayOutputStream常用于存储数据以用于一次写入。

（input不管在哪都是输入的意思，只不过针对的焦点不同，一般是针对程序，关于内存的是针对内存。）

BufferedReader用来一次性接收数据到缓存（不用知道其大小），如果是从键盘输入，则必须用这个类，因为可能出新中文乱码，一个中文字符占两个字节。而InputStreamReader是一次性接收到一个数组中，需要提前定义大小，所以从文件中读入也应该使用BufferedReader。而Scanner可以完成BufferedReader的功能，并且更强大，Scanner在java.util中，是一个可以使用正则表达式来解析基本类型和字符串的简单文本扫描器。

 iso8859-1

属于单字节编码，最多能表示的字符范围是0-255，应用于英文系列。比如，字母'a'的编码为0x61=97。

很明显，iso8859-1编码表示的字符范围很窄，无法表示中文字符。但是，由于是单字节编码，和计算机最基础的表示单位一致，所以很多时候，仍旧使用iso8859-1编码来表示。而且在很多协议上，默认使用该编码。比如，虽然"中文"两个字不存在iso8859-1编码，以gb2312编码为例，应该是"d6d0 cec4"两个字符，使用iso8859-1编码的时候则将它拆开为4个字节（遇到连续的这样四个字节即表示为“中文”）来表示："d6 d0 ce c4"（事实上，在进行存储的时候，也是以字节为单位处理的）。而如果是UTF编码，则是6个字节"e4 b8 ad e6 96 87"。很明显，这种表示方法还需要以另一种编码为基础。

## DataInputStream与DataOutputStream类

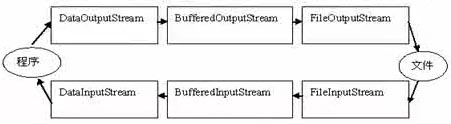
 (2005-12-27 11:07:15)

[[http://simg.sinajs.cn/blog7style/images/common/sg_trans.gif](javascript:;)转载▼](javascript:;)

|  |  |
| --- | --- |
|  | 分类： [JAVA](http://blog.sina.com.cn/s/articlelist_1449042313_5_1.html) |

DataInputStream与DataOutputStream类

* DataOutputStream类提供三个写入字符串的方法：  
     --  public final void writeBytes(String s)  //由于JAVA的字符编码是Unicode的，第个字符占两个字节，writeBytes方法只是将每个字符的低字节写入到目标设备中  
     --  public final void writeChars(String s)  //writeChars是将字符的两个字节都写入到目标设备中  
     --  public final void writeUTF(String str)  //writeUTF将字符串按照UTF编码写入到目标设备(其中包括长度)
* 为什么DataInputStream类中有一个readUTF方法，而没有"readBytes"和"readChars"方法呢？
* 编程实例：分别使用DataOutputStream类的writeUTF,writeBytes和writeChars方法,比较这几个方法的差异。程序中所使用的流栈如下：

[](http://blog.sina.com.cn/pic/565ea189a4d0df30329fb)

* 关闭流栈中的最上层的流对象（DataInputStream,DataOutputStream)，将会自动关闭流栈中的所有底层流对象。

//DataStreamTest.java

import java.io.\*;

public class DataStreamTest{

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

      FileOutputStream fos = new FileOutputStream("count.txt");

      BufferedOutputStream bos = new BufferedOutputStream(fos);

      DataOutputStream dos = new DataOutputStream(bos);

      dos.writeUTF("ab中国");

      dos.writeBytes("ab中国");

      dos.writeChars("ab中国");

      dos.close(); //只要关闭最上层流对象，下层的流对象也会关闭

      FileInputStream fis = new FileInputStream("count.txt");

      BufferedInputStream bis = new BufferedInputStream(fis);

      DataInputStream dis = new DataInputStream(bis);

      System.out.pritnln(dos.readUTF());

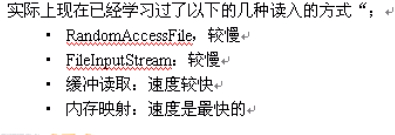
      dis.close();

   }

}

PrintStream类

* PrintStream类提供了一系列的print和println方法，可以将基本数据类型的数据格式化成字符串输出。
* 格式化输出是什么意思？例如：97被格式化输出的实际字节数据为0X39和0X37。
* PrintStream的3个构造函数:  
      --  PrintStream(OutputStream out)  
      --  PrintStream(OutputStream out,boolean autoflush)  
      --  PrintStream(OutputStream out,boolean autoflush,String encoding)
* 与PrintStream对应的PrintWriter类,即使遇到了文本换行标识符(\n),PrintWriter类也不会自动清空缓冲区。
* PrintWriter的println方法能根据操作系统的不同而生成相应的文本换行标识符。在Windows下的文本换行符是“\r\n“，而Linux下的文本换行标识符是“\n“。



但是内存映射最好只用在只读的时候，因为对内存的修改相当于对文件的修改

Java.nio中的Buffer是要结合Channel来完成读写操作。Nio主要使用来解决服务器端的通讯性能的。

使用Charset是为了用不同的编码系统对信息进行编码，而使用String.getBytes()，得到的是默认的编码。