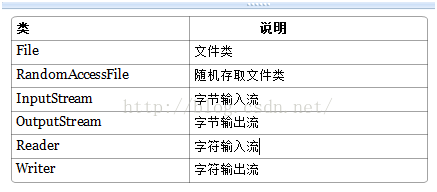
**JAVA IO**

IO中常用的六个类



**File**（文件特征与管理）：

用于文件或者目录的描述信息，例如生成新目录，修改文件名，删除文件，判断文件所在路径等。

**RandomAccessFile**（随机文件操作）：

一个独立的类，可以从文件的**任意位置**进行存取（输入输出）操作。

**InputStream**（二进制格式操作）：

抽象类，基于**字节**的输入操作。

**OutputStream**（二进制格式操作）：

抽象类，基于**字节**的输出操作。

**Reader**（文件格式操作）：

抽象类，基于**字符**的输入操作。

**Writer**（文件格式操作）：

抽象类，基于**字符**的输出操作。

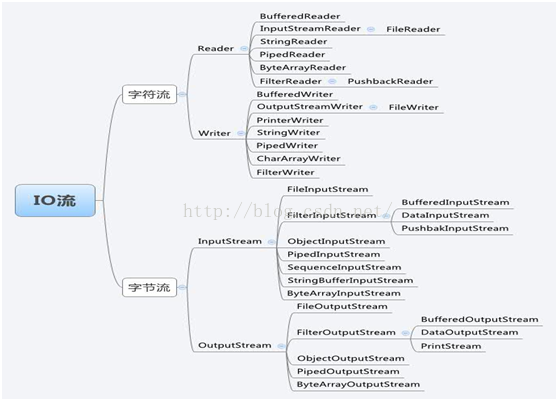
通过InputStreamReader和OutputStreamReader可以进行**字节**和**字符**的转换，设计Writer和Reader的目的是国际化，使IO操作支持16位的Unicode。



上图中的【其他】类指的是：

1、文件读取部分的与安全相关的类，如：SerializablePermission类，

2、与本地操作系统相关的文件系统的类，如：FileSystem类和Win32FileSystem类和WinNTFileSystem类。



流（Stream）是什么？

代表任何**有能力产出数据**的数据源对象或者是**有能力接受数据**的接收端对象，或者说是数据的序列。

本质：

数据传输，根据数据传输的特性将流（Stream）抽象为各种类。

作用：

为**数据源**和**目的地**建立一个输送通道。

流的分类：

根据**处理数据类型**的不同分为： 字符流和字节流

根据**数据流向**不同分为： 输入流和输出流

按数据来源（去向）分类：

1、File（文件）：

FileInputStream, FileOutputStream, FileReader, FileWriter

2、byte[]：

ByteArrayInputStream,ByteArrayOutputStream

3、Char[]:

CharArrayReader,CharArrayWriter

4、String:

StringBufferInputStream, StringReader, StringWriter

5、网络数据流：

InputStream,OutputStream,Reader, Writer

特性：

相对于程序来说，输出流是往存储介质或数据通道写入数据，而输入流是从存储介质或数据通道中读取数据，一般来说关于流的特性有下面几点：

1、先进先出。最先写入输出流的数据最先被输入流读取到。

2、顺序存取。可以一个接一个地往流中写入一串字节，读出时也将按写入顺序读取一串字节，不能随机访问中间的数据。（RandomAccessFile可以从文件的任意位置进行存取（输入输出）操作）

3、只读或只写。每个流只能是输入流或输出流的一种，不能同时具备两个功能，输入流只能进行读操作，输出流只能进行写操作。在一个数据传输通道中，如果既要写入数据，又要读取数据，则要分别提供两个流。

流结束的判断：

方法read()的返回值为-1时；readLine()的返回值为null时。

字符流与字节流转换（转换流的特点）：

1.是字符流和字节流之间的**桥梁**

2.可对读取到的字节数据经过指定编码转换成字符

InputStreamReader:**字节**到**字符**的桥梁，将字节流以字符流输出。

3.可对读取到的字符数据经过指定编码转换成字节

OutputStreamWriter:**字符**到**字节**的桥梁，将字节流以字符流输入。

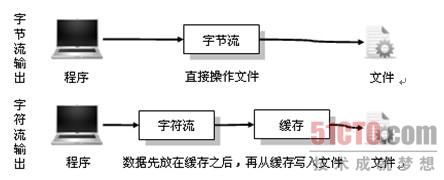
字节流和字符流的区别（重点）

字节流没有缓冲区，是直接输出的，而字符流是输出到**缓冲区**的。因此在输出时，字节流不调用colse()方法时，信息已经输出了，而字符流**只有**在调用close()方法关闭缓冲区时，信息才输出。要想字符流在未关闭时输出信息，则需要**手动**调用flush()方法。

1、**读写单位**不同：字节流以字节（8bit）为单位，字符流以字符为单位，根据码表映射字符，一次可能读多个字节。

2、**处理对象**不同：字节流能处理所有类型的数据（如图片、avi等），而字符流只能处理字符类型的数据。

结论：只要是处理纯文本数据，就优先考虑使用字符流。除此之外都使用字节流。



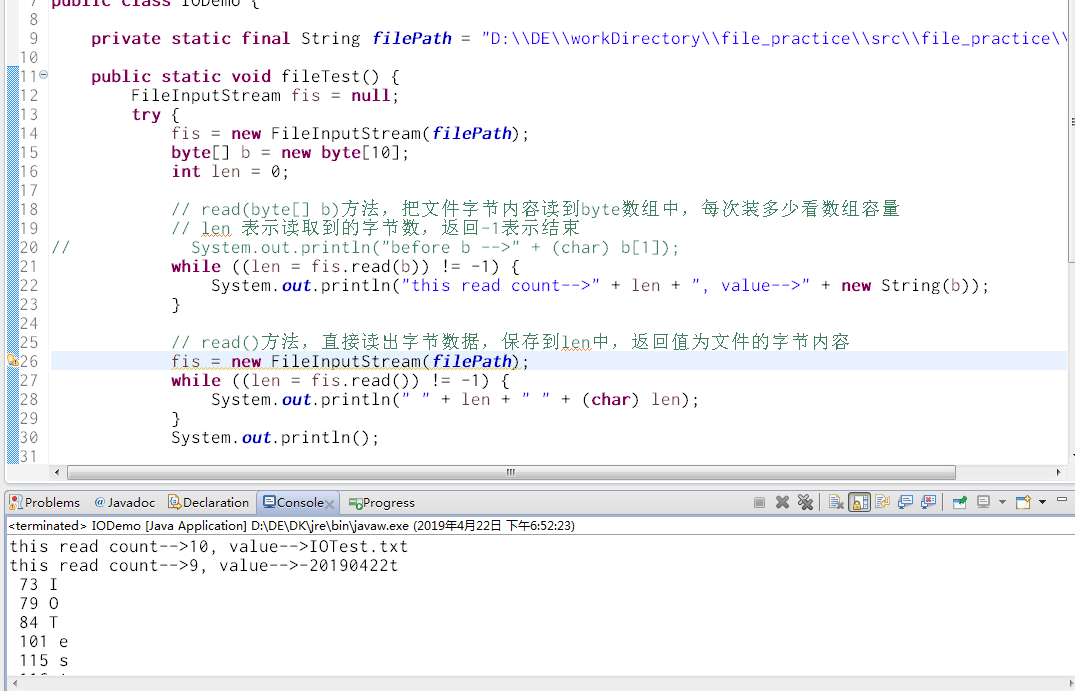
缓冲区：可以简单地把缓冲区理解为一段特殊的内存。某些情况下，如果一个程序频繁地操作一个资源（如文件或数据库），则性能会很低，此时为了提升性能，就可以将一部分数据暂时读入到内存的一块区域之中，以后直接从此区域中读取数据即可，因为读取内存速度会比较快，这样可以提升程序的性能。

# [深入理解Java中的IO](https://blog.csdn.net/qq_25184739/article/details/51205186)

# [Java 字节流与字符流的区别](https://blog.csdn.net/qq_25184739/article/details/51203733)

# [Java IO基础总结](https://www.cnblogs.com/dreamyu/p/6551137.html)

例子：读取本地一个文件里边的内容

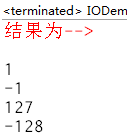
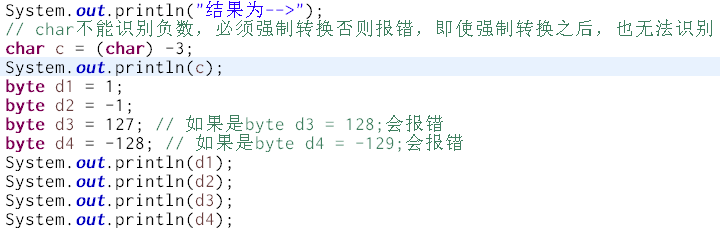


那么，为什么byte可以转成char类型呢？

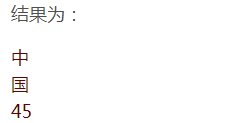
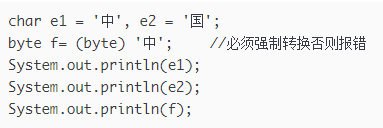
因为啊，byte 是字节数据类型 ，是**有符号型**的，占1个字节；大小范围为【-128 ~ 127】 。char 是字符数据类型，是**无符号型**（代表了它不能表示负数）的，占2字节（Unicode码），大小范围 是【0 ~ 65535】，char是一个16位二进制的Unicode字符，**JAVA用char来表示一个字符**。

区别：

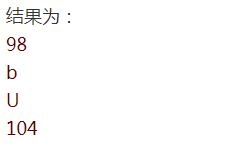
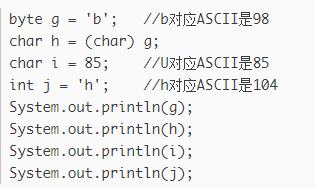
1、char是无符号型的，可以表示一个整数【0 ~ 65535】，不能表示负数；而byte是有符号型的，可以表示【-128 ~ 127】的数。



2、char可以表中文字符，byte不可以。

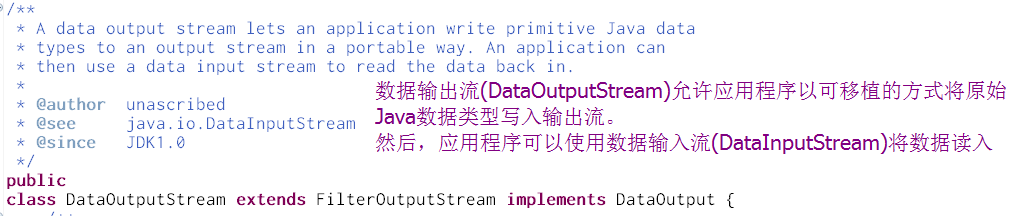


3、char、byte、int对于英文字符，可以相互转化。



# [char与byte的区别](https://blog.csdn.net/luoweifu/article/details/7770588)

DataOutputStream类和DataInputStream类



也就是说，使用这个类来进行流（Stream）的读写可以跨平台，不管用户是否在Linux、Windows或者Mac系统下。

套路就是，数据源 -> 目的地

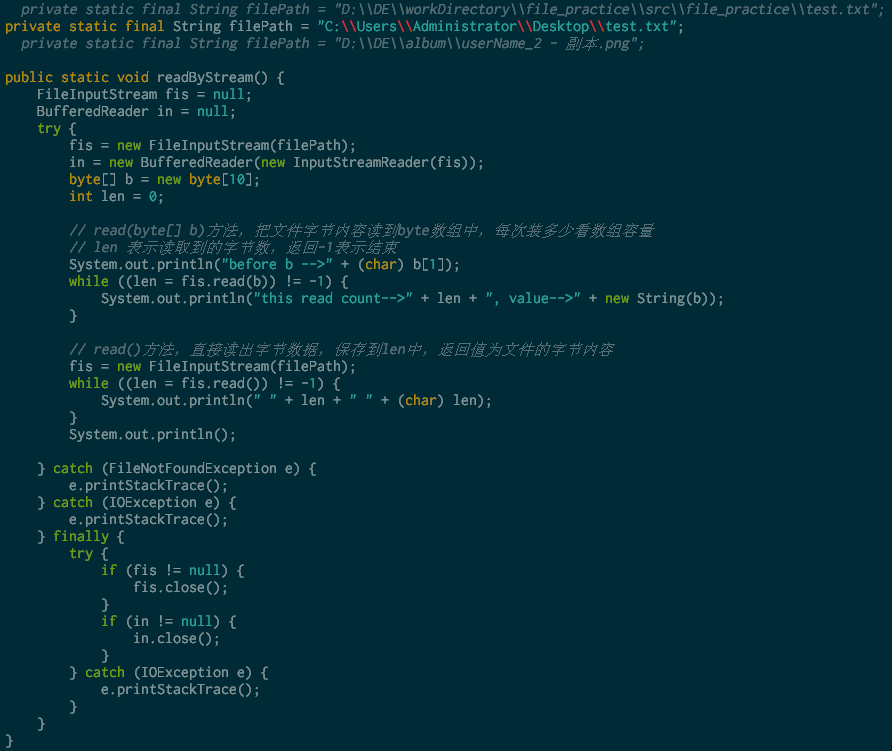
因为，流其实就是一个输送通道，什么输送通道，一个数据源和目的地的输送通道。

相对于输入流，那么基本的数据源要存在；

相对于输出流，那么基本的目的地要存在。

例子：

“字节流”读取文件



“字符流”读取文件



“字节流”写文件



“字符流”写文件

