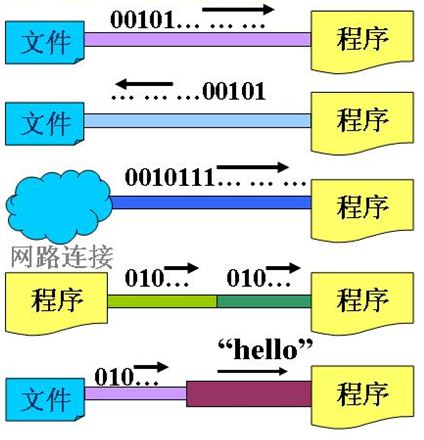
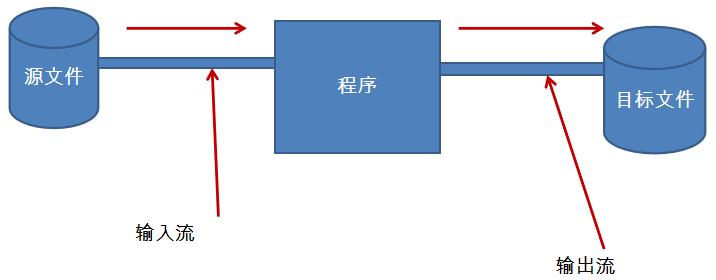
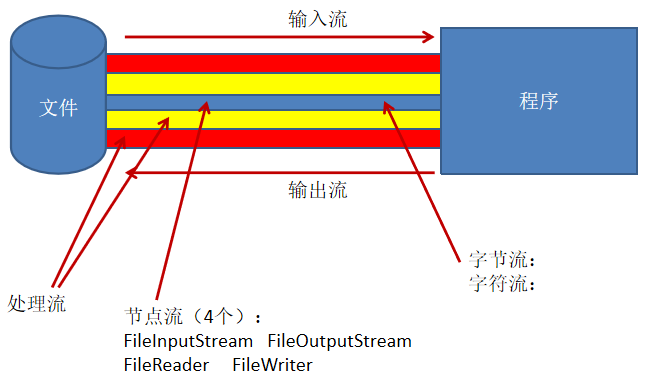
* **Java IO原理**
* **IO流用来处理设备之间的数据传输**。
* Java程序中，对于数据的输入/输出操作以”流(stream)”的方式进行。
* java.io包下提供了各种“流”类和接口，用以获取不同种类的数据，并通过标准的方法输入或输出数据。
* **输入input**：**读取外部数据（磁盘、光盘等存储设备的数据）到程序（内存）中**。
* **输出output**：**将程序（内存）数据输出到磁盘、光盘等存储设备中**。





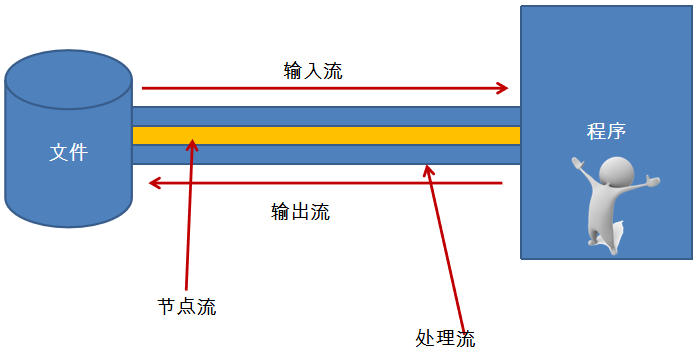
* **流的分类**
* 流的分类结构图：

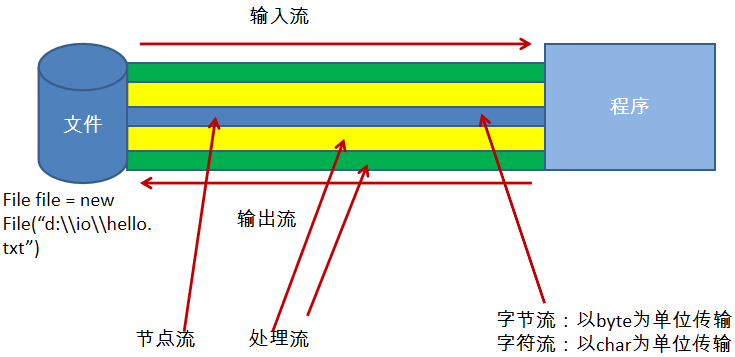


* 按操作**数据单位**不同分为：字节流(8 bit)，字符流(16 bit)。
* 按数据流的**流向**不同分为：输入流，输出流。
* 按流的**角色**的不同分为：节点流，处理流。

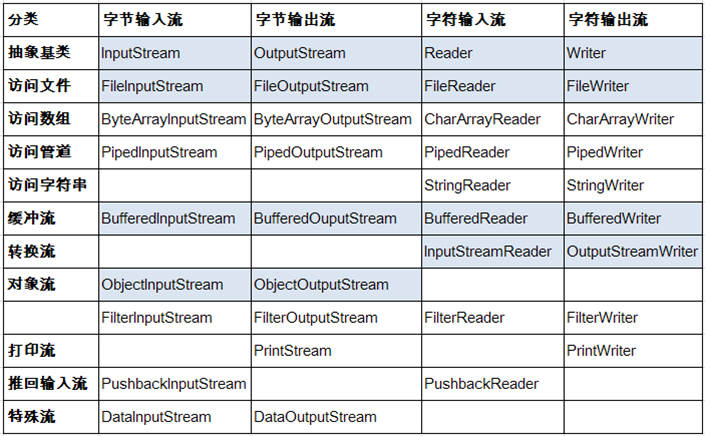
|  |  |  |
| --- | --- | --- |
| **(抽象基类)** | 字节流 | 字符流 |
| 输入流 | InputStream | Reader |
| 输出流 | OutputStream | Writer |

* Java的IO流共涉及40多个类，实际上非常规则，都是从如下4个抽象基类派生的。
* 由这四个类派生出来的子类名称都是以其父类名作为子类名后缀。





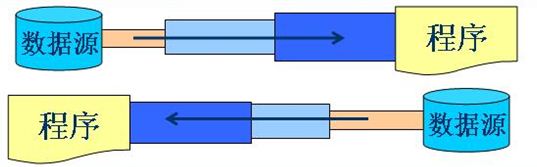
* **IO流体系**

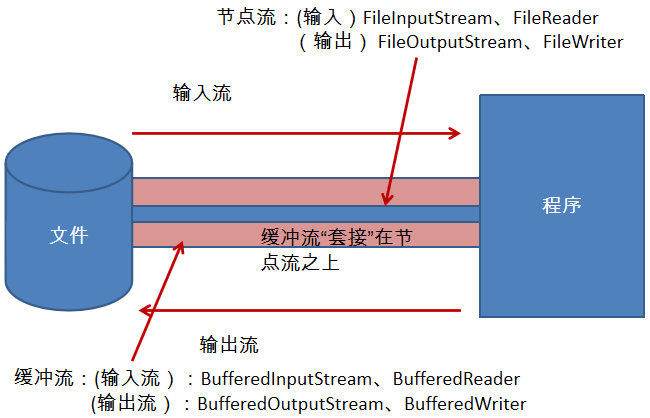


* **节点流和处理流**
* **节点流可以从一个特定的数据源读写数据**。



* 处理流是“连接”**在已存在的流（节点流或处理流）之上**，**通过对数据的处理为程序提供更为强大的读写功能**。





* **InputStream & Reader**
* **InputStream**和**Reader**是**所有输入流的基类**。
* InputStream（典型实现：**FileInputStream**）
  + int read()
  + int read(byte[]b)
  + int read(byte[]b,int off,int len)
* Reader（典型实现：**FileReader**）
  + int read()
  + **int read(char[]c)**
  + int read(char[]c,int off,int len)
* **程序中打开的文件IO资源不属于内存里的资源，垃圾回收机制无法回收该资源**，所以应该**显式关闭文件IO资源**。
* **OutputStream & Writer**
* OutputStream和Writer也非常相似：
  + void write(int b/int c);
  + void write(byte[]b/char[]cbuf);
  + void write(byte[]b/char[]buff,int off,int len);
  + void flush();
  + void close(); 需要先刷新，再关闭此流
* 因为字符流直接以字符作为操作单位，所以Writer可以用字符串来替换字符数组，即以String对象作为参数。
  + void write(String str);
  + void write(String str,int off,int len);