002.理解Java IO流

Java IO流的概念

大多数应用程序都需要实现与设备之间的数据传输，例如键盘可以输入数据，显示器可以显示程序的运行结果等。

在Java中，将这种通过不同输入输出设备（键盘，内存，显示器，网络等）之间的数据传输抽象的表述为“流”。

Java中的“流”都位于java.io包中，称之为IO（输入输出）流。

输入流和输出流是相对于内存设备而言的，将外设中的数据读取到内存中即输入，将内存的数据写入到外设中即输出。

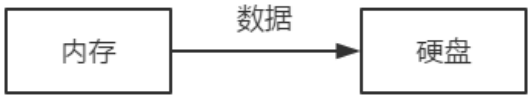
IO流的分类：

按照不同的分类方式，可以把流分为不同的类型。常用的分类有三种：

按照流的流向分，可以分为输入流和输出流。

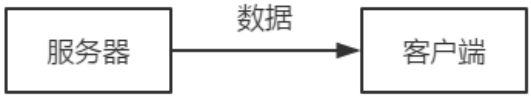
- 输入流： 只能从中读取数据，而不能向其写入数据。
- 输出流：只能向其写入数据，而不能向其读取数据。

此处的输入,输出涉及一个方向的问题，数据从内存到硬盘，通常称为输出流



数据从服务器通过网络流向客户端，在这种情况下,Server端的内存负责将数据输出到网络里，

因此Server端的程序使用输出流；



Client端的内存负责从网络中读取数据，因此Client端的程序应该使用输入流。

注：java的输入流主要是InputStream和Reader作为基类，

而输出流则是主要由outputStream和Writer作为基类。

它们都是一些抽象基类，无法直接创建实例。

按照操作单元划分，可以划分为字节流和字符流。

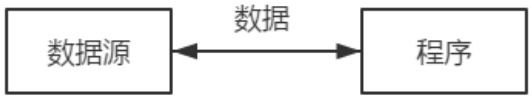
字节流和字符流的用法几乎完成全一样，区别在于字节流和字符流所操作的数据单元不同，字节流操作的单元是数据单元是8位的字节，字符流操作的是数据单元为16位的字符。

字节流主要是由InputStream和outPutStream作为基类，而字符流则主要有Reader和

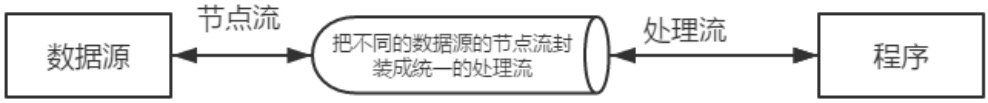
Writer作为基类。

按照流的角色划分为节点流和处理流。

可以从/向一个特定的IO设备（如磁盘，网络）读/写数据的流，称为节点流。**节点流也被称为低级流。**



处理流则用于对一个已存在的流进行连接和封装，通过封装后的流来实现数据的读/写功能。**处理流也被称为高级流。**



当使用处理流进行输入/输出时，程序并不会直接连接到实际的数据源，没有和实际的输入和输出节点连接。

使用处理流的一个明显的好处是，只要使用相同的处理流，程序就可以采用完全相同的输入/输出代码来访问不同的数据源

打个比方在进一步理解流的概念

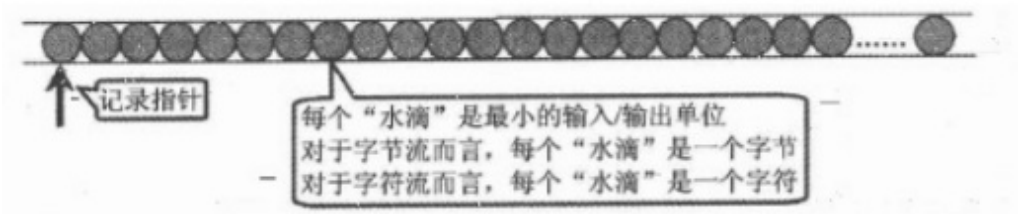
java把所有设备里的有序数据抽象成流模型，简化了输入/输出处理。

java IO流共涉及40多个类，这些类看上去很杂乱，但实际上很有规则，

而且彼此之间存在非常紧密的联系， Java Io流的40多个类都是从如下4个抽象类基类中派生出来的。

- **InputStream/Reader：所有的输入流的基类，前者是字节输入流，后者是字符输入流。**
- **OutputStream/Writer：所有输出流的基类，前者是字节输出流，后者是字符输出流。**

对于InputStream和Reader而言，它们把输入设备抽象成为一个”水管“，这个水管的每个“水滴”依次排列，如图

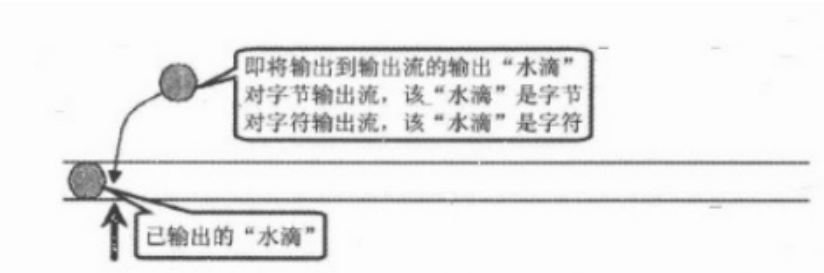


输入流使用隐式的记录指针来表示当前正准备从哪个“水滴”开始读取，

每当程序从InputStream或者Reader里面取出一个或者多个“水滴”后，

记录指针自定向后移动；除此之外，InputStream和Reader里面都提供了一些方法来控制记录指针的移动。

对于OutputStream和Writer而言，它们同样把输出设备抽象成一个”水管“，只是这个水管里面没有任何水滴，如图



当执行输出时，程序相当于依次把“水滴”放入到输出流的水管中，
输出流同样采用隐示指针来标识当前水滴即将放入的位置，
每当程序向OutputStream或者Writer里面输出一个或者多个水滴后，记录指针自动向后移动。

处理流可以“嫁接”在任何已存在的流的基础之上，这就允许Java应用程序采用相同的代码，
透明的方式来访问不同的输入和输出设备的数据流。