## 1.6 IO

### 1.6.1概念

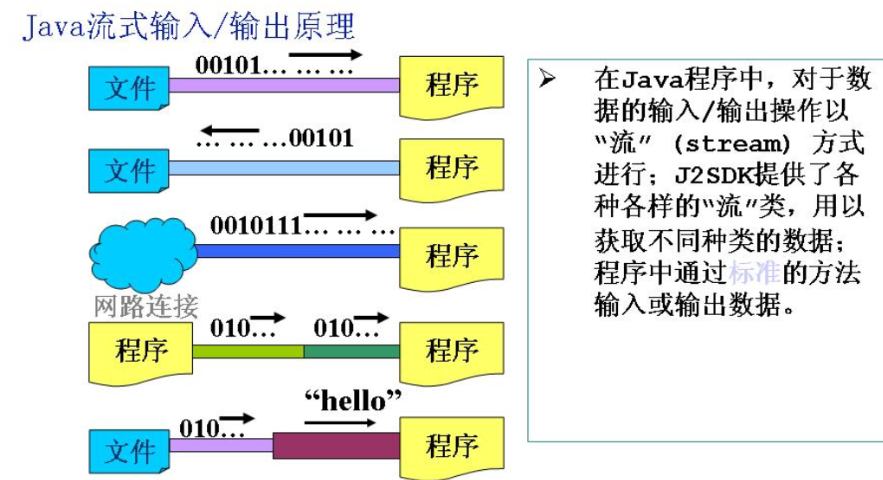
流 水流的流 流氓的流，，流 英文叫做stream，溪流

### 1.6.2 分类



### 1.6.3 Java流输入输出原理

流是用来读写数据的。



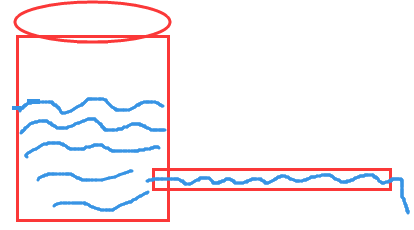
流就像水流一样，File类封装的是文件的名字，它是内存里头的一个对象，OK，但是真正的文件是在硬盘上的一块空间，在这文件里头，存着各种各样的数据，那么我们想读这数据怎么办呢？是这样来读，是通过一个流的方式来读。

**从文件读数据：**

如下图，比方说这是我们文件，咱们要想从程序读数据，大家都知道，对于计算机来说，你读任何的数据都是什么样的形式，是不是都是010101 这些形式，对吧 。



怎么样吧这些文件里边的内容个读出来，你就把这个文件想象成一个小桶。文件就是个桶，你怎么从桶里边取水呢？或者说怎么从文件里边取数据呢，常见的取法是这样的，我们弄一根管道怼到桶上，然后把这根管道，哐唧，一怼，怼到桶上边，这边打开水龙头，就开始出水，这就叫流，什么叫流，这根管道就叫流，水流哗啦哗啦哗啦流出来了，这就叫流，java里头的输入输出，跟这个水流一模一样，当你要从文件取数据的时候，一根管道咣唧怼上去，然后就顺着这根管道，你开始往外读数据。



**往文件写数据：**

反过来，咣唧一根管道怼上去，从管道这一头开始咵唧咵唧往里头灌水。

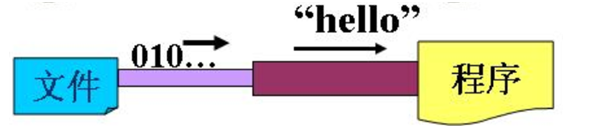


**从网络上读数据：**



比如这根管道咵唧从我机器上怼到你机器上了，我说一句话，你那边就收着了。你说一句话我这边就收着了

**处理流**，包装在其他流之上的流：



 有的时候一根管道不够使，比方说这根管道读过来的水，里有杂质，外边再包一层管道， 比方说你读过来的数据，最原始的数据肯定是010101，一个字节一个字节的往外读，当你这样读的时候你觉得这样的方法不合适，没关系。外边再包一层强大一点的管道，这个管道可以吧010101转换成字符串，哐当读出来，这个管道就包在了原始管道的外边， 这就是流式的输入输出，这些管道就是java的流。

### 1.6.4 流的分类

根据不同的角度分成不同的类型

输入流or输出流的划分：都是站在程序打的角度来说的，相对于程序来说的。

#### 字节流or字符流：

字节流：最原始的一个流， 读出来的数据就是010101，只不过是按照字节来读的，一个字节是8位。

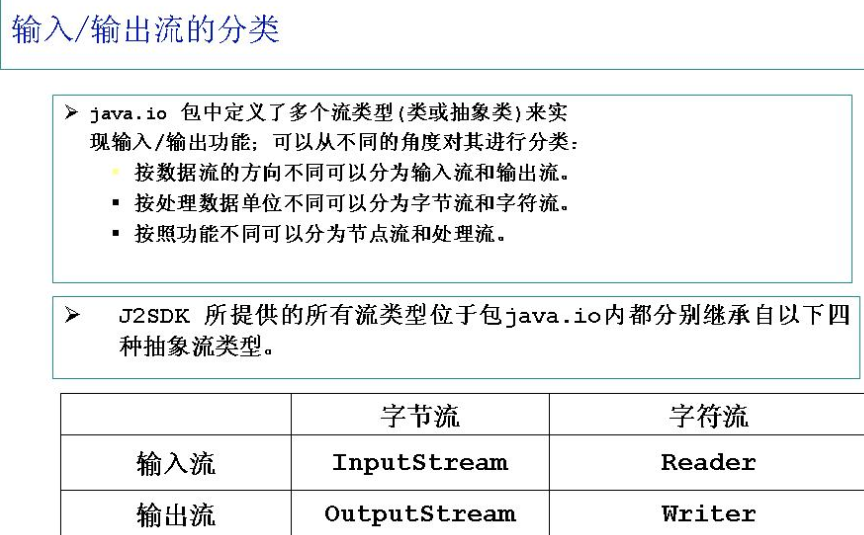
字符流：一个字符一个字符往外读，字符是2个字节。Java里边是Unicode字符串一个字符是2个字节。 Utf16  ok

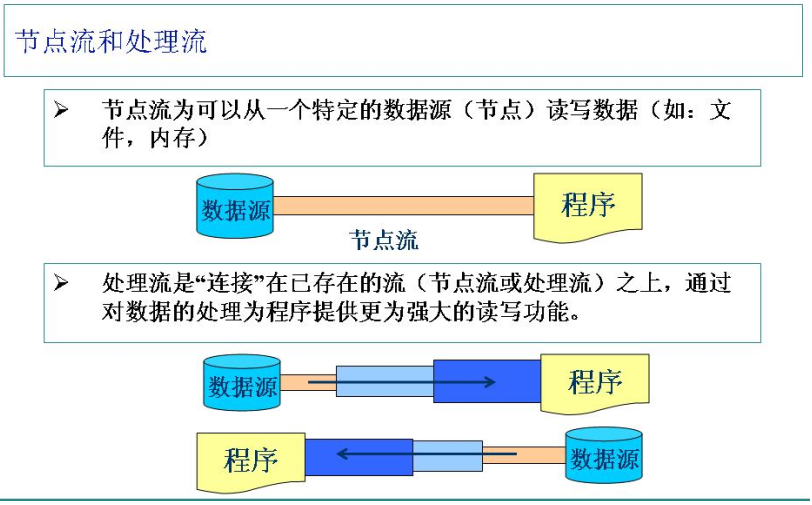
#### 节点流or处理流：

节点流：这根管道直接怼到数据源上的流。

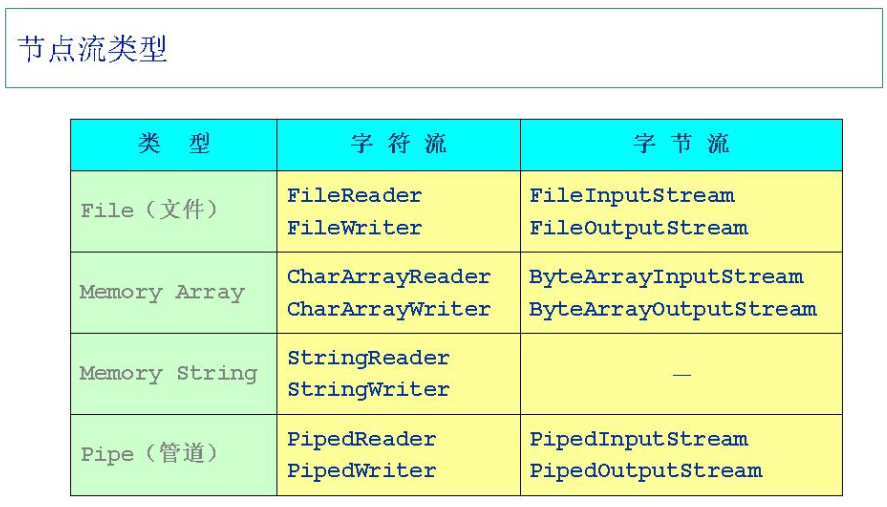
处理流：套在其他管道之上的流。

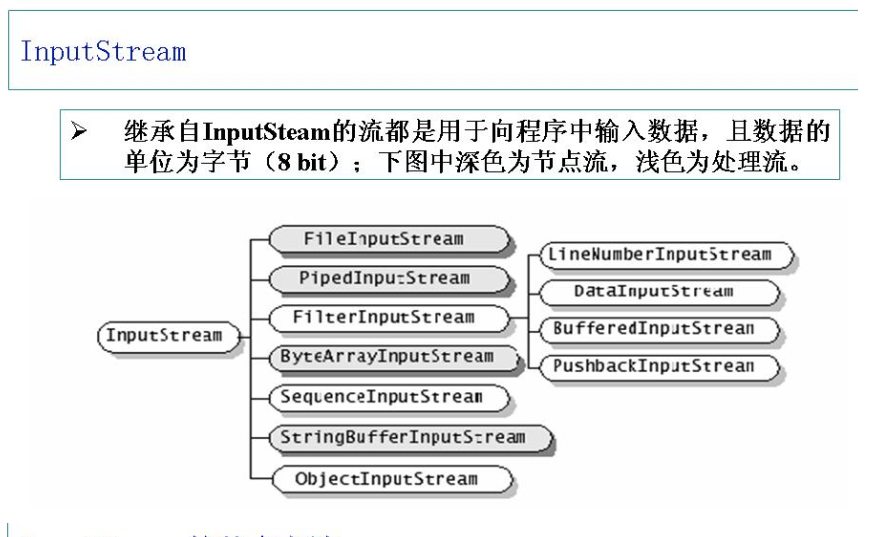
凡是以Stream结尾的都是字节流，最原始的读取数据的方式，一个字节一个字节的读。

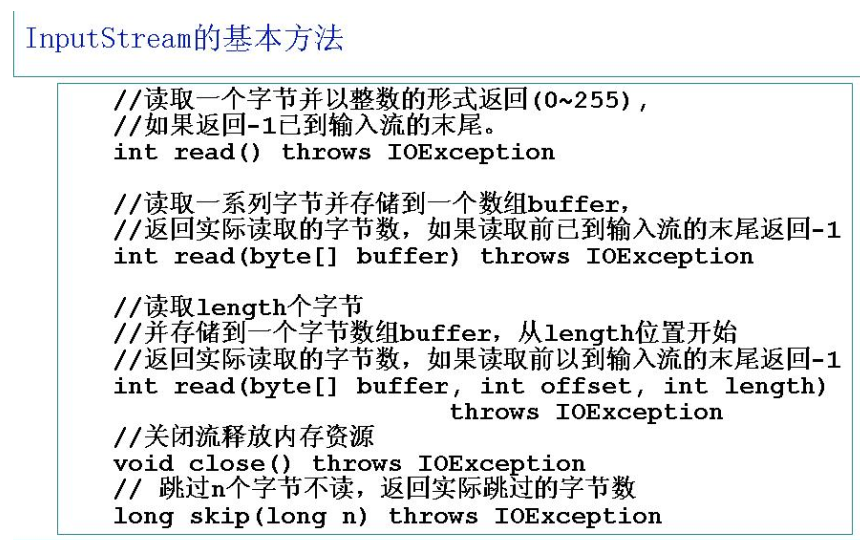




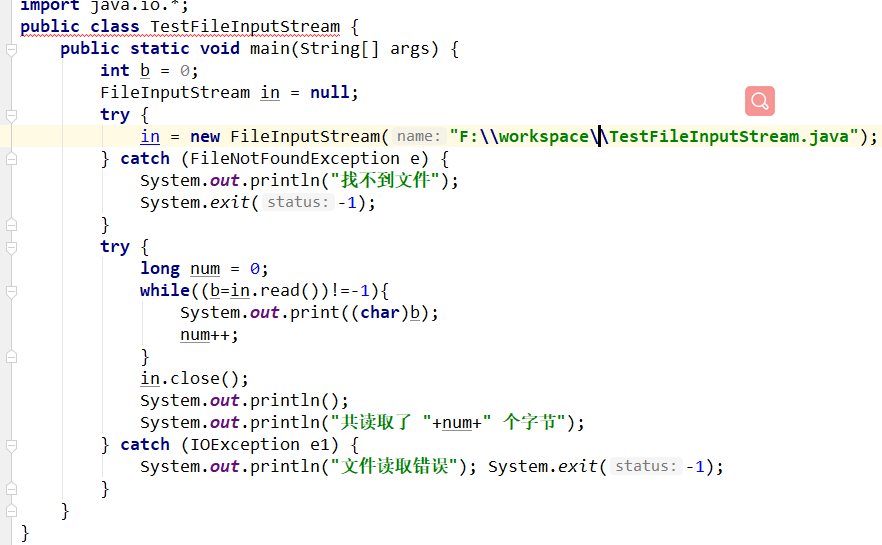
##### InputStream





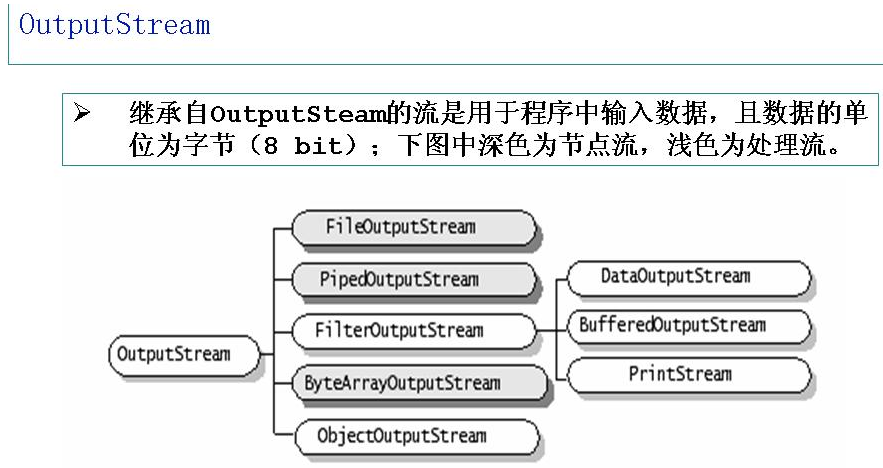


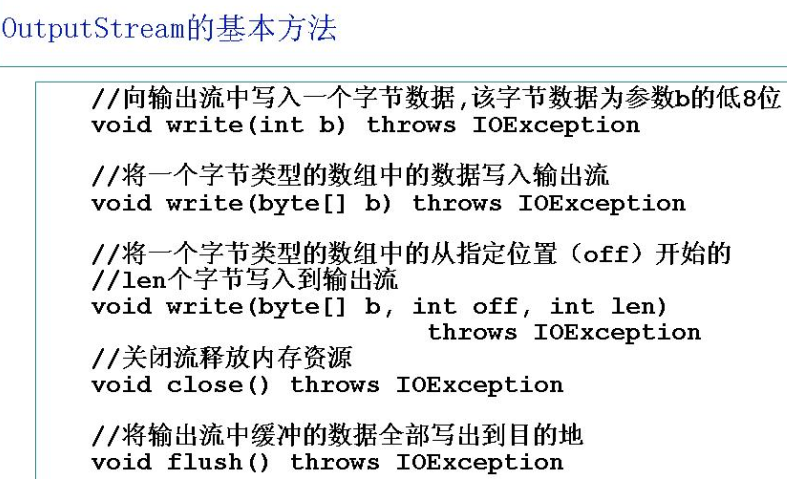
例子程序：读取工作空间下



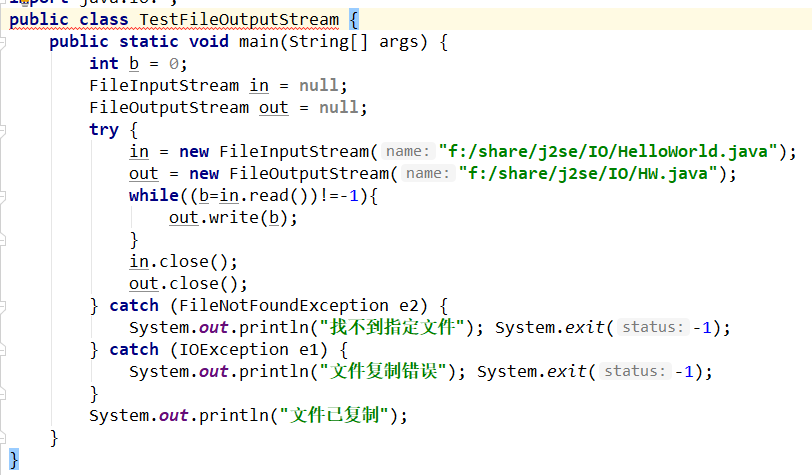
结果：由于read方法一次读取的是一个字节，一个汉字占的2个字节，所以打印出来汉字是乱码。

##### OutputStream

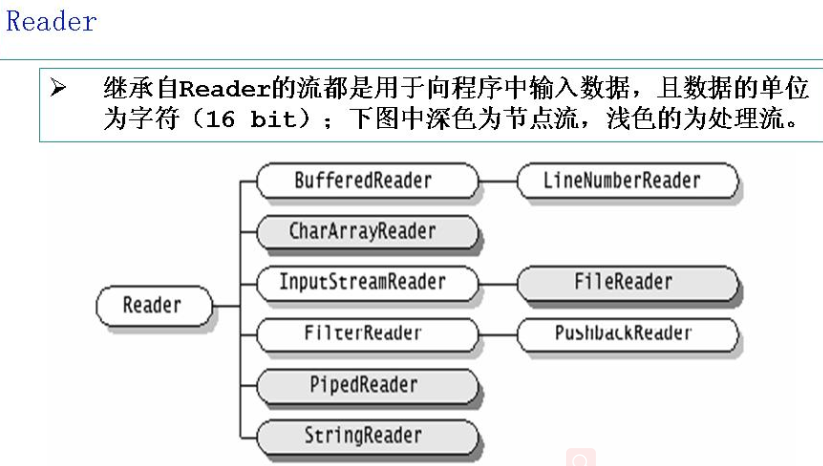


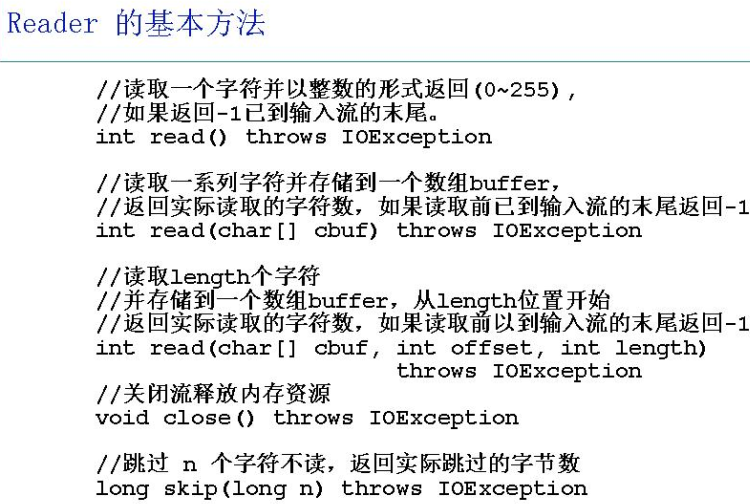


例子程序：文件的复制，如果文件不存在会自动创建HW.java 文件

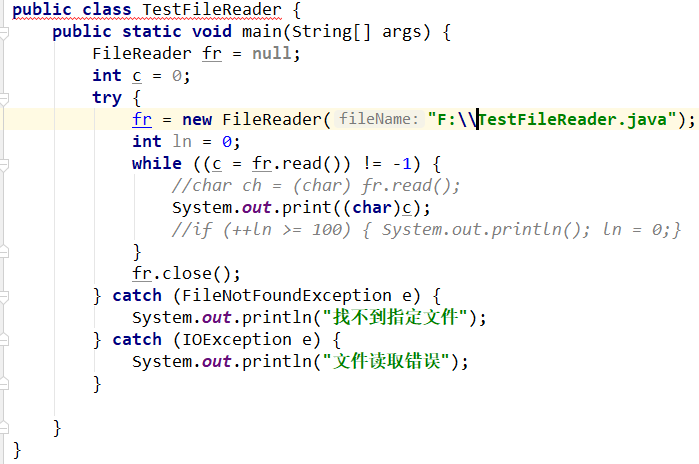


##### Reader

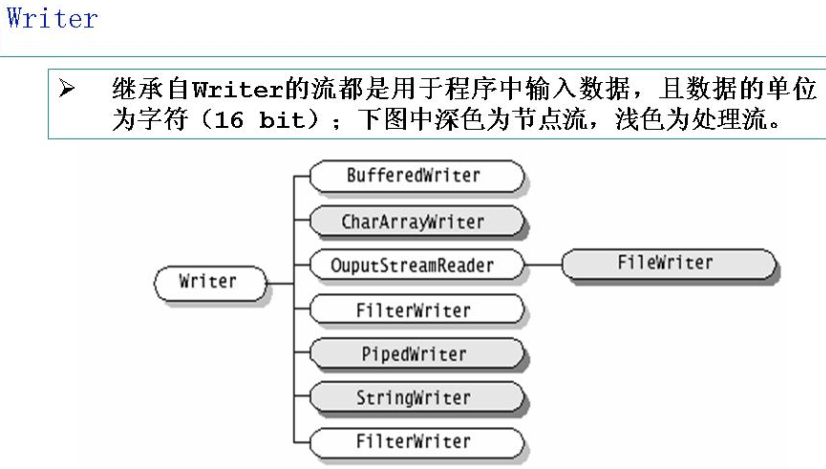


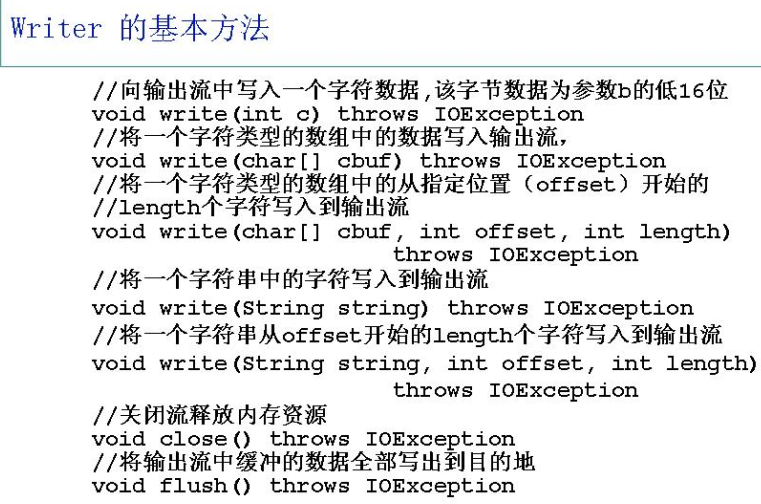


例子程序：读取本身



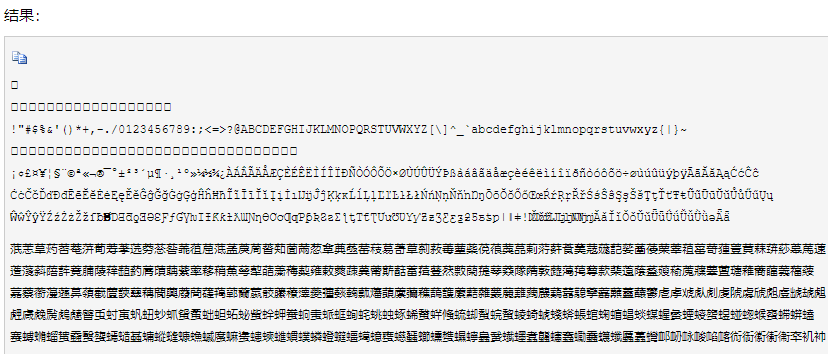
##### FileWriter



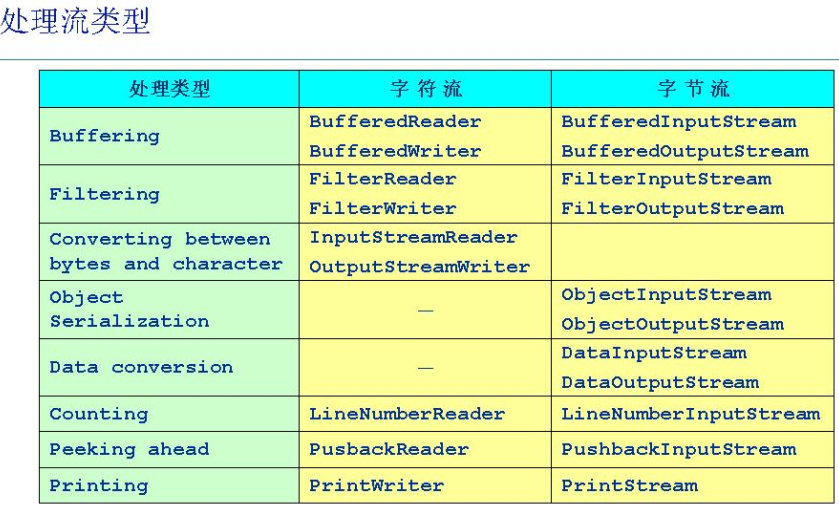


例子程序





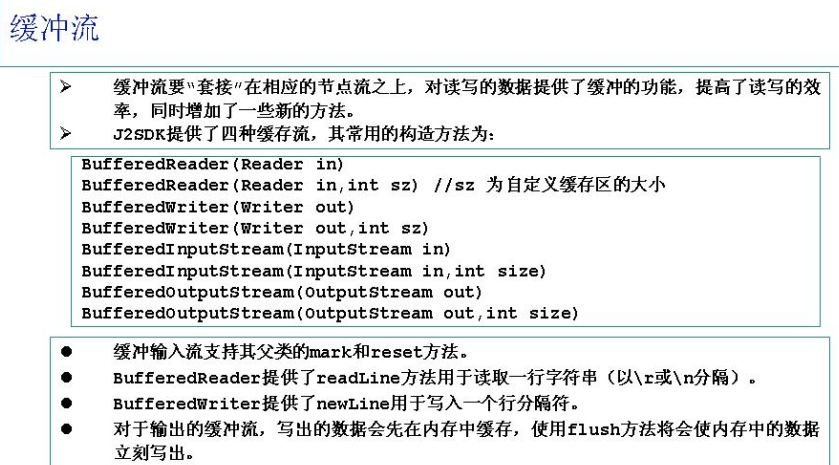
#### 处理流



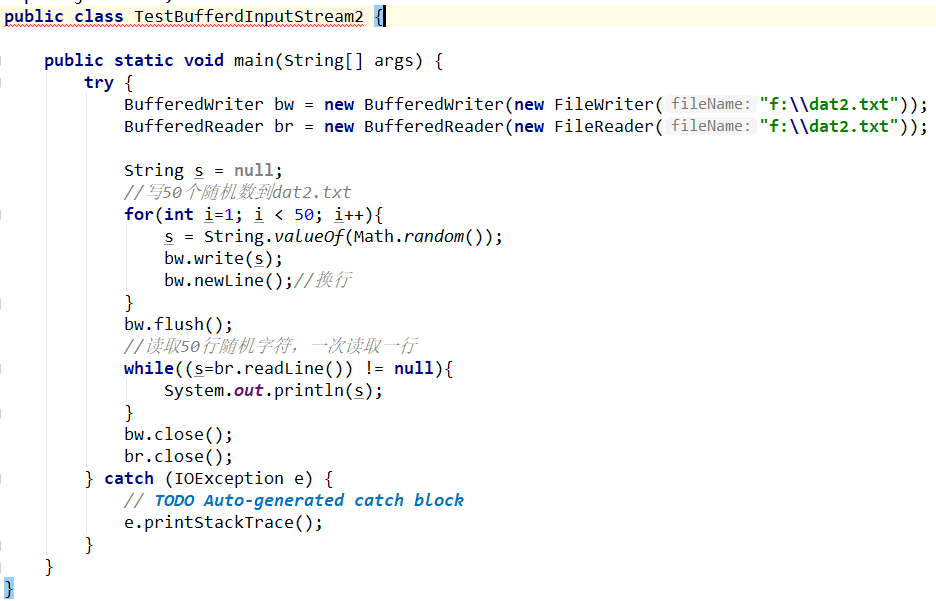
**BufferdInputStream：读取helloworld.java**



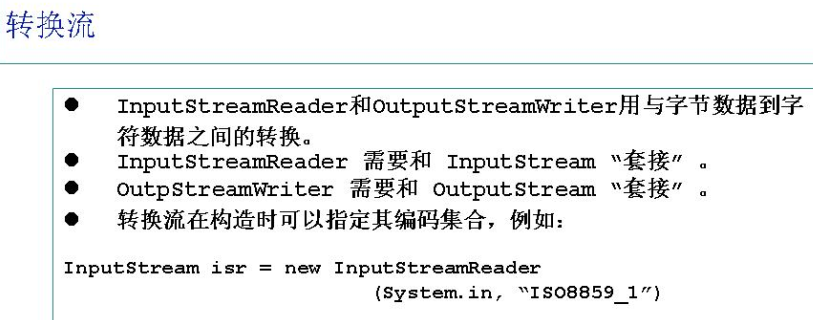
#### 缓冲流



**BufferedWriter\BufferedReader:**



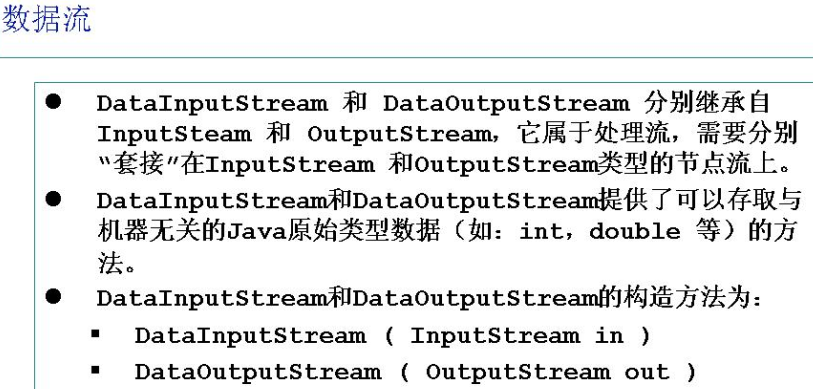
#### 转换流

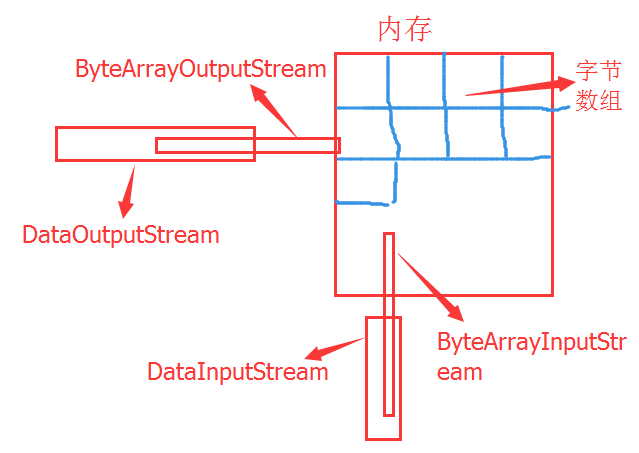


例子程序：



#### DataIO&ByteArrayIO







打印流

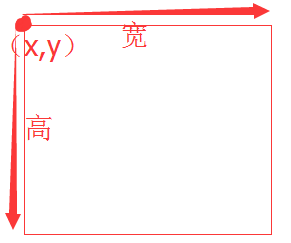
对象流

#### [JAVA对象流与序列化](https://www.cnblogs.com/lihaoyang/p/6922846.html)

Object流,直接把obj写入或读出。

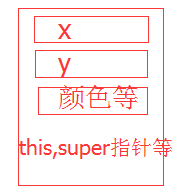
**前言：**

比如 画图的程序，咣当画一个三角形出来，咣当画一正方形出来。然后存盘，当你下次再打开软件的时候三角形、方块还在原来的位置上。如果用面向对象的思维，三角形，方块都有自己的成员变量，x、y坐标、颜色、线的颜色、粗细等，你存盘的时候要把这些内容写到硬盘上，写到文件里，这叫存盘。比如说一个方块，你存盘的时候只要把起始点的坐标，宽度，高度颜色等等存进去，当再显示的时候再在原处把它画出来。



**序列化：把一个object直接转换成字节流写到硬盘上或者写到网络上。**

实际在内存中会new出方块的一个对象来，有它各种的属性，存盘的时候要挨着把这些属性都给存到文件里，用DataOutputStream就可以实现，不管它的属性是int，dubbo等。既然把这个object的内容都要写到硬盘上，何不把整个object全部写进去呢，因为把object  new出来的时候无非也就是内存里的一块区域，也就是一个字节一个字节的，干脆把整个字节，这一小块内存全都写进硬盘，更合适，更方便，这就是object的初衷。当然New出来的对象除了自己的属性外，还有标识object的版本号，this、super指针，相关的内容都会写进去。这就是object流的作用。这就是序列化。



##### java.io.Serializable:

可序列化的。如果想吧一个类的对象写到硬盘上或者网络上，想把它序列化成一个字节流，必须实现序列化接口。

可序列化类的所有子类型本身都是可序列化的。

没有方法，标记性的接口，给编译器看的，编译器看到这个接口后就知道这个类能被序列化。

例子程序：

