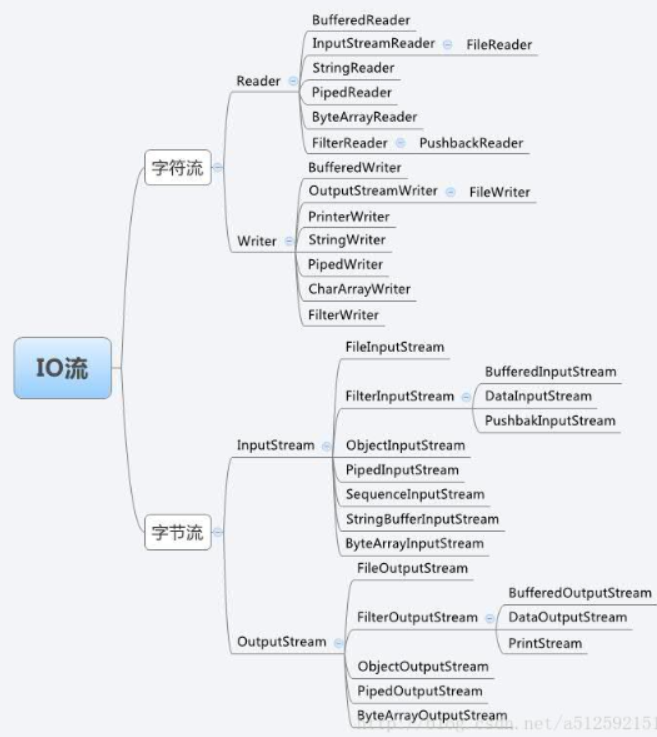
## IO 和NIO相关

I/O问题可以说是当今Web应用中所面临的主要问题之一，因为在当前这个海量数据时代，数据在网络中随处流动。在这个流动过程中都涉及到IO问题，可以说大部分Web应用系统的瓶颈都是IO瓶颈。

### JAVA中的IO基本架构

1. 基于字节操作的io接口: InputStream 和 OutputStream。
2. 基于字符操作的io接口:Writer和Reader。
3. 基于磁盘操作的io接口:File。
4. 基于网络操作的io接口:Socket。

1和2主要是数据传输的格式，后两组主要是数据传输的方式。

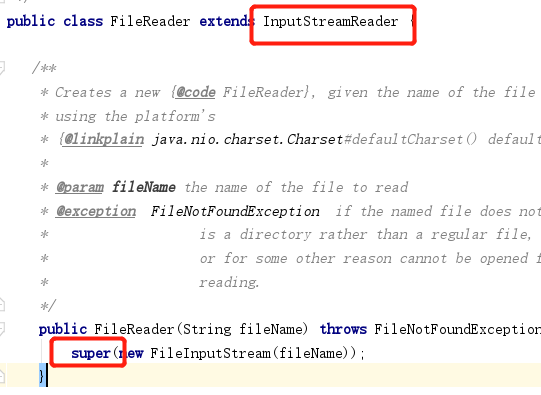
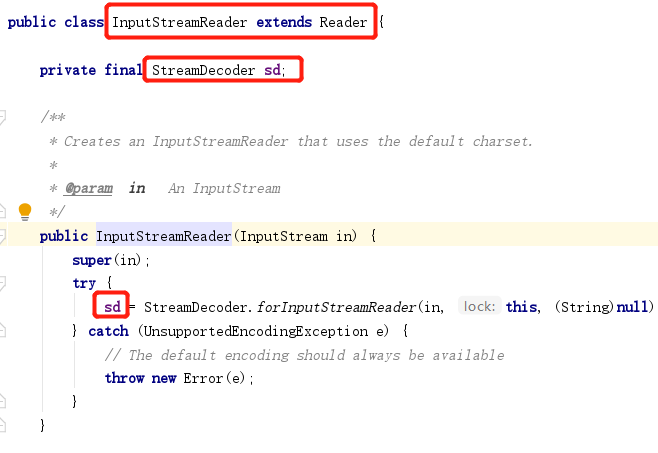


除此之外，还可以分为节点流(低级流)和高级流。

低级流：可以从或向一个特定的地方（节点）读写数据。如FileReader。

高级流：是对一个已存在的流的连接和封装，通过所封装的流的功能调用实现数据读写。如BufferedReader.处理流的构造方法总是要带一个其他的流对象做参数。一个流对象经过其他流的多次包装，称为流的链接。如：InputStreamReader OutputStreamReader BufferedInputStrean BufferedOutputStream BufferedReader BufferedWriter DataInputStream DataOutputStream。

\* 字节与字符的转化接口：数据持久化或网络传输都是以字节进行的，所以必须要有从字符到字节或从字节到字符的转化。InputStreamReader类是从字节到字符的转化桥梁，从Inputtream到Reader的过程要指定编码字符集，否则将采用系统默认的字符集，很可能出现乱码，StreamDecoder正式从你字节到字符的解码的实现类。例如，FileReader继承了InputStreamReader类，实际上是读取文件流，然后通过StreamDecoder解码成char。

### 磁盘IO工作机制

应用程序要访问物理设备只能通过系统调用的方式来工作，读和写分别是read()和write()两个系统调用。而只要是系统调用就可能存在内核空间地址和用户空间地址切换的问题，这是操作系统为了保护系统本身的运行按权，而将内核程序运行使用的内存空间和用户程序运行的内存空间进行隔离造成的。这样保证了内存程序运行的安全性，但是也必然存在数据可能需要从内存空间想用户空间复制的问题。，为了加速io访问，在内核空间使用缓存机制，也就是将从磁盘读取的文件按照一定的组织方式进行缓存，如果用户访问的是同一段磁盘地址的空间数据，那么操作系统将从内核缓存中直接取出返回给用户程序，这样可以减少io响应时间。

#### 几种访问文件的方式：

1. 标准访问文件的方式