# IO

## 基础概念

　Java语言的I/O库是对各种常见的流源，流汇以及处理过程的抽象化。客户端的Java程序不必知道最终的流源，流汇是磁盘上的文件还是数组等；也不必关心数据是否经过缓冲的，可否按照行号读取等处理的细节。

IO流分类

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

字节流字符流

字符流的由来：因为数据编码的不同，而有了对字符进行高效操作的流对象。本质其实就是基于字节流读取时，去查了指定的码表。字节流和字符流的区别：

（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。

（3）字节流在操作的时候本身是不会用到缓冲区的，是文件本身的直接操作的；而字符流在操作的时候下后是会用到缓冲区的，是通过缓冲区来操作文件。

Java I/O库具有两个对称性

　１　输入－输出对称性，比如InputStream和OutputStream各自占据Byte流的输入与输出的两个平行的等级结构的根部。而Reader和Writer各自占据Char流的输入与输出的两个平行的等级结构的根部。

　２　byte-char对称，InputStream和Reader的子类分别负责Byte和Char流的输入；OutputStream和Writer的子类分别负责Byte和Char流的输出，它们分别形成平行的等级结构。

Java IO的用途和特征

Java IO中包含了许多InputStream、OutputStream、Reader、Writer的子类。这样设计的原因是让每一个类都负责不同的功能。这也就是为什么IO包中有这么多不同的类的缘故。各类用途汇总如下：

文件访问、网络访问、内存缓存访问、线程内部通信(管道)、缓冲、过滤、解析、读写文本 (Readers / Writers)

读写基本类型数据 (long, int etc.)、读写对象

## IO流结构

定义和结构说明：

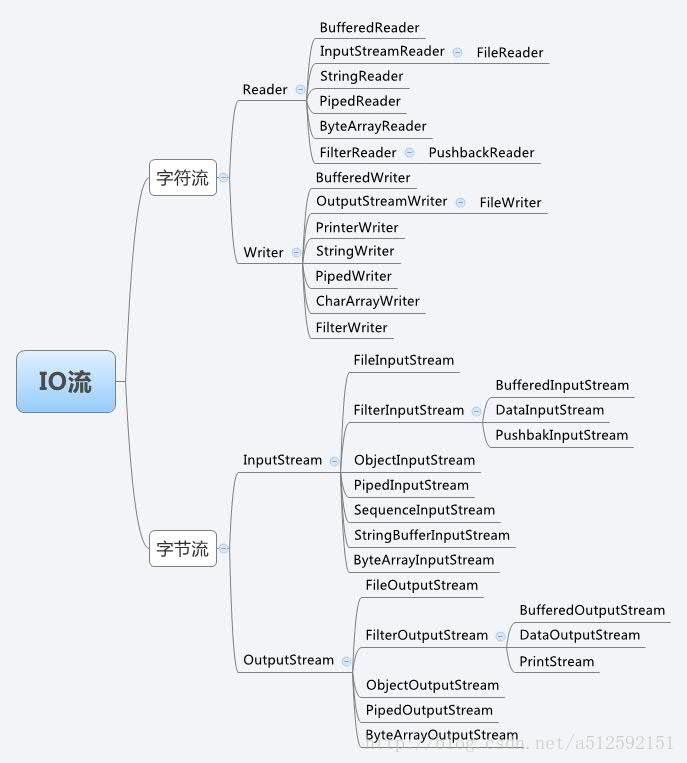
 从输入字节流的继承图可以看出：

 InputStream 是所有的输入字节流的父类，它是一个抽象类。

ByteArrayInputStream、StringBufferInputStream、FileInputStream 是三种基本的介质流，它们分别从Byte 数组、StringBuffer、和本地文件中读取数据。PipedInputStream 是从与其它线程共用的管道中读取数据，与Piped 相关的知识后续单独介绍。

 ObjectInputStream 和所有FilterInputStream的子类都是装饰流（装饰器模式的主角）。意思是FileInputStream类可以通过一个String路径名创建一个对象，FileInputStream(String name)。而DataInputStream必须装饰一个类才能返回一个对象，DataInputStream(InputStream in)。

IO流结构图



输出字节流OutputStream

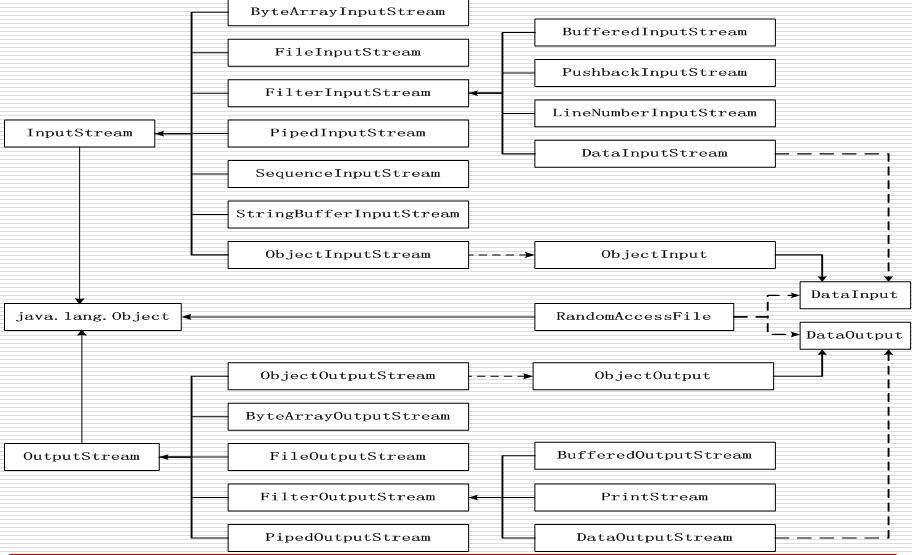
定义和结构说明：

IO 中输出字节流的继承图可见上图，可以看出：

OutputStream 是所有的输出字节流的父类，它是一个抽象类。

ByteArrayOutputStream、FileOutputStream是两种基本的介质流，它们分别向Byte 数组、和本地文件中写入数据。PipedOutputStream 是向与其它线程共用的管道中写入数据，

ObjectOutputStream 和所有FilterOutputStream的子类都是装饰流。具体例子跟InputStream是对应的。

以InputStream和OutputStream形成的层次关系

字符输入流Reader

定义和说明：

在上面的继承关系图中可以看出：

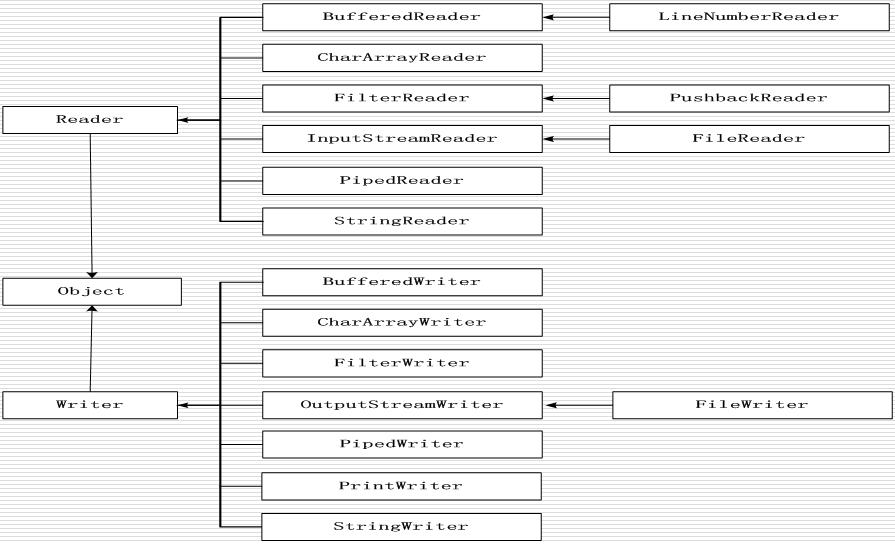
Reader 是所有的输入字符流的父类，它是一个抽象类。

CharReader、StringReader是两种基本的介质流，它们分别将Char 数组、String中读取数据。PipedReader 是从与其它线程共用的管道中读取数据。

BufferedReader 很明显就是一个装饰器，它和其子类负责装饰其它Reader 对象。

FilterReader 是所有自定义具体装饰流的父类，其子类PushbackReader 对Reader 对象进行装饰，会增加一个行号。

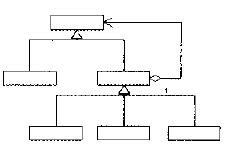
InputStreamReader 是一个连接字节流和字符流的桥梁，它将字节流转变为字符流。FileReader可以说是一个达到此功能、常用的工具类，在其源代码中明显使用了将FileInputStream 转变为Reader 的方法。我们可以从这个类中得到一定的技巧。Reader 中各个类的用途和使用方法基本和InputStream 中的类使用一致。后面会有Reader 与InputStream 的对应关系。

以Reader和Writer形成的层次关系

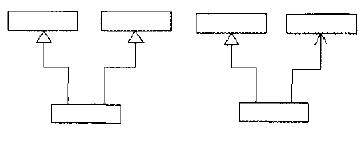
## Java I/O库的两个设计模式：

  Java的I/O库总体设计是符合装饰者模式（Decorator）跟适配器模式（Adapter）的。如前所述，这个库中处理流的类叫做流类。引子里所谈到的FileInputStream，FileOutputStream，DataInputStream及DataOutputStream都是流处理器的例子。

　１　装饰者模式：在由InputStream，OutputStream,Reader和Writer代表的等级结构内部，有一些流处理器可以对另一些流处理器起到装饰作用，形成新的，具有改善了的功能的流处理器。装饰者模式是Java I/O库的整体设计模式。这样的一个原则是符合装饰者模式的，如下图所示：



２　适配器模式：在由InputStream，OutputStream,Reader和Writer代表的等级结构内部，有一些流处理器是对其它类型的流源的适配。这就是适配器模式的应用，如下图所示。



　适配器模式应用到了原始流处理器的设计上面，构成了I/O库所有流处理器的起点。

### 装饰模式的应用

根据输入流的源的类型，可以将这些流类分成两种，即原始流类（Original Stream）和链接流处理器（Wrapper Stream）。

**原始流处理器**

　　原始流处理器接收一个Byte数组对象，String对象，FileDiscriptor对象或者不同类型的流源对象，根据上面的图，原始流处理器包括以下四种：

　　ByteArrayInputStream：为多线程的通信提供缓冲区操作功能，接收一个Byte数组作为流的源。

　　FileInputStream:建立一个与文件有关的输入流。接收一个File对象作为流的源。

　　PipedInputStream：可以与PipedOutputStream配合使用，用于读入一个数据管道的数据，接收一个PipedOutputStream作为源。

　　StringBufferInputStream：将一个字符串缓冲区转换为一个输入流。接收一个String对象作为流的源。（ＪＤＫ帮助文档上说明：**已过时。**此类未能正确地将字符转换为字节。从ＪＤＫ1.1开始，从字符串创建流的首选方法是通过StringReader类进行创建。只有字符串中每个字符的低八位可以由此类使用。）

**链接流处理器**

　所谓链接流处理器，就是可以接收另一个流对象作为源，并对之进行功能扩展的类。InputStream类型的链接处理器包括以下几种，它们都接收另一个InputStream对象作为流源。

　　（１）FilterInputStream称为过滤输入流，它将另一个输入流作为流源。这个类的子类包括以下几种：

　　BufferedInputStream：用来从硬盘将数据读入到一个内存缓冲区中，并从缓冲区提供数据。

　　DataInputStream：提供基于多字节的读取方法，可以读取原始类型的数据。

　　LineNumberInputStream：提供带有行计数功能的过滤输入流。

　　PushbackInputStream：提供特殊的功能，可以将已经读取的字节“推回”到输入流中。

　　（２）ObjectInputStream可以将使用ObjectInputStream串行化的原始数据类型和对象重新并行化。

　　（３）SeqcueneInputStream可以将两个已有的输入流连接起来，形成一个输入流，从而将多个输入流排列构成一个输入流序列。

### 适配器模式的应用

**InputStream原始流处理器中的适配器模式**

　InputStream类型的原始流处理器是适配器模式的应用。

　ByteArrayInputStream是一个适配器类

　ByteArrayInputStream继承了InputStream的接口，而封装了一个byte数组。换言之，它将一个byte数组的接口适配成InputStream流处理器的接口。

　我们知道Java语言支持四种类型：Java接口，Java类，Java数组，原始类型（即int,float等）。前三种是引用类型，类和数组的实例是对象，原始类型的值不是对象。

 也即，Java语言的数组是像所有的其他对象一样的对象，而不管数组中所存储的元素类型是什么。

 这样一来的话，ByteArrayInputStream就符合适配器模式的描述，是一个对象形式的适配器类。

**FileInputStream是一个适配器类**

 在FileInputStream继承了InputStrem类型，同时持有一个对FileDiscriptor的引用。这是将一个FileDiscriptor对象适配成InputStrem类型的对象形式的适配器模式

**OutputStream原始流处理器中的适配器模式**

 同样地，在OutputStream类型中，所有的原始流处理器都是适配器类。

   ByteArrayOutputStream继承了OutputStream类型，同时持有一个对byte数组的引用。它一个byte数组的接口适配成OutputString类型的接口，因此也是一个对象形式的适配器模式的应用。

**FileOutputStream是一个适配器类**

FileOutputStream继承了OutputStream类型，同时持有一个对FileDiscriptor对象的引用。这是一个将FileDiscriptor接口适配成OutputStream接口形式的对象形适配器模式。

**Reader原始流处理器中的适配器模式**

Reader类型的原始流处理器都是适配器模式的应用。

**StringReader是一个适配器类**

StringReader类继承了Reader类型，持有一个对String对象的引用。它将String的接口适 配成Reader类型的接口

## 扩展阅读：

File类

File类是对文件系统中文件以及文件夹进行封装的对象，可以通过对象的思想来操作文件和文件夹。 File类保存文件或目录的各种元数据信息，包括文件名、文件长度、最后修改时间、是否可读、获取当前文件的路径名，判断指定文件是否存在、获得当前目录中的文件列表，创建、删除文件和目录等方法。

【案例 】乱码的产生

9. public class CharSetDemo2{

10.     public static void main(String[] args) throws IOException{

11.         File file = new File("d:" + File.separator + "hello.txt");

12.         OutputStream out = new FileOutputStream(file);

13.         byte[] bytes = "你好".getBytes("ISO8859-1");

14.         out.write(bytes); 15.         out.close();

16.     }//输出结果为乱码，系统默认编码为GBK，而此处编码为ISO8859-1 17.

}

## 附录：

\* 字符流 : 字符流只能操作纯字符数据，比较方便。

 字节流   |--InputStream(字节输入流,字节流的抽象父类,抽象类不能直接创建对象,必须使用子类)

   |--FileInputStream(文件字节输入流,就是以字节的形式读取文件内容)     |--read()每次读取一个字节(byte),但是返回值类型,为int类型,防止出现文件提前结束

|--read(byte[] b)每次读取多个字节,将内容放入字节数组b中

   |--BufferedInputStream(字节输入缓冲流,其实就是一个包装类,创建对象时要接收一个InputStream,内部封装了一个1024\*8的字节数组)

    |--使用BufferedInputStream就不用再定义数组了

  |--OutputStream(字节输出流,字节流的抽象父类)

   |--FileOutputStream(文件字节输出流,就是以字节的形式将内容写入到文件中)

    |--write(int b)一次写出一个字节

    |--write(byte[] b)将字节数组b里的内容写入到文件中

|--write(byte[] b, int off, int len)将字节数组b里的内容,从索引为off的

位置开始,总共写len个

   |--BufferedOutputStream(字节输出缓冲流,其实就是一个包装类,创建对象

时要接收一个OutputStream,内部封装了一个1024\*8的字节数组)

 字符流

  |--Reader(字符输入流,字符流的抽象父类,抽象类不能直接创建对象,必须使用子类)

   |--FileReader(文件字符输入流,就是以字符的形式读取文件中的内容)     |--read()每次读取一个字符,即可以按照字符大小读取,返回值类型提升为int类型

|--read(char[] b)每次读取多个字符,将内容放入字符数组b中    |--BufferedReader(字符输入缓冲流,其实就是一个包装类,创建对象时要接收一个Reader,内部封装了一个1024\*8的字符数组)

    |--使用BufferedReader和BufferedWriter就不用再定义数组了     |--readLine()可以读取一行字符(不包含换行符号)

    |--LineNumberReader是BufferedReader的子类, 具有相同的功能, 并且可以统计行号

     |--getLineNumber()方法可以获取当前行号

|--setLineNumber()方法可以设置当前行号

  |--Writer(字符输出流,字符流的抽象父类)    |--FileWriter(文件字符输出流,就是以字符的形式将内容写入到文件中)     |--write('a')/write("aaaa")可以写入单个字符或者写入字符串

|--write(char[] b, int off, int len)将字符数组b里的内容,从所以为off的

位置开始,总共写len个

   |--BufferedWriter(字符输出缓冲流,其实就是一个包装类,创建对象时要接

收一个Writer,内部封装了一个1024\*8的字符数组)

    |--newLine()可以输出一个跨平台的换行符号"\r\n"

 转换流(使用指定的码表读写字符)

  |--InputStreamReader字节输入流到字符输入流的桥梁   |--OutputStreamWriter字符输出流到字节输出流的桥梁

 其它流

  |--SequenceInputStream序列流    |--序列流可以把多个字节输入流整合成一个, 从序列流中读取数据时, 将从被整合的第一个流开始读, 读完一个之后继续读第二个, 以此类推.

   |--SequenceInputStream(InputStream in1,InputStream in2)    |--SequenceInputStream(Enumeration en)

  |--ByteArrayOutputStream字节数组内存输出流    |--是将文件或键盘录入的数据写入到内存中的字节数组   |--ByteArrayInputStream字节数组内存输入流

   |--是以字节的方式读取内存中的字节数组数据    |--创建此对象要传递一个字节数组   |--RandomAccessFile随机读写访问流

   |--该流同时具有读和写的功能

   |--seek(long pos)设置读取和写入的位置

  |--对象操作流

   |--ObjectOutputStream将对象写入到文件,要求对象所属的类实现Serializable接口,该接口里面没有内容,只是一个标记

|--writeObject()将对象写入文件

   |--ObjectInputStream将上面存入到文件中的对象读取出来,因为存的时候所有对象是按照Object存入,所以取出来要进行强制类型转换

|--readObject()从文件读取一个对象出来

  |--数据输入输出流

   |--DataInputStream数据输入流,就是从文件读取特定类型的数据     |--readInt()从文件中读取一个int类型的数据

|--readLong()从文件读取一个long类型的数据