# 概述

## 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用于操作流的类都在IO包中

**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

### 按操作数据单位

字节流(8 bit)/二进制流: 字节流可以操作任何数据,因为在计算机中任何数据都是以字节的形式存储的

字符流(16 bit) : 字符流只能操作纯字符数据，比较方便。

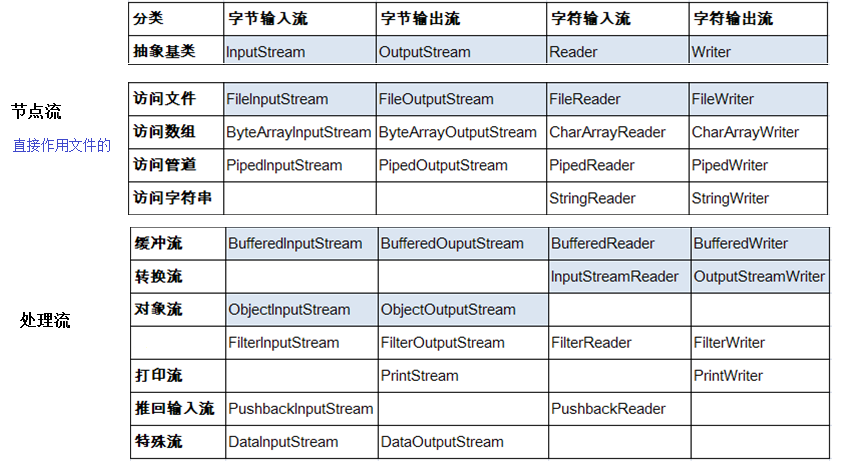
### 按流向

输入流，输出流

### 按流的角色的

**节点流**：直接作用文件的

**处理流**：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。





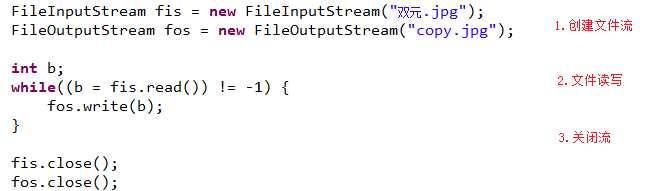
## 重要的继承关系

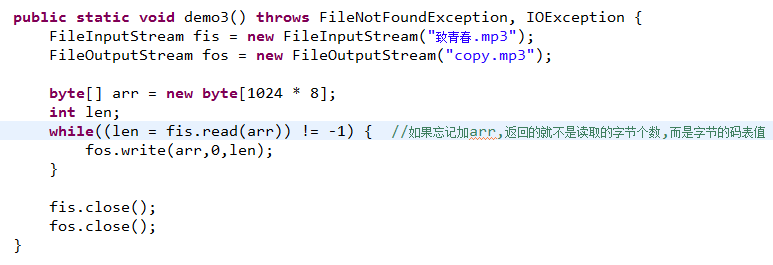
## IO程序书写及一般流程

使用前，导入IO包中的类

使用时，进行IO异常处理

使用后，释放资源





# 抽象基类

## 概述



IO流共涉及40多个类，都是从如下4个抽象基类派生的。

## InputStream

### 构造方法

[**InputStream**](mk:@MSITStore:F:\a桌面数据\参考API\jdk\JDK_API_1_6_zh_CN.CHM::/java/io/InputStream.html#InputStream())()

### 常用方法

int available() 获取读的文件所有的字节个数

void close()

关闭此输入流并释放与该流关联的所有系统资源。

void mark(int readlimit)

在此输入流中标记当前的位置。

boolean markSupported()

测试此输入流是否支持 mark 和 reset 方法。

abstract int read()

从输入流中读取数据的下一个字节。

int read(byte[] b) 一次读取一个字节数组

int read(byte[] b, int off, int len)

将输入流中最多 len 个数据字节读入 byte 数组。

void reset()

将此流重新定位到最后一次对此输入流调用 mark 方法时的位置。

long skip(long n)

跳过和丢弃此输入流中数据的 n 个字节。

### 注意

read()方法读取的是一个字节,为什么返回是int,而不是byte？

答：因为字节输入流可以操作任意类型的文件,比如图片音频等,这些文件底层都是以二进制形式的存储的,如果每次读取都返回byte,有可能在读到中间的时候遇到111111111

那么这11111111是byte类型的-1,我们的程序是遇到-1就会停止不读了,后面的数据就读不到了,所以在读取的时候用int类型接收,如果11111111会在其前面补上

24个0凑足4个字节,那么byte类型的-1就变成int类型的255了这样可以保证整个数据读完,而结束标记的-1就是int类型

\* E.小数组的读写和带Buffered的读取哪个更快?

\* 定义小数组如果是8192个字节大小和Buffered比较的话

\* 定义小数组会略胜一筹,因为读和写操作的是同一个数组

\* 而Buffered操作的是两个数组

flush和close方法的区别

flush()方法

用来刷新缓冲区的,刷新后可以再次写出

close()方法

用来关闭流释放资源的的,如果是带缓冲区的流对象的close()方法,不但会关闭流,还会再关闭流之前刷新缓冲区,关闭后不能再写出

都具备刷新功能，二者不能共存

qq会话：假入缓冲区1024kb，如果只用close，就必须要别人的会话容量达到屏幕才回显示消息，而flush可以实时刷新，输入一句话，就显示一条

## OutputStream

### 构造方法

[**OutputStream**](mk:@MSITStore:F:\a桌面数据\参考API\jdk\JDK_API_1_6_zh_CN.CHM::/java/io/OutputStream.html#OutputStream())()

### 常用方法

void write(byte[] b) 一次写出一个字节数组

void write(byte[] b, int off, int len) 将指定 byte 数组中从偏移量 off 开始的 len 个字节写入此输出流。

abstract void write(int b) 将指定的字节写入此输出流。

void close() 关闭此输出流并释放与此流有关的所有系统资源。

void flush() 刷新此输出流并强制写出所有缓冲的输出字节。

## Reader

### 字段

protected Object lock

用于同步针对此流的操作的对象。

### 构造方法

protected Reader()

创建一个新的字符流 reader，其重要部分将同步其自身的 reader。

protected Reader(Object lock)

创建一个新的字符流 reader，其重要部分将同步给定的对象。

### 常用方法

abstract void close()

关闭该流并释放与之关联的所有资源。

void mark(int readAheadLimit)

标记流中的当前位置。

boolean markSupported()

判断此流是否支持 mark() 操作。

int read()

读取单个字符。

int read(char[] cbuf)

将字符读入数组。

abstract int read(char[] cbuf, int off, int len)

将字符读入数组的某一部分。

int read(CharBuffer target)

试图将字符读入指定的字符缓冲区。

boolean ready()

判断是否准备读取此流。

void reset()

重置该流。

long skip(long n)

跳过字符。

## Wirte

### 字段

protected Object lock

用于同步针对此流的操作的对象。

### 构造方法

protected Writer()

创建一个新的字符流 writer，其关键部分将同步 writer 自身。

protected Writer(Object lock)

创建一个新的字符流 writer，其关键部分将同步给定的对象。

### 常用方法

Writer append(char c)

将指定字符添加到此 writer。

Writer append(CharSequence csq)

将指定字符序列添加到此 writer。

Writer append(CharSequence csq, int start, int end)

将指定字符序列的子序列添加到此 writer.Appendable。

abstract void close()

关闭此流，但要先刷新它。

abstract void flush()

刷新该流的缓冲。

void write(char[] cbuf)

写入字符数组。

abstract void write(char[] cbuf, int off, int len)

写入字符数组的某一部分。

void write(int c)

写入单个字符。

void write(String str)

写入字符串。

void write(String str, int off, int len)

写入字符串的某一部分。