## 1、Java IO 原理

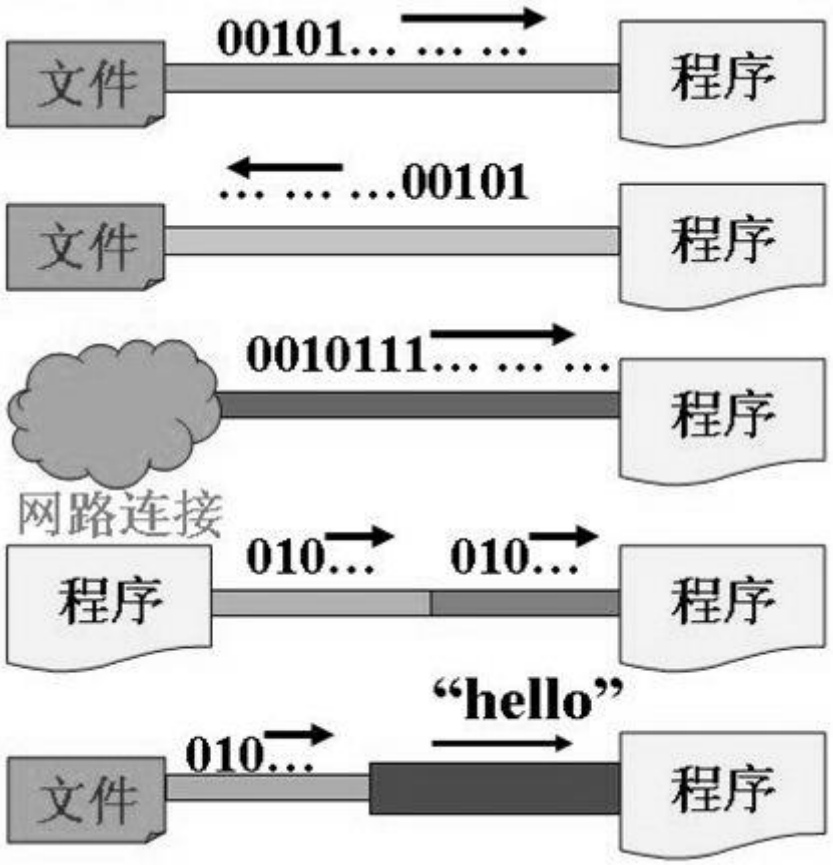
1. I/O 是Input / Output 的缩写, I/O 技术是非常实用的技术,用于处理设备之间的数据传输。如读/写文件、网络通信等。

2. Java 程序中，对于数据的输入/输出操作以"流（Stream）"的方式进行。

3. Java.io包下提供了各种"流"类和接口，用以获取不同种类的数据，并通过标准的方法输入或输出数据

4. 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。

5. 输出 output : 将程序（内存）数据输出到磁盘、光盘等存储设备中。




2、流的分类

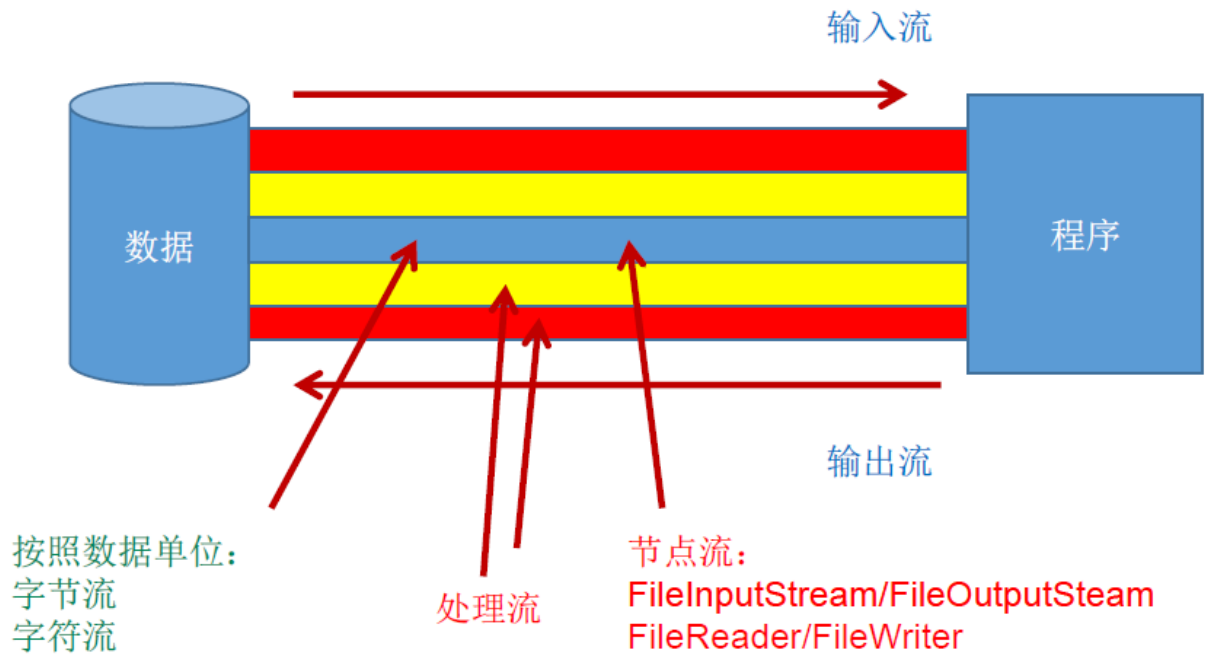
1. 按操作数据单位不同分为：字节流（8 bit）、字符流（16 bit）
2. 按数据流的流向不同分为：输入流、输出流
3. 按流的角色的不同分为：节点流、处理流

| (抽象基类) | 字节流 | 字符流 |
| --- | --- | --- |
| 输入流 | InputStream | Reader |
| 输出流 | OutputStream | Writer |

注：Java 的 IO 流共涉及 40 多个类，实际上非常规则，都是从以上 4 个抽象基类派生的

由这4个派生出来的子类名称都是以其父类名作为子类名的后缀

# 流的分类



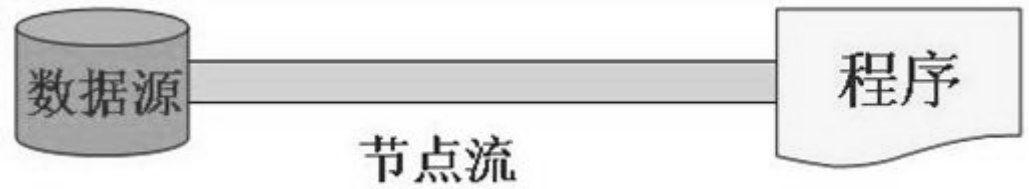

3、流的体系结构

# IO 流体系

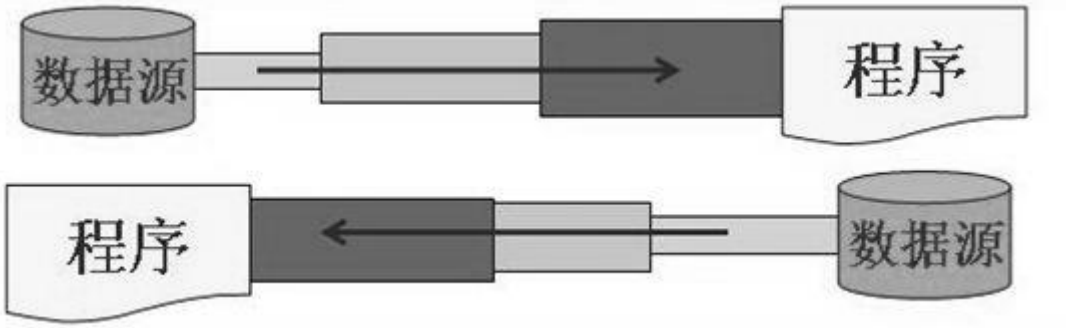
| 分类 | 字节输入流 | 字节输出流 | 字符输入流 | 字符输出流 |
|---|---|---|---|---|
| 抽象基类 | InputStream | OutputStream | Reader | Writer |
| 访问文件 | FileInputStream | FileOutputStream | FileReader | FileWriter |
| 访问数组 | ByteArrayInputStream | ByteArrayOutputStream | CharArrayReader | CharArrayWriter |
| 访问管道 | PipedInputStream | PipedOutputStream | PipedReader | PipedWriter |
| 访问字符串 | | | StringReader | StringWriter |
| 缓冲流 | BufferedInputStream | BufferedOuputStream | BufferedReader | BufferedWriter |
| 转换流 | | | InputStreamReader | OutputStreamWriter |
| 对象流 | ObjectInputStream | ObjectOutputStream | | |
| | FilterInputStream | FilterOutputStream | FilterReader | FilterWriter |
| 打印流 | | PrintStream | | PrintWriter |
| 推回输入流 | PushbackInputStream | | PushbackReader | |
| 特殊流 | DataInputStream | DataOutputStream | | |

1. 节点流（或文件流）

节点流: 直接从数据源或目的地读写数据




处理流: 不直接连接到数据源或目的地，而是“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。




1. InputStream & Reader

- InputStream 和 Reader 是所有输入流的基类
- InputStream （典型实现: FileInputStream)

a. int read()

b. int read(byte[ ] b)

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

- Reader （典型实现：FileReader)

d. int read()

e. int read(char[ ] c)

f. int read(char[] c,int off, int len)

- 程序中打开的文件 IO 资源不属于内存里的资源，垃圾回收机制无法回收该资源，所以应该显式关闭文件 IO 资源

- FileInputStream 从文件系统中的某个文件中获得输入字节。FileInputStream 用于读取非文本 数据 之类的原始字节流。要读取字符 需要使用 FileReader

1. InputStream

int read():从输入流中读取数据的下一个字节。返回0到255范围内的 int 字节值。如果因为已经到达流末尾而没有可用的字节 则返回值-1

int read(byte[] b): 从此输入流中将最多 b.length 个字节的数据读入一个 byte 数组中 。 如果因为已经到达流末尾而没有可用的字节,则返回值 -1 。 否则,以整数形式返回实际读取的字节数 。

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 。 尝试读取 len 个字节,但读取的字节也可能小于该值 。 以整数形式返回实际读取的字节数 。 如果因为流位于文件末尾而没有可用的字节,则返回值-1 。

public void close() throws IOException: 关闭此输入流并释放与该流关联的所有系统资源 。

2. Reader

int read():读取单个字符。作为整数读取的字符 范围在 0 到 65535 之间 0 x 00 0 xffff) (2 个字节的 Unicode 码) , 如果已到达流的末尾,则返回-1

int read(char[] cbuf) : 将字符读入数组。如果已到达流的末尾,则返回-1。 否则返回本次读取的字符数

int read(char[] cbuf,int off,int len): 将字符读入数组的某一部分。存到数组 cbuf 中 从 off 处开始存储 最多读 len 个字符 。 如果已到达流的

末尾,则返回-1 。 否则返回本次读取的字符数 。

public void close() throws IOException :关闭此输入流并释放与该流关联的所有系统资源 。

**2.** OutputStream & Writer

- OutputStream 和 Writer 也非常相似:
  1. void write( int b / int c)
  2. void write(byte[] b / char[] cbuf)
  3. void write(byte[] b / char[] buff,int off, int len)
  4. void flush()
  5. void close():需要先刷新，再关闭此流
- 因为 字符流 直接以字符作为操作单位，所以 Writer 可以用字符串来替换字符数组，即以 String 对象 作为
  1. void write(String str)
  2. void write(String str,int off, int len)
     - FileOutputStream 从文件系统中的某个文件中 获得输出字节 。 FileOutputStream用于 写出非文本数据之类的原始字节流。 要写出字符流需要使用 FileWriter

1. OutputStream

void write(int b)； 将指定的字节写入此输出流。write 的常规协定是：向输出流写入一个字节 。 要写入的字节是参数b的八个低位。b的24个高位将被忽略 。即写入0~255范围的

void write(byte[] b); 将b.length个字节从指定的 byte 数组写入此输出流。write(b)的常规协定是：应该与调用 write(b, 0 b length) 的效果完全相同。

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

public void flush()throws IOException: 刷新此输出流并强制写出所有缓冲的输出字节 调用此方法指示应将这些字节立即写入它们预期的目标 。

public void close() throws IOException: 关闭此输出流并释放与该流关联的所有系统资源

2. Writer

void write(int c):写入单个字符。要写入的字符包含在给定整数值的16个低位中,16个高位被忽略。即写入0到65535之间的Unicode码 。

void write(char[] cbuf):写入字符数组。

void write(char[] cbuf,int off,int len);写入字符数组的某一部分。从off开始写入len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。

void flush():刷新该流的缓冲则立即将它们写入预期目标 。

public void close() throws IOException:关闭此输出流并释放与该流关联的所有系统资源。

2. 缓冲流（处理流的一种）

a. BufferedInputStream (read(byte[] buffer))

b. BufferedOutputStream (write(byte[] buffer,0,len) / flush()

c. BufferedReader (read(char[] cbuf) / readLine())

d. BufferedWriter (write(char[] cbuf,0,len) / flush()