# java 的 IO流有什么用？

作用大了去了，java中的**输入输出都是以流的形式**，包括**文件读写和网络传输**。比如我要读取一段数据，我使用输入流的时候不用关心这数据是从哪儿读进来的，都直接用**read方法**就行了。

## 用法

Java 的控制台输入由 System.in 完成。

为了获得一个绑定到控制台的字符流，你可以把 System.in 包装在一个 BufferedReader 对象中来创建一个字符流。

下面是创建 BufferedReader 的基本语法：

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

# Android中用法

**抽象类InputStream中主要用read()与read(byte[] b)这两个方法，JDK API中是这样描述两者的：**

1. 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-1。在输入数据可用、检测到流末尾或者抛出异常前，此方法一直阻塞。

2. read(byte[] b)

从输入流中读取一定数量的字节，并将其存储在缓冲区数组 b 中。以整数形式返回实际读取的字节数。在输入数据可用、检测到文件末尾或者抛出异常前，此方法一直阻塞。如果 b 的长度为 0，则不读取任何字节并返回 0；否则，尝试读取至少一个字节。如果因为流位于文件末尾而没有可用的字节，则返回值 -1；否则，至少读取一个字节并将其存储在 b 中。将读取的第一个字节存储在元素 b[0] 中，下一个存储在 b[1] 中，依次类推。读取的字节数最多等于 b 的长度。设 k 为实际读取的字节数；这些字节将存储在 b[0] 到 b[k-1] 的元素中，不影响 b[k] 到 b[b.length-1] 的元素。

由帮助文档中的解释可知，read()方法每次只能读取一个字节，所以也只能读取由ASCII码范围内的一些字符。这些字符主要用于显示现代英语和其他西欧语言。而对于汉字等unicode中的字符则不能正常读取。只能以乱码的形式显示。

对于read()方法的上述缺点，在read(byte[] b)中则得到了解决，就拿汉字来举例，一个汉字占有两个字节，则可以把参数数组b定义为大小为2的数组即可正常读取汉字了。当然b也可以定义为更大，比如如果b=new byte[4]的话，则每次可以读取两个汉字字符了，但是需要注意的是，如果此处定义b 的大小为3或7等奇数，则对于全是汉字的一篇文档则不能全部正常读写了。

## Ds

## Ds

## D

## Ds

### Ds

## D

### S

## Ds

### D

#### S

### Ds

d