java InputStream read使用及读取数据注意

```
public abstract int read() throws IOException;
```

从输入流中读取下一个字节的数据。值字节以int的形式返回，范围在0-255之间。如果由于到达流的末尾而没有字节可用，则返回值-1。此方法会一直阻塞，直到输入数据可用、检测到流结束或抛出异常为止。

```
public int read(byte b[], int off, int len) throws IOException {...}
```

从输入流中读取一定数量的字节并将其存储到缓冲区数组b。实际读取的字节数以整数形式返回。该方法一直阻塞，直到输入数据可用，检测到文件结束，或引发异常。off 表示从 b 的什么位置开始（b的偏移量），len表示读取多少长度。

```
public int read(byte b[]) throws IOException {
        return read(b, 0, b.length);
    }
```

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

1. 关于InputStream.read()

在从数据流里读取数据时，为图简单，经常用InputStream.read()方法。这个方法是从流里每次只读取读取一个字节，效率会非常低。 更好的方法是用InputStream.read(byte[] b)或者InputStream.read(byte[] b,int off,int len)方法，一次读取多个字节。

2. 关于InputStream类的available()方法

要一次读取多个字节时，经常用到InputStream.available()方法，这个方法可以在读写操作前先得知数据流里有多少个字节可以读取。需要注意的是，如果这个方法用在从本
地文件读取数据时，一般不会遇到问题，但如果是用于网络操作，就经常会遇到一些麻烦。比如，Socket通讯时，对方明明发来了1000个字节，但是自己的程序调用available()方法却只得到900，或者100，甚至是0，感觉有点莫名其妙，怎么也找不到原因。其实，这是因为网络通讯往往是间断性的，一串字节往往分几批进行发送。本地程序调用available()方法有时得到0，这可能是对方还没有响应，也可能是对方已经响应了，但是数据还没有送达本地。对方发送了1000个字节给你，也许分成3批到达，这你就要调用3次available()方法才能将数据总数全部得到。

如果这样写代码：

```
int count = in.available();
byte[] b = new byte[count];
in.read(b);
```

在进行网络操作时往往出错，因为你调用available()方法时，对发发送的数据可能还没有到达，你得到的count是0。

需要改成这样：

```
int count = 0;
while (count == 0) {
 count = in.available();
}
byte[] b = new byte[count];
in.read(b);
```

3. 关于InputStream.read(byte[] b)和InputStream.read(byte[] b,int off,int len)这两个方法都是用来从流里读取多个字节的，有经验的程序员就会发现，这两个方法经常 读取不到自己想要读取的个数的字节。比如第一个方法，程序员往往希望程序能读取到b.length个字节，而实际情况是，系统往往读取不了这么多。仔细阅读Java的API说明就发现了，这个方法 并不保证能读取这么多个字节，它只能保证最多读取这么多个字节(最少1个)。因此，如果要让程序读取count个字节，最好用以下代码：

```
byte[] b = new byte[count];
int readCount = 0; // 已经成功读取的字节的个数
while (readCount < count) {
 readCount += in.read(bytes, readCount, count - readCount);
}
```

用这段代码可以保证读取count个字节，除非中途遇到IO异常或者到了数据流的结尾(EOFException)

部分转自：java InputStream读取数据问题_java inputstream.read(buffer)) 每次返回的字节长度不一定相同,为什么-CSDN博客