java.io

**Interface DataInput**

* **All Known Subinterfaces:**

[ImageInputStream](https://docs.oracle.com/javase/8/docs/api/javax/imageio/stream/ImageInputStream.html), [ImageOutputStream](https://docs.oracle.com/javase/8/docs/api/javax/imageio/stream/ImageOutputStream.html), [ObjectInput](https://docs.oracle.com/javase/8/docs/api/java/io/ObjectInput.html)

**All Known Implementing Classes:**

[DataInputStream](https://docs.oracle.com/javase/8/docs/api/java/io/DataInputStream.html), [FileCacheImageInputStream](https://docs.oracle.com/javase/8/docs/api/javax/imageio/stream/FileCacheImageInputStream.html), [FileCacheImageOutputStream](https://docs.oracle.com/javase/8/docs/api/javax/imageio/stream/FileCacheImageOutputStream.html), [FileImageInputStream](https://docs.oracle.com/javase/8/docs/api/javax/imageio/stream/FileImageInputStream.html), [FileImageOutputStream](https://docs.oracle.com/javase/8/docs/api/javax/imageio/stream/FileImageOutputStream.html), [ImageInputStreamImpl](https://docs.oracle.com/javase/8/docs/api/javax/imageio/stream/ImageInputStreamImpl.html), [ImageOutputStreamImpl](https://docs.oracle.com/javase/8/docs/api/javax/imageio/stream/ImageOutputStreamImpl.html), [MemoryCacheImageInputStream](https://docs.oracle.com/javase/8/docs/api/javax/imageio/stream/MemoryCacheImageInputStream.html), [MemoryCacheImageOutputStream](https://docs.oracle.com/javase/8/docs/api/javax/imageio/stream/MemoryCacheImageOutputStream.html), [ObjectInputStream](https://docs.oracle.com/javase/8/docs/api/java/io/ObjectInputStream.html), [RandomAccessFile](https://docs.oracle.com/javase/8/docs/api/java/io/RandomAccessFile.html)

public interface **DataInput**

DataInput interface提供了从二进制流中读取字节,并将读取的字节重构成基本数据类型的功能. 另外还可以将utf-8编码([modified UTF-8](https://docs.oracle.com/javase/8/docs/api/java/io/DataInput.html#modified-utf-8) )的数据重构成字符串

如果未读取完指定个数的字节之前,就到了流的结尾,那么就会报EOFException异常(IO异常的一种),如果无法读取任何的字节,报IO异常

[modified UTF-8](https://docs.oracle.com/javase/8/docs/api/java/io/DataInput.html#modified-utf-8)

java中所有的内容都是用unicode字符集(每个字符都有一个unicode字符集合中的对应的二进制值).所谓modified utf-8是指，将unicode对应的二进制值在包装下，变成按照utf-8编码方式对二进制进行了一次编码（包装）

这也就是为什么有可能一个字符使用utf-8编码后，长度有可能会变长（如原先一个字符占一个字节，经过utf-8编码后，占2个字节。），关于变长的原因，看下面表格：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Unicode值的范围为 '\u0001' - '\u007F' 之间的，utf-8编码后占1个字节 | | | | | |
|  | **Bit Values** | | | | |
| **Byte 1** | 0 | bits 6-0 | | | |
| Unicode值为 '\u0000' 或者值的范围在 '\u0080' - '\u07FF' 之间的，utf-8编码后占2个字节 | | | | | |
|  | **Bit Values** | | | | |
| **Byte 1** | 1 | 1 | 0 | bits 10-6 | |
| **Byte 2** | 1 | 0 | bits 5-0 | | |
| Unicode值的范围在  '\u0800' - '\uFFFF'之间的，utf-8编码后占3个字节 | | | | | |
|  | **Bit Values** | | | | |
| **Byte 1** | 1 | 1 | 1 | 0 | bits 15-12 |
| **Byte 2** | 1 | 0 | bits 11-6 | | |
| **Byte 3** | 1 | 0 | bits 5-0 | | |

看Byte1和Bit Values交叉的那一行，有几个1，就说明当前utf-8编码后占几个字节。后面的bits的值（例如bits5-0）的值连接起来就是字符真正的值。例如

汉字严为例，演示如何实现 UTF-8 编码。

严的 Unicode 是4E25（100111000100101），根据上表，可以发现4E25处在第三行的范围内（\u0800 - \uFFFF），因此严的 UTF-8 编码需要三个字节，即格式是1110xxxx 10xxxxxx 10xxxxxx。然后，从严的最后一个二进制位开始，依次从后向前填入格式中的x，多出的位补0。这样就得到了，严的 UTF-8 编码是11100100 10111000 10100101，转换成十六进制就是E4B8A5