Java IO系列课程03

------ Java IO流使用（二）

本文档包含以下内容：

1. 学习目标
2. 字符编码表
3. 如何处理中文乱码
4. 转换流
5. 练习题

**前言：**

编码表：就是生活中的字符和计算机二机制的对应关系表

ASCII表：一个字符中的7位就可以表示：对应的字节都是正数 0-XXXXXXX

ISO-8859-1：拉丁码表latin,用一个字节中的8位表示，可表示整数和负数 1-XXXXXXX

GBK：目前最常用的中文码表，用2个字节表示，约2万的中文和符号，2个字节的第一个字节开图是1，第二个字节的开头是0

Unicode：国际标准码表，无论什么文字，都用2个字节存储

utf-8：基于Unicode，一个字节就可以存储数据，不要用两个字节存储，而且这个码表更加标准化，

编码解码

能识别中文的码表：GBK，utf-8；正因为识别中文码表不唯一，涉及到编码解码问题

字符->字节：编码过程 eg:"haha".getByte() byte[]

字节->字符：解码过程 eg:byte[] b = {23,34,54}， new String(b,”GBK /”)

字节流与字符流的区别  
字节流和字符流使用是非常相似的，那么除了操作代码的不同之外，还有哪些不同呢？  
字节流在操作的时候本身是不会用到缓冲区（内存）的，是与文件本身直接操作的，而字符流在操作的时候是使用到缓冲区的  
字节流在操作文件时，即使不关闭资源（close方法），文件也能输出，但是如果字符流不使用close方法的话，则不会输出任何内容，说明字符流用的是缓冲区，并且可以使用flush方法强制进行刷新缓冲区，这时才能在不close的情况下输出内容

为什么字符流使用到了缓冲区？

什么叫缓冲区？  
 在很多地方都碰到缓冲区这个名词，那么到底什么是缓冲区？又有什么作用呢？  
回答：缓冲区可以简单地理解为一段内存区域。  
可以简单地把缓冲区理解为一段特殊的内存。  
 某些情况下，如果一个程序频繁地操作一个资源（如文件或数据库），则性能会很低，此时为了提升性能，就可以将一部分数据暂时读入到内存的一块区域之中，以后直接从此区域中读取数据即可，因为读取内存速度会比较快，这样可以提升程序的性能。  
 在字符流的操作中，所有的字符都是在内存中形成的，在输出前会将所有的内容暂时保存在内存之中，所以使用了缓冲区暂存数据。  
 如果想在不关闭时也可以将字符流的内容全部输出，则可以使用Writer类中的flush()方法完成。

那开发中究竟用字节流好还是用字符流好呢？  
 在所有的硬盘上保存文件或进行传输的时候都是以字节的方法进行的，包括图片也是按字节完成，而字符是只有在内存中才会形成的，所以使用字节的操作是最多的。

所以字节使用的多，但是其实没有好坏之分要视情况而定。

那我们遇到了中文字符该怎么办？

在Java里面如果编码相同的情况下，我们使用字符流就可以解决

两种解决方法： 我们先来看一下编码表

一种是newString（byte[],"gbk"）

第二种就是使用转换流。 （转换流的作用并不是这一个）

|  |
| --- |
| public class FileStreamReaderTest {  public static void main(String[] args) {  String filename = "d:/FileWriterTest1.txt"; // read(filename);  readStreamReader(filename);  }  public static void read(String fileName) {  File f = new File(fileName);  Reader input = null;  try {  input = new FileReader(f);  String line = null;  BufferedReader bufferedReader = new BufferedReader(input);  while ((line = bufferedReader.readLine()) != null) {  System.out.println(new String(line.getBytes("GBK"), "utf-8"));  }  } catch (FileNotFoundException e) {  e.printStackTrace();  } catch (IOException e) {  e.printStackTrace();  }  }   public static void readStreamReader(String fileName) {  try {  BufferedReader br = new BufferedReader(new InputStreamReader(new FileInputStream(fileName), "GBK"));  br.skip(3);  String line = null;  while ((line = br.readLine()) != null) {  System.out.println(line);  }  br.close();  } catch (IOException e) {  e.printStackTrace();  }  } } |

转换流

转换流的特点：

1. 其是字符流和字节流之间的桥梁

2. 可对读取到的字节数据经过指定编码转换成字符

3. 可对读取到的字符数据经过指定编码转换成字节

什么时候使用转换流呢？

1，源或者目的对应的设备是字节流，但是操作的却是文本数据，可以使用转换作为桥梁。提高对文本操作的便捷。

2，一旦操作文本涉及到具体的指定编码表时，必须使用转换流 。

具体的对象体现：

1. InputStreamReader:字节到字符的桥梁

2. OutputStreamWriter:字符到字节的桥梁

这两个流对象是字符体系中的成员，它们有转换作用，本身又是字符流，所以在构造的时候需要传入字节流对象进来。