
黑马程序员™
www.itheima.com
传智播客旗下
高端IT教育品牌
IO流

目录 Contents

IO流的概述和分类

以前是如何存储数据的?

```
int a = 10;
int [] arr = {1,2,3,4,5};
ArrayList<String> list = new ArrayList<>();
```

不能永久化存储，只要代码运行结束，所有数据都会丢失。

IO流的概述和分类

学习IO流的目的?

1，将数据写到文件中，实现数据永久化存储

2，读取文件中已经存在的数据

IO流的概述和分类

IO流概述

其中：I表示intput，是数据从硬盘进内存的过程，称之为读。

O表示output，是数据从内存到硬盘的过程。称之为写。

IO流的概述和分类

思考一个问题?

在数据传输的过程中，是谁在读？是谁在写？这个参照物是谁？

IO的数据传输，可以看做是一种数据的流动，按照流动的方向，以内存为参照物，进行读写操作。

简单来说：内存在读，内存在写。

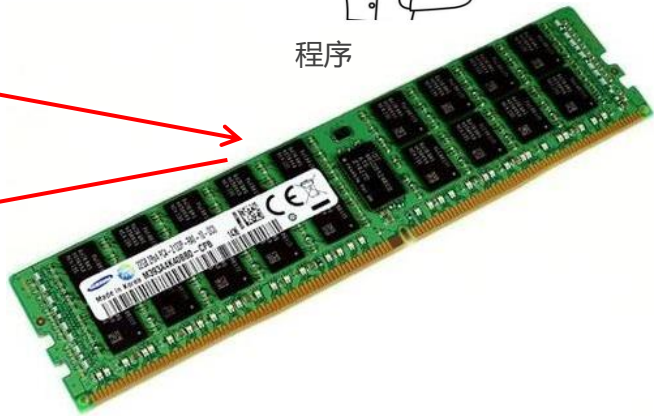
IO流的概述和分类

IO流概述和分类

输入：读数据

输出：写数据

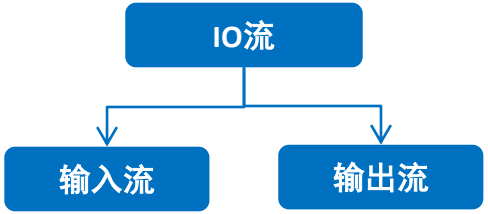
程序



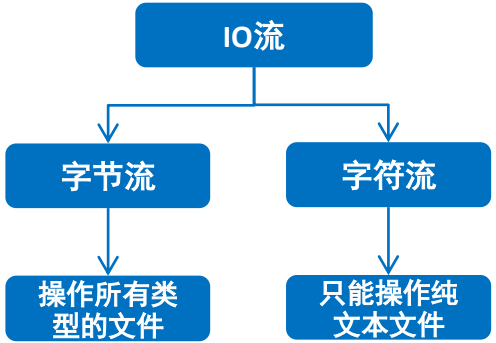
IO流的概述和分类

IO流的分类

- 按流向分



- 按数据类型分



包括音频视频图片等

包括java文件，txt文件等

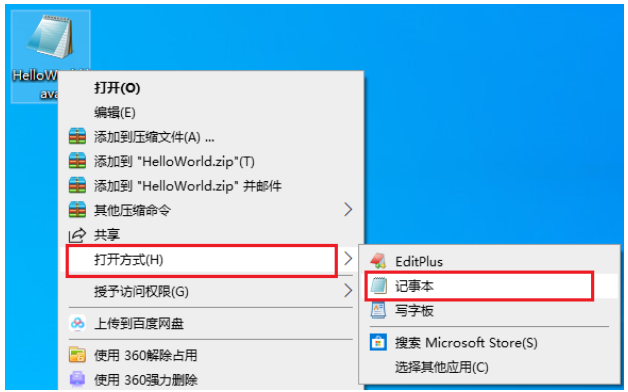
一般来说，我们说IO流的分类是按照**数据类型**来分的

IO流的概述和分类

IO流的技术选型

什么是纯文本文件?

用windows记事本打开能读的懂，那么这样的文件就是纯文本文件。



思考：office文件可以用字符流操作吗?

思考：下面这些文件分别可以用什么流操作?

目录 Contents

字节流

字节流写数据

创建随机数的步骤：

① 创建Random的对象

② 调用生成随机数的方法

字节流写数据的步骤：

① 创建字节输出流对象。

② 写数据

③ 释放资源

字节流写数据

步骤：

① 创建字节输出流对象。

注意事项：

如果文件不存在，就创建。

如果文件存在就清空。

② 写数据

注意事项：

写出的整数，实际写出的是整数在码表上对应的字母。

③ 释放资源

注意事项：

每次使用完流必须要释放资源。

字节流

字节流写数据的3种方式

方法名	说明
void write(int b)	一次写一个字节数据
void write(byte[] b)	一次写一个字节数组数据
void write(byte[] b, int off, int len)	一次写一个字节数组的部分数据

字节流写数据的两个小问题

字节流写数据如何实现换行呢?

- 写完数据后，加换行符

  windows:**\r\n**

  linux:**\n**

  mac:**\r**

字节流写数据如何实现追加写入呢?

- public FileOutputStream(String name，boolean append)
- 创建文件输出流以指定的名称写入文件。如果第二个参数为true ， 不会清空文件里面的内容

字节流写数据加try...catch异常处理

```
try {
    FileOutputStream fos = new FileOutputStream("a.txt");
    fos.write(97);
    fos.close();
} catch (IOException e) {
    e.printStackTrace();
}
```

思考：那么我们如何操作才能让close方法一定执行呢?

finally：在异常处理时提供finally块来执行所有清除操作。比如说IO流中的释放资源

特点：被finally控制的语句一定会执行，除非JVM退出

异常处理标准格式：**try....catch...finally**

字节流写数据加try...catch异常处理

```
try {
    FileOutputStream fos = new FileOutputStream("a.txt");
    fos.write(97);
    fos.close();
} catch (IOException e) {
    e.printStackTrace();
} finally{

}
```

小结：

步骤：

① 创建字节输出流对象

文件不存在，就创建。

文件存在就清空。如果不想被清空则加true

② 写数据

可以写一个字节，写一个字节数组，写一个字节数组的一部分

写一个回车换行：\r\n

③ 释放资源

字节流读数据(一次读一个字节)

步骤：

① 创建字节输入流对象。

注意事项：

如果文件不存在，就直接报错。

② 读数据

注意事项：

读出来的是文件中数据的码表值。 a → 97

③ 释放资源

注意事项：

每次使用完流必须要释放资源。

2. 字节流

案例场景

兄弟，U盘给你，分享一下先

于是，你哥们就把他C:\\itheima\\a.avi，拷贝到你U盘上了

案例：复制文件

需求：把“C:\\itheima\\a.avi”复制到当前模块下

分析：

① 复制文件，其实就把文件的内容从一个文件中读取出来(**数据源**)，然后写入到另一个文件中(**目的地**)

② 数据源：

C:\\itheima\\a.avi --- 读数据 --- FileInputStream

③ 目的地：

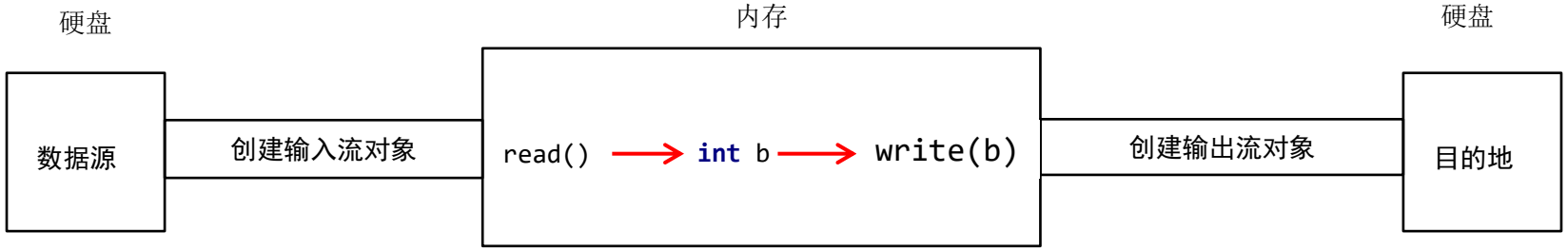
模块名称\\copy.avi --- 写数据 --- FileOutputStream

字节流

思考：如果操作的文件过大，那么速度会不会有影响？

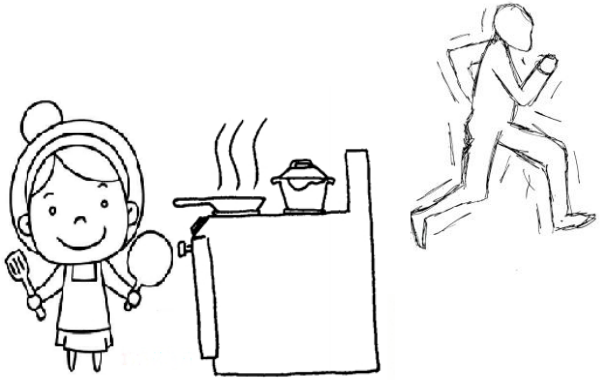
一个字：

字节流

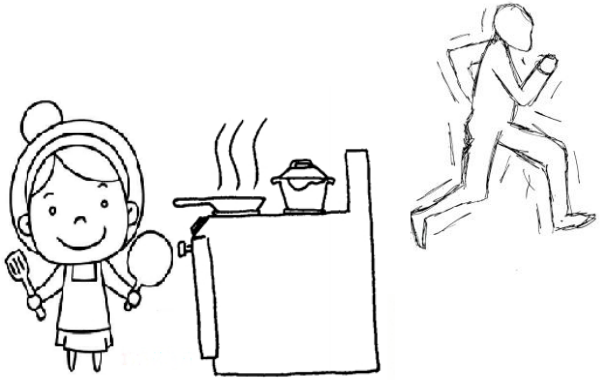



```
int b;
while((b = fis.read())!=-1){
    fos.write(b);
}
```

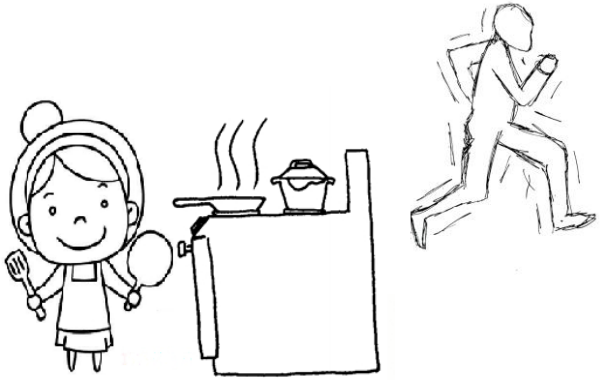
2. 字节流



2. 字节流



2. 字节流



提高拷贝速度的解决方案

为了解决速度问题，字节流通过创建字节数组，可以一次读写多个数据。

一次读一个字节数组的方法：

- public int read(byte[] b)：从输入流读取最多b.length个字节的数据
- **返回的是读入缓冲区的总字节数,也就是实际的读取字节个数**

字节流缓冲流

字节缓冲流：

- BufferOutputStream：缓冲输出流
- BufferedInputStream：缓冲输入流

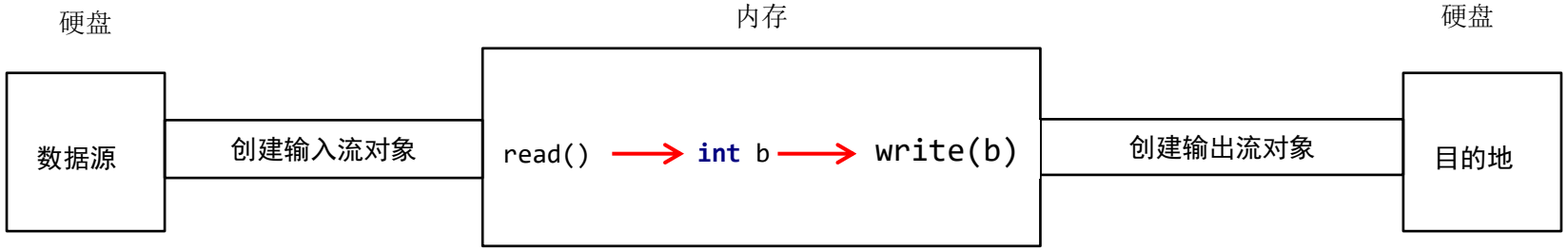
构造方法：

- 字节缓冲输出流：BufferedOutputStream(OutputStream out)
- 字节缓冲输入流：BufferedInputStream(InputStream in)

为什么构造方法需要的是字节流，而不是具体的文件或者路径呢？

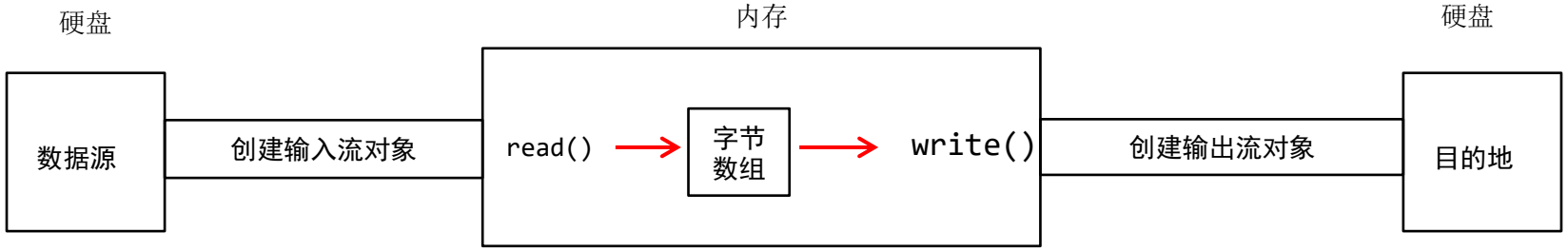
- 字节缓冲流**仅仅提供缓冲区**，而真正的读写数据还得依靠基本的字节流对象进行操作

字节流



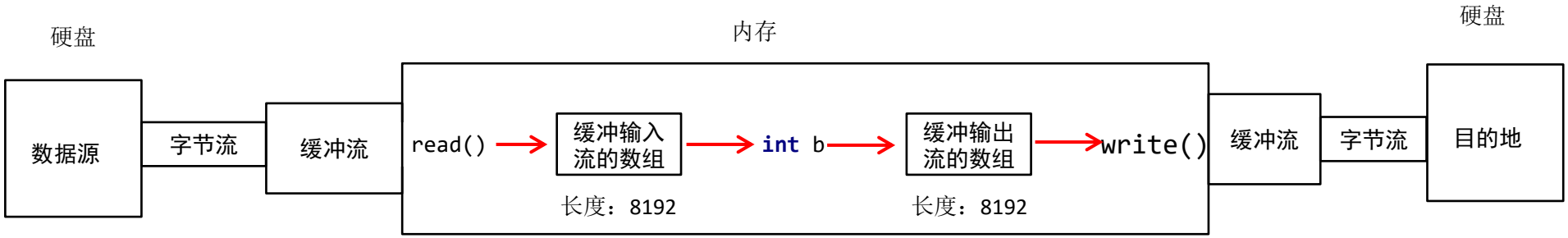

```
int b;
while((b = fis.read())!=-1){
    fos.write(b);
}
```

字节流



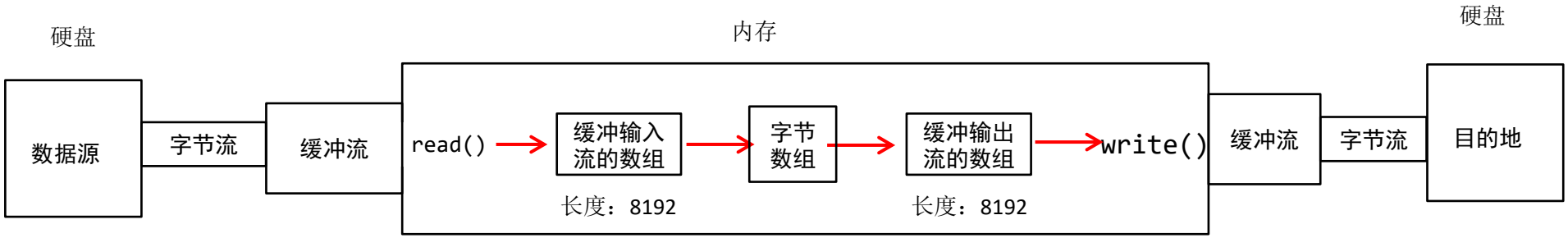

```
byte [] bytes = new byte[1024];
int len;

while((len = fis.read(bytes))!=-1){
    fos.write(bytes, off: 0,len);
}
```

字节流




```
int b;
while ((b = bis.read()) !=-1){
    bos.write(b);
}
```

字节流




```
byte [] bytes = new byte[1024];
int len;
while ((len = bis.read(bytes)) !=-1){
    bos.write(bytes, off: 0,len);
}
```

字节流

案例：复制视频

需求：把“E:\\itheima\\a.avi”复制到模块目录下的“b.avi”

思路：

① 根据数据源创建字节输入流对象

② 根据目的地创建字节输出流对象

③ 读写数据，复制视频

④ 释放资源

小结

字节流：

可以操作所有类型的文件

字节缓冲流：

可以提高读写效率

目录 Contents

思考：既然字节流可以操作所有文件，那么为什么还要学习字符流?

把文件中的数据读取到内存时，如果此时文件中出现了中文，那么字节流就会出现乱码现象。所以纯文本的文件，我们就需要使用字符流来进行操作。

为什么字节流读取纯文本文件，可能会出现乱码?

编码表

基础知识：

- 计算机中储存的信息都是用**二进制**数表示的；我们在屏幕上看到的英文、汉字等字符是二进制数转换之后的结果
- 按照某种规则，将字符存储到计算机中，称为**编码**。
- 按照同样的规则，将存储在计算机中的二进制数解析显示出来，称为**解码** 。
- 编码和解码的方式必须一致，否则会导致乱码。

简单理解：

存储一个字符a，首先需在码表中查到对应的数字是97，然后按照转换成二进制的规则进行存储。

读取的时候，先把二进制解析出来，再转成97，通过97查找到对应的字符是a。

编码表

ASCII字符集：

- **ASCII**(American Standard Code for Information Interchange，美国信息交换标准代码)：包括了数字，大小写字符和一些常见的标点符号。

注意：ASCII码表中是没有中文的。

- **GBK**：window系统默认的码表。兼容ASCII码表，也包含了21003个汉字，并支持繁体汉字以及部分日韩文字。

注意：GBK是中国的码表，一个中文以**两个字节**的形式存储。但不包含世界上所有国家的文字。

编码表

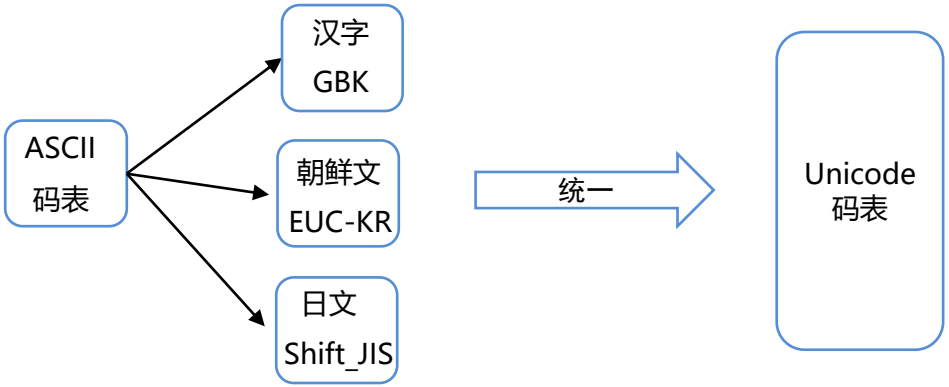
Unicode码表：

由国际组织ISO 制定，是统一的万国码，计算机科学领域里的一项业界标准，容纳世界上大多数国家的所有常见文字和符号。

但是因为表示的字符太多，所以Unicode码表中的数字不是直接以二进制的形式存储到计算机的，会先通过UTF-7，UTF-7.5，UTF-8，UTF-16，以及 UTF-32的编码方式再存储到计算机，其中最为常见的就是UTF-8。

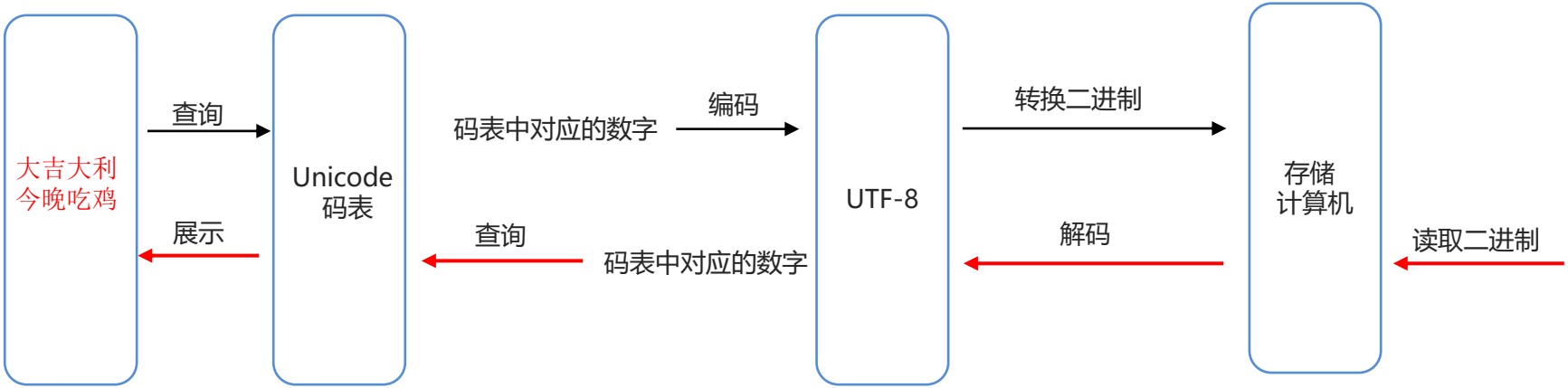
注意： Unicode是万国码，以UTF-8编码后一个中文以**三个字节**的形式存储

编码表小结



字符流

汉字存储和展示过程解析



重点：windows默认使用码表为：**GBK**，一个字符**两**个字节。

idea和以后工作默认使用Unicode的**UTF-8**编解码格式，一个中文**三**个字节。

字符串中的编码解码问题

编码：

- byte[] getBytes()：使用平台的默认字符集将该 String编码为一系列字节，将结果存储到新的字节数组中
- byte[] getBytes(String charsetName)：使用指定的字符集将该 String编码为一系列字节，将结果存储到新的字节数组中

解码：

- String(byte[] bytes)：通过使用平台的默认字符集解码指定的字节数组来构造新的 String
- String(byte[] bytes, String charsetName)：通过指定的字符集解码指定的字节数组来构造新的 String

为什么字节流读取纯文本文件，可能会出现乱码？

因为字节流一次读一个字节，而不管GBK还是UTF-8一个中文都是多个字节，用字节流每次只能读其中的一部分，所以就会出现乱码问题。

abc　　97　98　99

为什么字节流读取纯文本文件，可能会出现乱码？

因为字节流一次读一个字节，而不管GBK还是UTF-8一个中文都是多个字节，用字节流每次只能读其中的一部分，所以就会出现乱码问题。

abc　　97　98　99

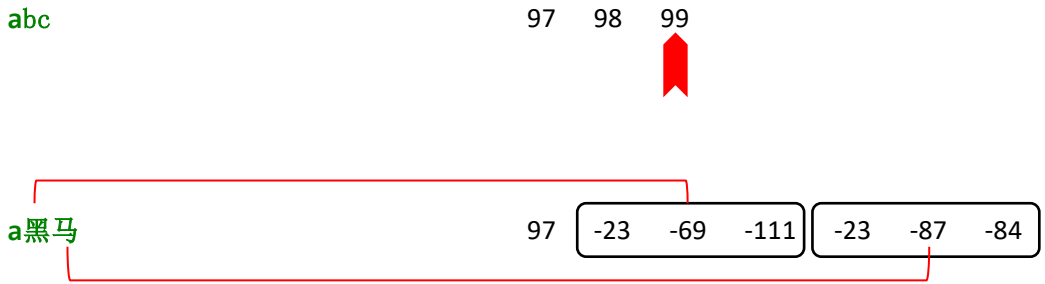
为什么字节流读取纯文本文件，可能会出现乱码？

因为字节流一次读一个字节，而不管GBK还是UTF-8一个中文都是多个字节，用字节流每次只能读其中的一部分，所以就会出现乱码问题。

abc　　97　98　99

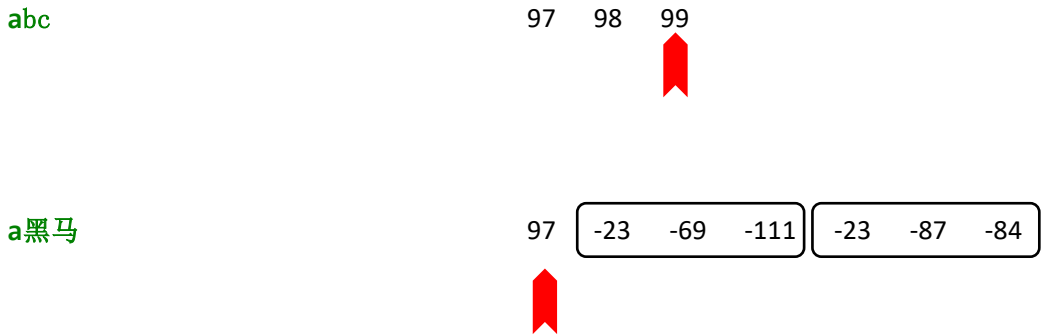
为什么字节流读取纯文本文件，可能会出现乱码？

因为字节流一次读一个字节，而不管GBK还是UTF-8一个中文都是多个字节，用字节流每次只能读其中的一部分，所以就会出现乱码问题。



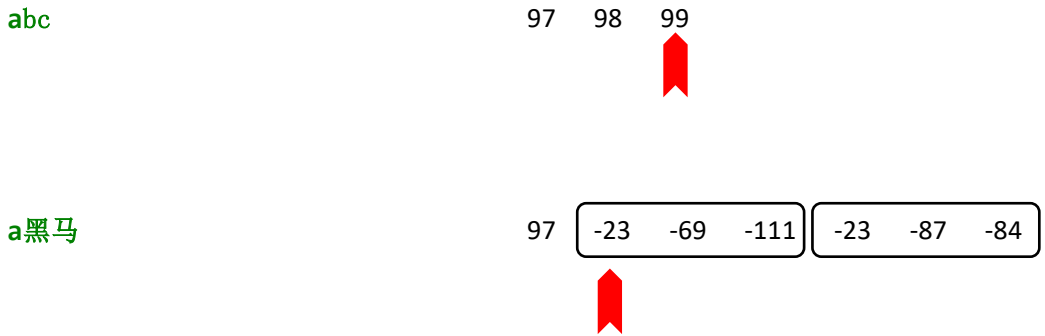
为什么字节流读取纯文本文件，可能会出现乱码？

因为字节流一次读一个字节，而不管GBK还是UTF-8一个中文都是多个字节，用字节流每次只能读其中的一部分，所以就会出现乱码问题。



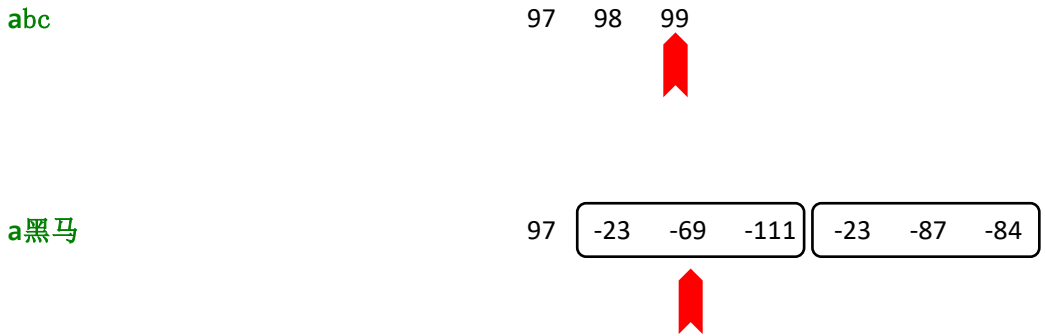
为什么字节流读取纯文本文件，可能会出现乱码？

因为字节流一次读一个字节，而不管GBK还是UTF-8一个中文都是多个字节，用字节流每次只能读其中的一部分，所以就会出现乱码问题。



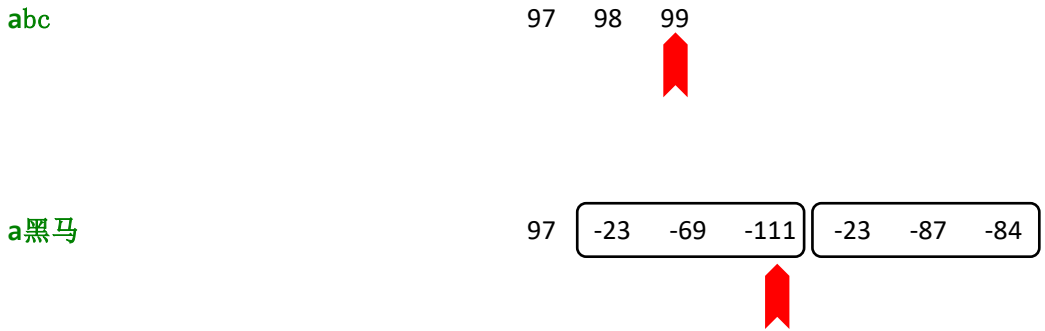
为什么字节流读取纯文本文件，可能会出现乱码？

因为字节流一次读一个字节，而不管GBK还是UTF-8一个中文都是多个字节，用字节流每次只能读其中的一部分，所以就会出现乱码问题。



为什么字节流读取纯文本文件，可能会出现乱码？

因为字节流一次读一个字节，而不管GBK还是UTF-8一个中文都是多个字节，用字节流每次只能读其中的一部分，所以就会出现乱码问题。



为什么字节流读取纯文本文件，可能会出现乱码？

因为字节流一次读一个字节，而不管GBK还是UTF-8一个中文都是多个字节，用字节流每次只能读其中的一部分，所以就会出现乱码问题。

abc　　97　98　99

a黑马　　97　-23　-69　-111　-23　-87　-84

为什么字节流读取纯文本文件，可能会出现乱码？

因为字节流一次读一个字节，而不管GBK还是UTF-8一个中文都是多个字节，用字节流每次只能读其中的一部分，所以就会出现乱码问题。

abc　　97　98　99

a黑马　　97　-23　-69　-111　-23　-87　-84

为什么字节流读取纯文本文件，可能会出现乱码？

因为字节流一次读一个字节，而不管GBK还是UTF-8一个中文都是多个字节，用字节流每次只能读其中的一部分，所以就会出现乱码问题。

abc　　97　98　99

a黑马　　97　-23　-69　-111　-23　-87　-84

结论：

因为字节流读中文，每次只能读一部分所以出现了乱码。

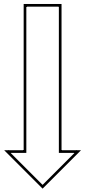
字节流拷贝不会出现乱码的问题?

如果用字节流将文件中的数据读到内存中，打印在控制台会出现乱码。

但是我们发现字节流进行拷贝的时候确不会出现乱码。这是为什么呢?

a黑马

97 -23 -69 -111 -23 -87 -84



97 -23 -69 -111 -23 -87 -84

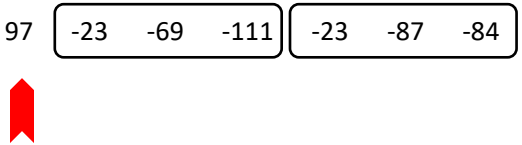
字符流读取中文的过程

- 字符流 = 字节流 + 编码表

基础知识：

不管是在哪张码表中，中文的第一个字节一定是负数。

a黑马



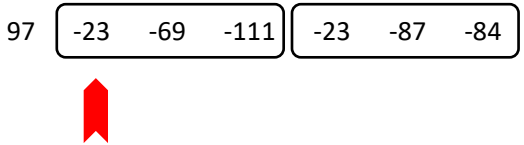
字符流读取中文的过程

- 字符流 = 字节流 + 编码表

基础知识：

不管是在哪张码表中，中文的第一个字节一定是负数。

a黑马



字符流读取中文的过程

- 字符流 = 字节流 + 编码表

基础知识：

不管是在哪张码表中，中文的第一个字节一定是负数。

字符流读取中文的过程

1，想要进行拷贝，一律使用字节流或者字节缓冲流

2，想要把文件中的数据读到内存中打印或者读到内存中运算，请使用字符输入流。
想要把集合，数组，键盘录入等数据写到文件中，请使用字符输出流

3，GBK码表一个中文两个字节，UTF-8编码格式一个中文3个字节。

字符流

字符流写数据

步骤：

① 创建字符输出流对象。

② 写数据

③ 释放资源

字符流

字符流写数据的5种方式

方法名	说明
void write(int c)	写一个字符
void write(char[] cbuf)	写入一个字符数组
void write(char[] cbuf, int off, int len)	写入字符数组的一部分
void write(String str)	写一个字符串
void write(String str, int off, int len)	写一个字符串的一部分

字符流写数据

步骤：

① 创建字符输出流对象。

注意事项：

如果文件不存在，就创建。但是要保证父级路径存在。

如果文件存在就清空。

② 写数据

注意事项：

1，写出int类型的整数，实际写出的是整数在码表上对应的字母。

2，写出字符串数据，是把字符串本身原样写出。

③ 释放资源

注意事项：

每次使用完流必须要释放资源。

字符流写数据的5种方式

方法名	说明
flush()	刷新流，还可以继续写数据
close()	关闭流，释放资源，但是在关闭之前会先刷新流。一旦关闭，就不能再写数据

字符流

字符流读数据的2种方式

方法名	说明
int read()	一次读一个字符数据
int read(char[] cbuf)	一次读一个字符数组数据

案例：保存键盘录入的数据

需求：将用户键盘录入的用户名和密码保存到本地实现永久化存储。

步骤：

① 用户键盘录入用户名

② 将用户名和密码写到本地文件中

字符缓冲流

字符缓冲流：

- BufferedWriter：将文本写入字符输出流，缓冲字符，以提供单个字符，数组和字符串的高效写入，可以指定缓冲区大小，或者可以接受默认大小。默认值足够大，可用于大多数用途
- BufferedReader：从字符输入流读取文本，缓冲字符，以提供字符，数组和行的高效读取，可以指定缓冲区大小，或者可以使用默认大小。 默认值足够大，可用于大多数用途

构造方法：

- BufferedWriter(Writer out)
- BufferedReader(Reader in)

字符缓冲流特有功能

BufferedWriter：

- void newLine()：写一行行分隔符，行分隔符字符串由系统属性定义

BufferedReader：

- public String readLine()：读一行文字。 结果包含行的内容的字符串，不包括任何行终止字符，如果流的结尾已经到达，则为null

案例：读取文件中的数据排序后再次写到本地

需求：读取文件中的数据，排序后再次写到本地文件

步骤：

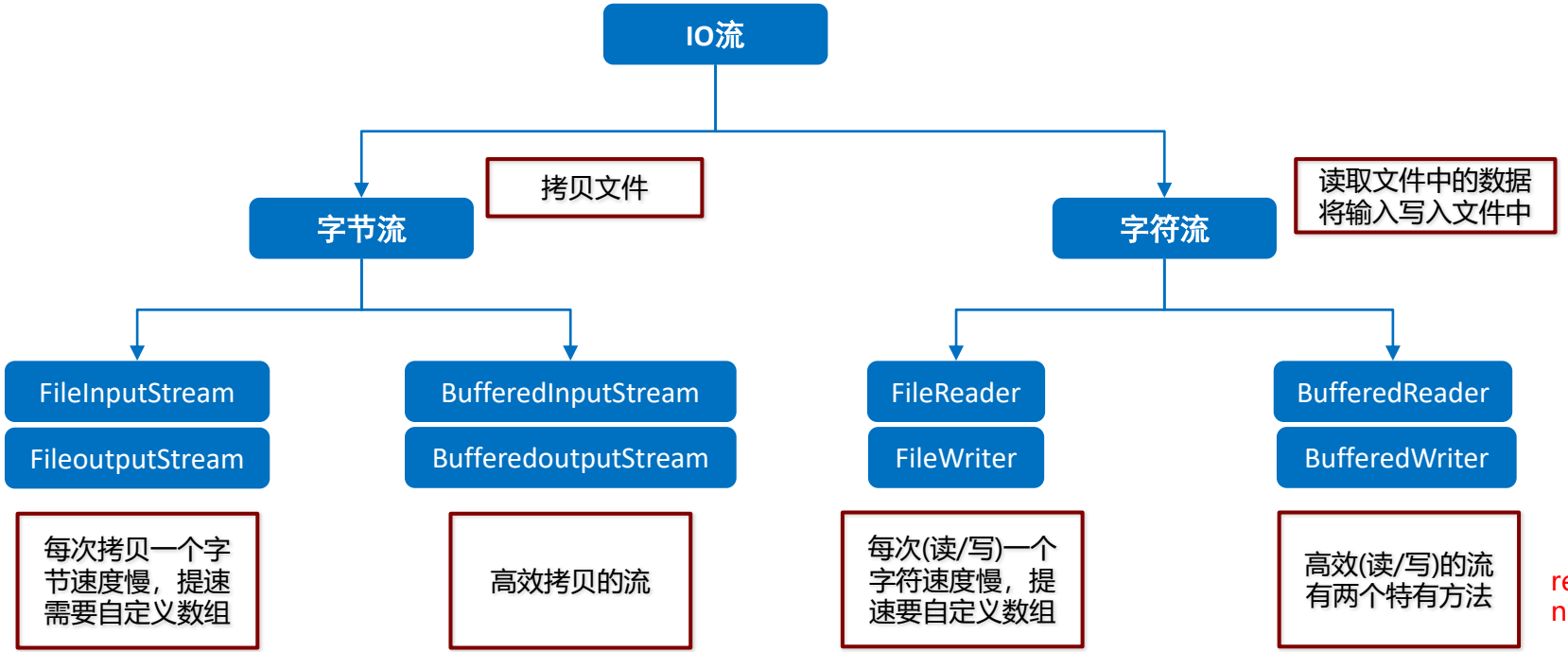
① 读取数据

② 将数据排序

③ 写回本地

字符流

IO流小结



复制文件的异常处理

基本做法：

```
try{
    可能出现异常的代码;
}catch(异常类名 变量名){
    异常的处理代码;
}finally{
    执行所有清除操作;
}
```

手动释放资源

JDK7改进方案：

```
try(定义流对象){
    可能出现异常的代码;
}catch(异常类名 变量名){
    异常的处理代码;
}
```

自动释放资源

JDK9改进方案：

```
定义输入流对象;
定义输出流对象;
try(输入流对象;输出流对象){
    可能出现异常的代码;
}catch(异常类名 变量名){
    异常的处理代码;
}
```

自动释放资源