# 字节缓冲流

## 1.1字节缓冲区流的概述和使用

### 1.1.1 字节缓冲流的作用是?

字节流一次读写一个数组的速度比一次读写一个字节的速度快很多，这是加入了数组这样的缓冲区效果，java本身在设计的时候，也考虑到了这样的设计思想，所以提供了字节缓冲区流

字节缓冲流 :

BufferedOutputStream:字节缓冲输出流

BufferedInputStream:字节缓冲输入流

### 1.1.2 为什么字节缓冲流的构造方法需要传入一个OutputStream

字节缓冲区流仅仅提供缓冲区，而真正的底层的读写数据还得需要基本的流对象进行操作。

### 1.1.3 案例代码

**public** **class** BufferedStreamDemo {

**public** **static** **void** main(String[] args) **throws** IOException {

// BufferedOutputStream(OutputStream out)

// FileOutputStream fos = new FileOutputStream("a.txt");

// BufferedOutputStream bos = new BufferedOutputStream(fos);

// 上面的两句等价于下面的这一句

// BufferedOutputStream bos = new BufferedOutputStream(new

// FileOutputStream("a.txt"));

// bos.write("hello".getBytes());

// bos.close();

// BufferedInputStream(InputStream in)

BufferedInputStream bis = **new** BufferedInputStream(**new** FileInputStream("a.txt"));

//方式1：一次读取一个字节

// int by;

// while((by=bis.read())!=-1) {

// System.out.print((char)by);

// }

//方式2：一次读取一个字节数组

**byte**[] bys = **new** **byte**[1024];

**int** len;

**while**((len=bis.read(bys))!=-1) {

System.***out***.print(**new** String(bys,0,len));

}

bis.close();

}

}

## 1.2 字节流四种方式复制AVI并测试效率

### 1.2.1 方法摘要

public static long currentTimeMillis():返回以毫秒为单位的当前时间。

### 1.2.2 案例代码

**public** **class** CopyAviTest {

**public** **static** **void** main(String[] args) **throws** IOException {

//记录开始时间

**long** start = System.*currentTimeMillis*();

// method1();

// method2();

// method3();

*method4*();

//记录结束时间

**long** end = System.*currentTimeMillis*();

System.***out***.println("共耗时："+(end-start)+"毫秒");

}

//缓冲字节流一次读写一个字节数组

**private** **static** **void** method4() **throws** IOException {

//封装数据源

BufferedInputStream bis = **new** BufferedInputStream(**new** FileInputStream("d:\\复制图片.avi"));

//封装目的地

BufferedOutputStream bos = **new** BufferedOutputStream(**new** FileOutputStream("copy.avi"));

**byte**[] bys = **new** **byte**[1024];

**int** len;

**while**((len=bis.read(bys))!=-1) {

bos.write(bys,0,len);

}

bos.close();

bis.close();

}

//缓冲字节流一次读写一个字节

**private** **static** **void** method3() **throws** IOException {

//封装数据源

BufferedInputStream bis = **new** BufferedInputStream(**new** FileInputStream("d:\\复制图片.avi"));

//封装目的地

BufferedOutputStream bos = **new** BufferedOutputStream(**new** FileOutputStream("copy.avi"));

**int** by;

**while**((by=bis.read())!=-1) {

bos.write(by);

}

bos.close();

bis.close();

}

//基本字节流一次读写一个字节数组

**private** **static** **void** method2() **throws** IOException {

//封装数据源

FileInputStream fis = **new** FileInputStream("d:\\复制图片.avi");

//封装目的地

FileOutputStream fos = **new** FileOutputStream("copy.avi");

**byte**[] bys = **new** **byte**[1024];

**int** len;

**while**((len=fis.read(bys))!=-1) {

fos.write(bys,0,len);

}

fos.close();

fis.close();

}

//基本字节流一次读写一个字节

**private** **static** **void** method1() **throws** IOException {

//封装数据源

FileInputStream fis = **new** FileInputStream("d:\\复制图片.avi");

//封装目的地

FileOutputStream fos = **new** FileOutputStream("copy.avi");

**int** by;

**while**((by=fis.read())!=-1) {

fos.write(by);

}

fos.close();

fis.close();

}

}

# 转换流

## 2.1 转换流出现的原因

### 2.1.1字节流读数据可能出现问题

字节流一次读取一个字节的方式读取带有汉字的文件是有问题的，因为你读取到一个字节后就转为字符在控制台输出了，

而汉字是由2个字节组成的，所以这里会出问题。

文件复制的时候，字节流读取一个字节，写入一个字节，这个没有出现问题，是因为最终底层会根据字节做拼接，

得到汉字。

汉字存储的规则：

左边的字节数据肯定是负数，右边的字节数据可能是负数，也可能是正数，大部分情况下是负数。

### 2.1.2 案例代码

**public** **class** FileInputStreamDemo {

**public** **static** **void** main(String[] args) **throws** IOException {

//基本字节流一次读取一个字节

// FileInputStream fis = new FileInputStream("a.txt");

//

// int by;

// while((by=fis.read())!=-1) {

// System.out.print((char)by);

// }

//

// fis.close();

//String s = "hello";

//[104, 101, 108, 108, 111]

String s = "你好";

//[-60, -29, -70, -61]

**byte**[] bys = s.getBytes();

System.***out***.println(Arrays.*toString*(bys));

}

}

### 2.1.3 转换流的组成部分

转换流 = 字节流 + 编码表

这句话怎么理解 ? 往下看, 先说编码表

## 2.2 编码表概述和常见编码表

### 2.2.1 什么是编码表?

编码表:

由字符及其对应的数据组成的一张表

ASCII:

‘a’ 97

‘A’ 65

‘0’ 48

常见的编码表:

ASCII : 美国标准信息交换码, 用一个字节的7位表示数据

ISO-8859-1 : 欧洲码表, 用一个字节的8位表示数据, 兼容ASCII

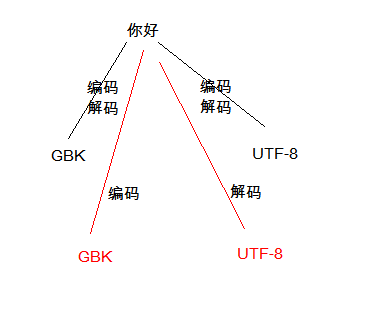
GB2312 : 中文码表的升级版, 融合了更多的中文文字符号, 兼容ASCII

UTF-8 : 是一种可变长度的字符编码, 用1-3个字节表示数据, 又称为万国码, 兼容ASCII

用在网页上可以统一页面中的中文简体繁体和其他语言的显示.

### 2.2.2 乱码问题

针对同一个数据, 采用的编码和解码不一致导致



## 2.3 String类中的编码和解码问题

### 2.3.1 方法摘要&编码和解码

* 编码

把看得懂的变成看不懂的

public byte[] getBytes(String charsetName) throws UnsupportedEncodingException

使用指定的字符集将此 String 编码为 byte 序列，并将结果存储到一个新的 byte 数组中。

* 解码

把看不懂的变成看得懂的

public String(byte[] bytes, String charsetName)

通过使用指定的 charset解码指定的 byte 数组，构造一个新的 String。

重点强调 : 编码和解码的方式需要一致

### 2.3.2 案例代码

**public** **class** StringDemo {

**public** **static** **void** main(String[] args) **throws** UnsupportedEncodingException {

//定义一个字符串

String s = "你好";

//编码

//byte[] bys = s.getBytes();//使用平台的默认字符集将此 String 编码为 byte 序列

//默认编码是GBK

//[-60, -29, -70, -61]

//byte[] bys = s.getBytes("GBK"); //指定编码GBK

//[-60, -29, -70, -61]

**byte**[] bys = s.getBytes("UTF-8"); //指定编码UTF-8

//[-28, -67, -96, -27, -91, -67]

System.***out***.println(Arrays.*toString*(bys));

//解码

//String ss = new String(bys); //通过使用平台的默认字符集解码指定的 byte 数组

// String ss = new String(bys,"GBK");//指定编码GBK

String ss = **new** String(bys,"UTF-8");//指定编码UTF-8

System.***out***.println(ss);

}

}

## 2.4 转换流中的编码和解码问题

### 2.4.1 转换流指的是?

转换流其实就是一个字符流。

转换流 = 字节流 + 编码表

### 2.4.2 转换流的构造方法

OutputStreamWriter 字符输出流

public OutputStreamWriter(OutputStream out)

根据默认编码把字节流的数据转换为字符流

public OutputStreamWriter(OutputStream out,String charsetName)

根据指定编码把字节流数据转换为字符流

InputStreamReader 字符输入流

public InputStreamReader(InputStream in)

用默认的编码读数据

public InputStreamReader(InputStream in,String charsetName)

### 2.4.3 案例代码

**public** **class** ConversionStreamDemo {

**public** **static** **void** main(String[] args) **throws** IOException {

//public OutputStreamWriter(OutputStream out):默认编码GBK

//OutputStreamWriter osw = new OutputStreamWriter(new FileOutputStream("osw.txt"));

//public OutputStreamWriter(OutputStream out,String charsetName)

//OutputStreamWriter osw = new OutputStreamWriter(new FileOutputStream("osw.txt"),"GBK");

OutputStreamWriter osw = **new** OutputStreamWriter(**new** FileOutputStream("osw.txt"),"UTF-8");

//调用写数据的方法

osw.write("你好");

//释放资源

osw.close();

System.***out***.println("------------------------");

//public InputStreamReader(InputStream in):默认编码GBK

//InputStreamReader isr = new InputStreamReader(new FileInputStream("osw.txt"));

//public InputStreamReader(InputStream in,String charsetName)

//InputStreamReader isr = new InputStreamReader(new FileInputStream("osw.txt"),"GBK");

InputStreamReader isr = **new** InputStreamReader(**new** FileInputStream("osw.txt"),"UTF-8");

//读数据：一次读取一个字符数据

**int** ch;

**while**((ch=isr.read())!=-1) {

System.***out***.print((**char**)ch);

}

//释放资源

isr.close();

}

}

## 2.5 OutputStreamWriter写数据的5种方式

### 2.5.1 方法摘要

\* OutputStreamWriter写数据方法

\* public void write(int c):写一个字符

\* public void write(char[] cbuf):写一个字符数组

\* public void write(char[] cbuf,int off,int len):写一个字符数组的一部分

\* public void write(String str):写一个字符串

\* public void write(String str,int off,int len):写一个字符串的一部分

### 2.5.2 案例代码

**public** **class** OutputStreamWriterDemo {

**public** **static** **void** main(String[] args) **throws** IOException {

//创建字符输出流对象

OutputStreamWriter osw = **new** OutputStreamWriter(**new** FileOutputStream("b.txt"));

//public void write(int c):写一个字符

// osw.write(97);

// osw.write('a');

//写完数据后，没有发现数据，为什么呢?

//1字符=2字节

//文件中的数据存储的基本单位是字节

//public void write(char[] cbuf):写一个字符数组

// char[] chs = {'a','b','c','d','e'};

// osw.write(chs);

//public void write(char[] cbuf,int off,int len):写一个字符数组的一部分

// char[] chs = {'a','b','c','d','e'};

// osw.write(chs, 1, 3);

//public void write(String str):写一个字符串

// osw.write("hello");

//public void write(String str,int off,int len):写一个字符串的一部分

osw.write("hello", 0, 3);

// //void flush():刷新该流的缓冲

// osw.flush();

//

// //释放资源

osw.close(); //关闭此流，但要先刷新它

}

}

## 2.6 InputStreamReader读数据的2种方式

### 2.6.1 方法摘要

\* public int read():一次读取一个字符

\* public int read(char[] cbuf):一次读取一个字符数组

### 2.6.2 案例代码

**public** **class** InputStreamReaderDemo {

**public** **static** **void** main(String[] args) **throws** IOException {

//创建字符输入流对象

// InputStreamReader isr = new InputStreamReader(new FileInputStream("a.txt"));

InputStreamReader isr = **new** InputStreamReader(**new** FileInputStream("OutputStreamWriterDemo.java"));

//public int read():一次读取一个字符

// int ch;

// while((ch=isr.read())!=-1) {

// System.out.print((char)ch);

// }

//public int read(char[] cbuf):一次读取一个字符数组

**char**[] chs = **new** **char**[1024];

**int** len;

**while**((len=isr.read(chs))!=-1) {

System.***out***.print(**new** String(chs,0,len));

}

//释放资源

isr.close();

}

}

# 字符流

## 字符流的练习之复制Java文件

### 案例代码

/\*

\* 把当前项目目录下的StringDemo.java内容复制到当前项目目录下的Copy.java中

\*

\* 数据源：

\* StringDemo.java---读数据---字符流---InputStreamReader

\* 目的地：

\* Copy.java---写数据---字符流---OutputStreamWriter

\*/

**public** **class** CopyJavaTest {

**public** **static** **void** main(String[] args) **throws** IOException {

//封装数据源

InputStreamReader isr = **new** InputStreamReader(**new** FileInputStream("StringDemo.java"));

//封装目的地

OutputStreamWriter osw = **new** OutputStreamWriter(**new** FileOutputStream("Copy.java"));

//读写数据

//方式1：一次读写一个字符

// int ch;

// while((ch=isr.read())!=-1) {

// osw.write(ch);

// }

//方式2：一次读写一个字符数组

**char**[] chs = **new** **char**[1024];

**int** len;

**while**((len=isr.read(chs))!=-1) {

osw.write(chs, 0, len);

}

//释放资源

osw.close();

isr.close();

}

}

## 字符流的练习之复制Java文件改进版

### 改进原因

转换流的名字比较长，而我们常见的操作都是按照本地默认编码实现的，所以，为了简化我们的书写，转换流提供了对应的子类。

FileWriter

FileReader

### 3.2.2 案例代码

**public** **class** CopyJavaTest2 {

**public** **static** **void** main(String[] args) **throws** IOException {

//封装数据源

FileReader fr = **new** FileReader("StringDemo.java");

//封装目的地

FileWriter fw = **new** FileWriter("Copy.java");

//读写数据

//一次读写一个字符

// int ch;

// while((ch=fr.read())!=-1) {

// fw.write(ch);

// }

//一次读写一个字符数组

**char**[] chs = **new** **char**[1024];

**int** len;

**while**((len=fr.read(chs))!=-1) {

fw.write(chs, 0, len);

}

//释放资源

fw.close();

fr.close();

}

}

## 3.3 字符缓冲区流的概述和使用

### 3.3.1字符缓冲区流的概述

BufferedWriter

将文本写入字符输出流，缓冲各个字符，从而提供单个字符、数组和字符串的高效写入。

可以指定缓冲区的大小，或者接受默认的大小。在大多数情况下，默认值就足够大了。

构造方法：

BufferedWriter(Writer out)

BufferedReader

从字符输入流中读取文本，缓冲各个字符，从而实现字符、数组和行的高效读取。

可以指定缓冲区的大小，或者可使用默认的大小。大多数情况下，默认值就足够大了。

构造方法：

BufferedReader(Reader in)

### 3.3.2 案例代码

**public** **class** BufferedStreamDemo {

**public** **static** **void** main(String[] args) **throws** IOException {

// //创建字符缓冲输出流对象

// BufferedWriter bw = new BufferedWriter(new FileWriter("a.txt"));

// //调用写数据的方法

// bw.write("hello");

// //释放资源

// bw.close();

//创建字符缓冲输入流对象

BufferedReader br = **new** BufferedReader(**new** FileReader("BufferedStreamDemo.java"));

//方式1：一次读取一个字符

// int ch;

// while((ch=br.read())!=-1) {

// System.out.print((char)ch);

// }

//方式2：一次读取一个字符数组

**char**[] chs = **new** **char**[1024];

**int** len;

**while**((len=br.read(chs))!=-1) {

System.***out***.print(**new** String(chs,0,len));

}

//释放资源

br.close();

}

}

## 3.4 字符缓冲区流的练习之复制文本文件

### 3.4.1 案例代码

/\*

\* 把项目目录下的a.txt内容复制到项目目录下的b.txt中

\*

\* 数据源：

\* a.txt---读数据---字符流---InputStreamReader---FileReader---BufferedReader

\* 目的地：

\* b.txt---写数据---字符流---OutputStreamWriter---FileWriter---BufferedWriter

\*/

**public** **class** CopyTxtTest {

**public** **static** **void** main(String[] args) **throws** IOException {

//封装数据源

BufferedReader br = **new** BufferedReader(**new** FileReader("a.txt"));

//封装目的地

BufferedWriter bw = **new** BufferedWriter(**new** FileWriter("b.txt"));

//读写数据

//一次读写一个字符数组

**char**[] chs = **new** **char**[1024];

**int** len;

**while**((len=br.read(chs))!=-1) {

bw.write(chs, 0, len);

}

//释放资源

bw.close();

br.close();

}

}

## 3.5 字符缓冲区流的特殊功能

### 3.5.1 方法摘要

\* BufferedWriter

\* void newLine():写入一个行分隔符,这个行分隔符是由系统决定的

\*

\* BufferedReader

\* String readLine():包含该行内容的字符串，不包含任何行终止符，如果已到达流末尾，则返回 null

### 3.5.2 案例代码

**public** **class** BufferedStreamDemo {

**public** **static** **void** main(String[] args) **throws** IOException {

// //创建字符缓冲输出流对象

// BufferedWriter bw = new BufferedWriter(new FileWriter("bw.txt"));

//

// //写数据

// for(int x=0; x<3; x++) {

// bw.write("hello");

//// bw.write("\r\n");

// bw.newLine();

// bw.flush();

// }

//

// //释放资源

// bw.close();

//创建字符缓冲输入流对象

BufferedReader br = **new** BufferedReader(**new** FileReader("bw.txt"));

// //读取一次

// String line = br.readLine();

// System.out.println(line);

// //在读取一次

// line = br.readLine();

// System.out.println(line);

// line = br.readLine();

// System.out.println(line);

// //多读取两次

// line = br.readLine();

// System.out.println(line);

// line = br.readLine();

// System.out.println(line);

//最终版代码

String line;

**while**((line=br.readLine())!=**null**) {

System.***out***.println(line);

}

//释放资源

br.close();

}

}

## 3.6字符缓冲区流的特殊功能复制Java文件

### 3.6.1 案例代码

/\*

\* 字符缓冲流特殊功能复制Java文件

\*

\* 数据源：

\* BufferedStreamDemo.java---BufferedReader

\* 目的地：

\* Copy.java---BufferedWriter

\*/

**public** **class** CopyJavaTest {

**public** **static** **void** main(String[] args) **throws** IOException {

//封装数据源

BufferedReader br = **new** BufferedReader(**new** FileReader("BufferedStreamDemo.java"));

//封装目的地

BufferedWriter bw = **new** BufferedWriter(**new** FileWriter("Copy.java"));

//读写数据

String line;

**while**((line=br.readLine())!=**null**) {

bw.write(line);

bw.newLine();

bw.flush();

}

//释放资源

bw.close();

br.close();

}

}

# 字符流相关练习

## 4.1字符流的练习之5种方式复制文本文件

### 4.1.1 字符流复制文本文件的五种方式分别是?

基本字符流一次读写一个字符

基本字符流一次读写一个字符数组

缓冲字符流一次读写一个字符

缓冲字符流一次读写一个字符数组

缓冲字符串一次读写一个字符串

### 4.1.2 案例代码

**public** **class** CopyFileTest {

**public** **static** **void** main(String[] args) **throws** IOException {

*method1*();

// method2();

// method3();

// method4();

// method5();

}

//缓冲字符流一次读写一个字符串

**private** **static** **void** method5() **throws** IOException {

BufferedReader br = **new** BufferedReader(**new** FileReader("d:\\林青霞.txt"));

BufferedWriter bw = **new** BufferedWriter(**new** FileWriter("窗里窗外.txt"));

String line;

**while**((line=br.readLine())!=**null**) {

bw.write(line);

bw.newLine();

bw.flush();

}

bw.close();

br.close();

}

//缓冲字符流一次读写一个字符数组

**private** **static** **void** method4() **throws** IOException {

BufferedReader br = **new** BufferedReader(**new** FileReader("d:\\林青霞.txt"));

BufferedWriter bw = **new** BufferedWriter(**new** FileWriter("窗里窗外.txt"));

**char**[] chs = **new** **char**[1024];

**int** len;

**while**((len=br.read(chs))!=-1) {

bw.write(chs, 0, len);

}

bw.close();

br.close();

}

//缓冲字符流一次读写一个字符

**private** **static** **void** method3() **throws** IOException {

BufferedReader br = **new** BufferedReader(**new** FileReader("d:\\林青霞.txt"));

BufferedWriter bw = **new** BufferedWriter(**new** FileWriter("窗里窗外.txt"));

**int** ch;

**while**((ch=br.read())!=-1) {

bw.write(ch);

}

bw.close();

br.close();

}

// 基本字符流一次读写一个字符数组

**private** **static** **void** method2() **throws** IOException {

FileReader fr = **new** FileReader("d:\\林青霞.txt");

FileWriter fw = **new** FileWriter("窗里窗外.txt");

**char**[] chs = **new** **char**[1024];

**int** len;

**while**((len=fr.read(chs))!=-1) {

fw.write(chs, 0, len);

}

fw.close();

fr.close();

}

// 基本字符流一次读写一个字符

**private** **static** **void** method1() **throws** IOException {

FileReader fr = **new** FileReader("d:\\林青霞.txt");

FileWriter fw = **new** FileWriter("窗里窗外.txt");

**int** ch;

**while** ((ch = fr.read()) != -1) {

fw.write(ch);

}

fw.close();

fr.close();

}

}

## 4.2 字符流的练习之把集合中的字符串数据存储到文本文件

### 4.2.1 案例分析:

A:创建集合对象

B:往集合中添加字符串元素

C:创建字符缓冲输出流对象

D:遍历集合，得到每一个字符串元素，把字符串元素作为数据写入到文本文件

E:释放资源

### 4.2.2 案例代码

**public** **class** ArrayListToFileTest {

**public** **static** **void** main(String[] args) **throws** IOException {

//创建集合对象

ArrayList<String> array = **new** ArrayList<String>();

//往集合中添加字符串元素

array.add("hello");

array.add("world");

array.add("java");

//创建字符缓冲输出流对象

BufferedWriter bw = **new** BufferedWriter(**new** FileWriter("array.txt"));

//遍历集合，得到每一个字符串元素，把字符串元素作为数据写入到文本文件

**for**(String s : array) {

bw.write(s);

bw.newLine();

bw.flush();

}

//释放资源

bw.close();

}

}

## 4.3 字符流的练习之把文本文件中的字符串数据读取到集合

### 4.3.1 案例分析

需求: 从文本文件中读取数据到ArrayList集合中，并遍历集合

每一行数据作为一个字符串元素

分析:

A:创建字符缓冲输入流对象

B:创建集合对象

C:读取数据，每一次读取一行，并把该数据作为元素存储到集合中

D:释放资源

E:遍历集合

### 4.3.2 案例代码

**public** **class** FileToArrayListTest {

**public** **static** **void** main(String[] args) **throws** IOException {

//创建字符缓冲输入流对象

BufferedReader br = **new** BufferedReader(**new** FileReader("array.txt"));

//创建集合对象

ArrayList<String> array = **new** ArrayList<String>();

//读取数据，每一次读取一行，并把该数据作为元素存储到集合中

String line;

**while**((line=br.readLine())!=**null**) {

array.add(line);

}

//释放资源

br.close();

//遍历集合

**for**(String s : array) {

System.***out***.println(s);

}

}

}

## 4.4 字符流的练习之把集合中的学生对象数据存储到文本文件

### 4.4.1案例分析

需求:

把ArrayList集合中的学生数据存储到文本文件

每一个学生数据作为文件中的一行数据

定义一个学生类。

学生：

学号，姓名，年龄，所在城市

it001,张曼玉,35,北京

it002,王祖贤,33,上海

it003,林青霞,30,西安

分析:

A:创建集合对象

B:创建学生对象

C:把学生对象添加到集合中

D:创建字符缓冲输出流对象

E:遍历集合，得到每一个学生对象，然后把该对象的数据拼接成一个指定格式的字符串写到文本文件

F:释放资源

### 4.4.2 案例代码

**public** **class** ArrayListToFileTest {

**public** **static** **void** main(String[] args) **throws** IOException {

// 创建集合对象

ArrayList<Student> array = **new** ArrayList<Student>();

// 创建学生对象

Student s1 = **new** Student("it001", "张曼玉", 35, "北京");

Student s2 = **new** Student("it002", "王祖贤", 33, "上海");

Student s3 = **new** Student("it003", "林青霞", 30, "西安");

// 把学生对象添加到集合中

array.add(s1);

array.add(s2);

array.add(s3);

// 创建字符缓冲输出流对象

BufferedWriter bw = **new** BufferedWriter(**new** FileWriter("students.txt"));

// 遍历集合，得到每一个学生对象，然后把该对象的数据拼接成一个指定格式的字符串写到文本文件

**for** (Student s : array) {

// it001,张曼玉,35,北京

StringBuilder sb = **new** StringBuilder();

sb.append(s.getSid()).append(",").append(s.getName()).append(",").append(s.getAge()).append(",")

.append(s.getCity());

bw.write(sb.toString());

bw.newLine();

bw.flush();

}

//释放资源

bw.close();

}

}

## 4.5 字符流的练习之把文本文件中的学生对象数据读取到集合

### 4.5.1 案例分析

需求:

从文本文件中读取学生数据到ArrayList集合中，并遍历集合

每一行数据作为一个学生元素

it001,张曼玉,35,北京

这里我们要使用String类中的一个方法：split()

分析:

A:创建字符缓冲输入流对象

B:创建集合对象

C:读取数据，每一次读取一行数据，把该行数据想办法封装成学生对象，并把学生对象存储到集合中

D:释放资源

E:遍历集合

### 4.5.2 案例代码

**public** **class** FileToArrayListTest {

**public** **static** **void** main(String[] args) **throws** IOException {

// 创建字符缓冲输入流对象

BufferedReader br = **new** BufferedReader(**new** FileReader("students.txt"));

// 创建集合对象

ArrayList<Student> array = **new** ArrayList<Student>();

// 读取数据，每一次读取一行数据，把该行数据想办法封装成学生对象，并把学生对象存储到集合中

String line;

**while** ((line = br.readLine()) != **null**) {

// it001,张曼玉,35,北京

String[] strArray = line.split(",");

Student s = **new** Student();

s.setSid(strArray[0]);

s.setName(strArray[1]);

s.setAge(Integer.*parseInt*(strArray[2]));

s.setCity(strArray[3]);

array.add(s);

}

// 释放资源

br.close();

// 遍历集合

**for** (Student s : array) {

System.***out***.println(s.getSid() + "---" + s.getName() + "---" + s.getAge() + "---" + s.getCity());

}

}

}