1.字节缓冲流

1.1字节缓冲流构造方法【应用】

- 字节缓冲流介绍
  - lBufferOutputStream：该类实现缓冲输出流。 通过设置这样的输出流，应用程序可以向底层输出流写入字节，而不必为写入的每个字节导致底层系统的调用
  - lBufferedInputStream：创建BufferedInputStream将创建一个内部缓冲区数组。 当从流中读取或跳过字节时，内部缓冲区将根据需要从所包含的输入流中重新填充，一次很多字节
- 构造方法：

方法名	说明
BufferedOutputStream(OutputStream out)	创建字节缓冲输出流对象
BufferedInputStream(InputStream in)	创建字节缓冲输入流对象

- 示例代码

```
public class BufferStreamDemo {
    public static void main(String[] args) throws IOException {
        //字节缓冲输出流：BufferedOutputStream(OutputStream out)

        BufferedOutputStream bos = new BufferedOutputStream(new
                                    FileOutputStream("myByteStream\\bos.txt"));
        //写数据
        bos.write("hello\r\n".getBytes());
        bos.write("world\r\n".getBytes());
        //释放资源
        bos.close();


        //字节缓冲输入流：BufferedInputStream(InputStream in)
        BufferedInputStream bis = new BufferedInputStream(new
                                    FileInputStream("myByteStream\\bos.txt"));

        //一次读取一个字节数据
//        int by;
//        while ((by=bis.read())!=-1) {
//            System.out.print((char)by);
//        }

        //一次读取一个字节数组数据
        byte[] bys = new byte[1024];
        int len;
        while ((len=bis.read(bys))!=-1) {
            System.out.print(new String(bys,0,len));
```

```
        }

        //释放资源
        bis.close();
    }
}
```

1.2字节流复制视频【应用】

- 案例需求

  把“E:\itcast\字节流复制图片.avi”复制到模块目录下的“字节流复制图片.avi”

- 实现步骤
  - 根据数据源创建字节输入流对象
  - 根据目的地创建字节输出流对象
  - 读写数据，复制视频
  - 释放资源
- 代码实现

```
public class CopyAviDemo {
    public static void main(String[] args) throws IOException {
        //记录开始时间
        long startTime = System.currentTimeMillis();

        //复制视频
//        method1();
//        method2();
//        method3();
        method4();

        //记录结束时间
        long endTime = System.currentTimeMillis();
        System.out.println("共耗时: " + (endTime - startTime) + "毫秒");
    }

    //字节缓冲流一次读写一个字节数组
    public static void method4() throws IOException {
        BufferedInputStream bis = new BufferedInputStream(new
FileInputStream("E:\\itcast\\字节流复制图片.avi"));
        BufferedOutputStream bos = new BufferedOutputStream(new
FileOutputStream("myByteStream\\字节流复制图片.avi"));

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

        bos.close();
        bis.close();
    }
```

```
    //字节缓冲流一次读写一个字节
    public static void method3() throws IOException {
        BufferedInputStream bis = new BufferedInputStream(new
FileInputStream("E:\\itcast\\字节流复制图片.avi"));
        BufferedOutputStream bos = new BufferedOutputStream(new
FileOutputStream("myByteStream\\字节流复制图片.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
    public static void method2() throws IOException {
        //E:\\itcast\\字节流复制图片.avi
        //模块目录下的 字节流复制图片.avi
        FileInputStream fis = new FileInputStream("E:\\itcast\\字节流复制图片.avi");
        FileOutputStream fos = new FileOutputStream("myByteStream\\字节流复制图
片.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
    public static void method1() throws IOException {
        //E:\\itcast\\字节流复制图片.avi
        //模块目录下的 字节流复制图片.avi
        FileInputStream fis = new FileInputStream("E:\\itcast\\字节流复制图片.avi");
        FileOutputStream fos = new FileOutputStream("myByteStream\\字节流复制图
片.avi");

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

        fos.close();
        fis.close();
    }
}
```

2.字符流

2.1为什么会出现字符流【理解】

- 字符流的介绍

  由于字节流操作中文不是特别的方便，所以Java就提供字符流

  字符流 = 字节流 + 编码表

- 中文的字节存储方式

  用字节流复制文本文件时，文本文件也会有中文，但是没有问题，原因是最终底层操作会自动进行字节拼接成中文，如何识别是中文的呢?

  汉字在存储的时候，无论选择哪种编码存储，第一个字节都是负数

2.2编码表【理解】

- 什么是字符集

  是一个系统支持的所有字符的集合，包括各国家文字、标点符号、图形符号、数字等

  l计算机要准确的存储和识别各种字符集符号，就需要进行字符编码，一套字符集必然至少有一套字符编码。常见字符集有ASCII字符集、GBXXX字符集、Unicode字符集等

- 常见的字符集
  - ASCII字符集：

    lASCII：是基于拉丁字母的一套电脑编码系统，用于显示现代英语，主要包括控制字符(回车键、退格、换行键等)和可显示字符(英文大小写字符、阿拉伯数字和西文符号)

    基本的ASCII字符集，使用7位表示一个字符，共128字符。ASCII的扩展字符集使用8位表示一个字符，共256字符，方便支持欧洲常用字符。是一个系统支持的所有字符的集合，包括各国家文字、标点符号、图形符号、数字等

  - GBXXX字符集：

    GBK：最常用的中文码表。是在GB2312标准基础上的扩展规范，使用了双字节编码方案，共收录了21003个汉字，完全兼容GB2312标准，同时支持繁体汉字以及日韩汉字等

  - Unicode字符集：

    UTF-8编码：可以用来表示Unicode标准中任意字符，它是电子邮件、网页及其他存储或传送文字的应用中，优先采用的编码。互联网工程工作小组（IETF）要求所有互联网协议都必须支持UTF-8编码。它使用一至四个字节为每个字符编码

    编码规则：

    128个US-ASCII字符，只需一个字节编码

    拉丁文等字符，需要二个字节编码

    大部分常用字（含中文），使用三个字节编码

    其他极少使用的Unicode辅助字符，使用四字节编码

2.3字符串中的编码解码问题【应用】

- 相关方法

方法名	说明
byte[] getBytes()	使用平台的默认字符集将该 String编码为一系列字节
byte[] getBytes(String charsetName)	使用指定的字符集将该 String编码为一系列字节
String(byte[] bytes)	使用平台的默认字符集解码指定的字节数组来创建字符串
String(byte[] bytes, String charsetName)	通过指定的字符集解码指定的字节数组来创建字符串

- 代码演示

```
public class StringDemo {
    public static void main(String[] args) throws UnsupportedEncodingException {
        //定义一个字符串
        String s = "中国";

        //byte[] bys = s.getBytes(); //[-28, -72, -83, -27, -101, -67]
        //byte[] bys = s.getBytes("UTF-8"); //[-28, -72, -83, -27, -101, -67]
        byte[] bys = s.getBytes("GBK"); //[-42, -48, -71, -6]
        System.out.println(Arrays.toString(bys));

        //String ss = new String(bys);
        //String ss = new String(bys,"UTF-8");
        String ss = new String(bys,"GBK");
        System.out.println(ss);
    }
}
```

2.4字符流中的编码解码问题【应用】

- 字符流中和编码解码问题相关的两个类
  - InputStreamReader：是从字节流到字符流的桥梁

    它读取字节，并使用指定的编码将其解码为字符

    它使用的字符集可以由名称指定，也可以被明确指定，或者可以接受平台的默认字符集
  - OutputStreamWriter：是从字符流到字节流的桥梁

    是从字符流到字节流的桥梁，使用指定的编码将写入的字符编码为字节

    它使用的字符集可以由名称指定，也可以被明确指定，或者可以接受平台的默认字符集
- 构造方法

方法名	说明
InputStreamReader(InputStream in)	使用默认字符编码创建InputStreamReader对象
InputStreamReader(InputStream in,String chatset)	使用指定的字符编码创建InputStreamReader对象
OutputStreamWriter(OutputStream out)	使用默认字符编码创建OutputStreamWriter对象
OutputStreamWriter(OutputStream out,String charset)	使用指定的字符编码创建OutputStreamWriter对象

- 代码演示

```
public class ConversionStreamDemo {
    public static void main(String[] args) throws IOException {
        //OutputStreamWriter osw = new OutputStreamWriter(new
                    FileOutputStream("myCharStream\\osw.txt"));
        OutputStreamWriter osw = new OutputStreamWriter(new
                    FileOutputStream("myCharStream\\osw.txt"),"GBK");
        osw.write("中国");
        osw.close();

        //InputStreamReader isr = new InputStreamReader(new
                    FileInputStream("myCharStream\\osw.txt"));
        InputStreamReader isr = new InputStreamReader(new
                    FileInputStream("myCharStream\\osw.txt"),"GBK");
        //一次读取一个字符数据
        int ch;
        while ((ch=isr.read())!=-1) {
            System.out.print((char)ch);
        }
        isr.close();
    }
}
```

2.5字符流写数据的5种方式【应用】

- 方法介绍

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

- 刷新和关闭的方法

方法名	说明
flush()	刷新流，之后还可以继续写数据
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

- 代码演示

```
public class OutputStreamWriterDemo {
    public static void main(String[] args) throws IOException {
        OutputStreamWriter osw = new OutputStreamWriter(new
FileOutputStream("myCharStream\\osw.txt"));

        //void write(int c): 写一个字符
//        osw.write(97);
//        osw.write(98);
//        osw.write(99);

        //void writ(char[] cbuf): 写入一个字符数组
        char[] chs = {'a', 'b', 'c', 'd', 'e'};
//        osw.write(chs);

        //void write(char[] cbuf, int off, int len): 写入字符数组的一部分
//        osw.write(chs, 0, chs.length);
//        osw.write(chs, 1, 3);

        //void write(String str): 写一个字符串
//        osw.write("abcde");

        //void write(String str, int off, int len): 写一个字符串的一部分
//        osw.write("abcde", 0, "abcde".length());
        osw.write("abcde", 1, 3);

        //释放资源
        osw.close();
    }
}
```

2.6字符流读数据的2种方式【应用】

- 方法介绍

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

- 代码演示

```
public class InputStreamReaderDemo {
    public static void main(String[] args) throws IOException {

        InputStreamReader isr = new InputStreamReader(new
FileInputStream("myCharStream\\ConversionStreamDemo.java"));

        //int read(): 一次读一个字符数据
//        int ch;
//        while ((ch=isr.read())!=-1) {
//            System.out.print((char)ch);
//        }

        //int read(char[] cbuf): 一次读一个字符数组数据
        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();
    }
}
```

2.7字符流复制Java文件【应用】

- 案例需求

  把模块目录下的“ConversionStreamDemo.java” 复制到模块目录下的“Copy.java”

- 实现步骤
  - 根据数据源创建字符输入流对象
  - 根据目的地创建字符输出流对象
  - 读写数据，复制文件
  - 释放资源
- 代码实现

```
public class CopyJavaDemo01 {
    public static void main(String[] args) throws IOException {
        //根据数据源创建字符输入流对象
```

```
        InputStreamReader isr = new InputStreamReader(new
FileInputStream("myCharStream\\ConversionStreamDemo.java"));
        //根据目的地创建字符输出流对象
        OutputStreamWriter osw = new OutputStreamWriter(new
FileOutputStream("myCharStream\\Copy.java"));

        //读写数据，复制文件
        //一次读写一个字符数据
//        int ch;
//        while ((ch=isr.read())!=-1) {
//            osw.write(ch);
//        }

        //一次读写一个字符数组数据
        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();
    }
}
```

2.8字符流复制Java文件改进版【应用】

- 案例需求

  使用便捷流对象，把模块目录下的“ConversionStreamDemo.java” 复制到模块目录下的“Copy.java”

- 实现步骤
  - 根据数据源创建字符输入流对象
  - 根据目的地创建字符输出流对象
  - 读写数据，复制文件
  - 释放资源
- 代码实现

```
public class CopyJavaDemo02 {
    public static void main(String[] args) throws IOException {
        //根据数据源创建字符输入流对象
        FileReader fr = new FileReader("myCharStream\\ConversionStreamDemo.java");
        //根据目的地创建字符输出流对象
        FileWriter fw = new FileWriter("myCharStream\\Copy.java");

        //读写数据，复制文件
//        int ch;
//        while ((ch=fr.read())!=-1) {
//            fw.write(ch);
//        }

        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();
    }
}
```

2.9字符缓冲流【应用】

- 字符缓冲流介绍
  - 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

方法名	说明
BufferedWriter(Writer out)	创建字符缓冲输出流对象
BufferedReader(Reader in)	创建字符缓冲输入流对象

- 代码演示

```
public class BufferedStreamDemo01 {
    public static void main(String[] args) throws IOException {
        //BufferedWriter(Writer out)
        BufferedWriter bw = new BufferedWriter(new
                            FileWriter("myCharStream\\bw.txt"));
        bw.write("hello\r\n");
        bw.write("world\r\n");
        bw.close();

        //BufferedReader(Reader in)
        BufferedReader br = new BufferedReader(new
                           FileReader("myCharStream\\bw.txt"));

        //一次读取一个字符数据
//        int ch;
//        while ((ch=br.read())!=-1) {
//            System.out.print((char)ch);
//        }

        //一次读取一个字符数组数据
        char[] chs = new char[1024];
        int len;
        while ((len=br.read(chs))!=-1) {
```

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

        br.close();
    }
}
```

2.10字符缓冲流复制Java文件【应用】

- 案例需求

  把模块目录下的ConversionStreamDemo.java 复制到模块目录下的 Copy.java

- 实现步骤
  - 根据数据源创建字符缓冲输入流对象
  - 根据目的地创建字符缓冲输出流对象
  - 读写数据，复制文件，使用字符缓冲流特有功能实现
  - 释放资源
- 代码实现

```
public class CopyJavaDemo01 {
    public static void main(String[] args) throws IOException {
        //根据数据源创建字符缓冲输入流对象
        BufferedReader br = new BufferedReader(new
FileReader("myCharStream\\ConversionStreamDemo.java"));
        //根据目的地创建字符缓冲输出流对象
        BufferedWriter bw = new BufferedWriter(new
FileWriter("myCharStream\\Copy.java"));

        //读写数据，复制文件
        //一次读写一个字符数据
//        int ch;
//        while ((ch=br.read())!=-1) {
//            bw.write(ch);
//        }

        //一次读写一个字符数组数据
        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();
    }
}
```

2.11字符缓冲流特有功能【应用】

- 方法介绍

BufferedWriter：

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

BufferedReader:

方法名	说明
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

- 代码演示

```
public class BufferedStreamDemo02 {
    public static void main(String[] args) throws IOException {

        //创建字符缓冲输出流
        BufferedWriter bw = new BufferedWriter(new
                        FileWriter("myCharStream\\bw.txt"));

        //写数据
        for (int i = 0; i < 10; i++) {
            bw.write("hello" + i);
            //bw.write("\r\n");
            bw.newLine();
            bw.flush();
        }

        //释放资源
        bw.close();

        //创建字符缓冲输入流
        BufferedReader br = new BufferedReader(new
                        FileReader("myCharStream\\bw.txt"));

        String line;
        while ((line=br.readLine())!=null) {
            System.out.println(line);
        }

        br.close();
    }
}
```

2.12字符缓冲流特有功能复制Java文件【应用】

- 案例需求

  使用特有功能把模块目录下的ConversionStreamDemo.java 复制到模块目录下的 Copy.java

- 实现步骤
  - 根据数据源创建字符缓冲输入流对象
  - 根据目的地创建字符缓冲输出流对象
  - 读写数据，复制文件，使用字符缓冲流特有功能实现
  - 释放资源
- 代码实现

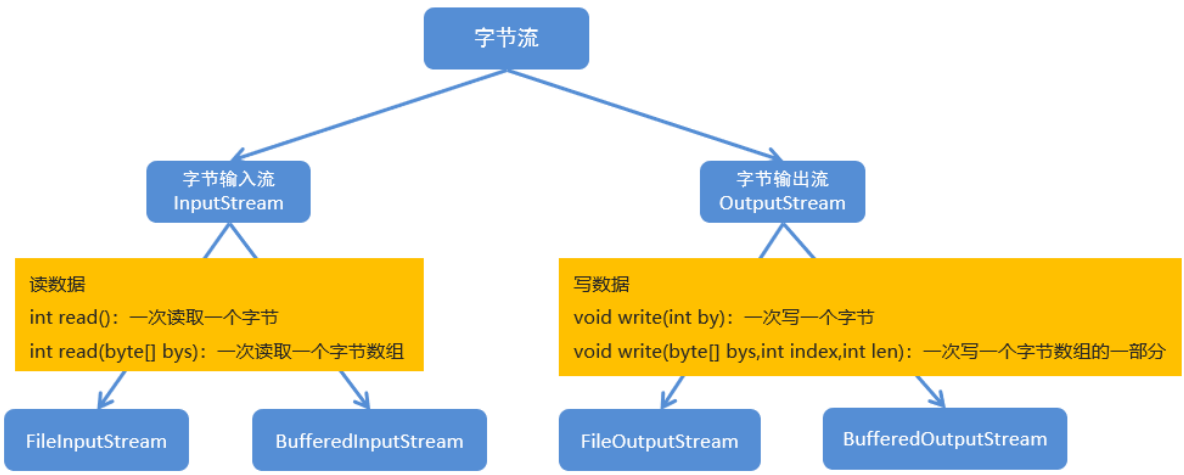
```
public class CopyJavaDemo02 {
    public static void main(String[] args) throws IOException {
        //根据数据源创建字符缓冲输入流对象
        BufferedReader br = new BufferedReader(new
FileReader("myCharStream\\ConversionStreamDemo.java"));
        //根据目的地创建字符缓冲输出流对象
        BufferedWriter bw = new BufferedWriter(new
FileWriter("myCharStream\\Copy.java"));

        //读写数据，复制文件
        //使用字符缓冲流特有功能实现
        String line;
        while ((line=br.readLine())!=null) {
            bw.write(line);
            bw.newLine();
            bw.flush();
        }

        //释放资源
        bw.close();
        br.close();
    }
}
```

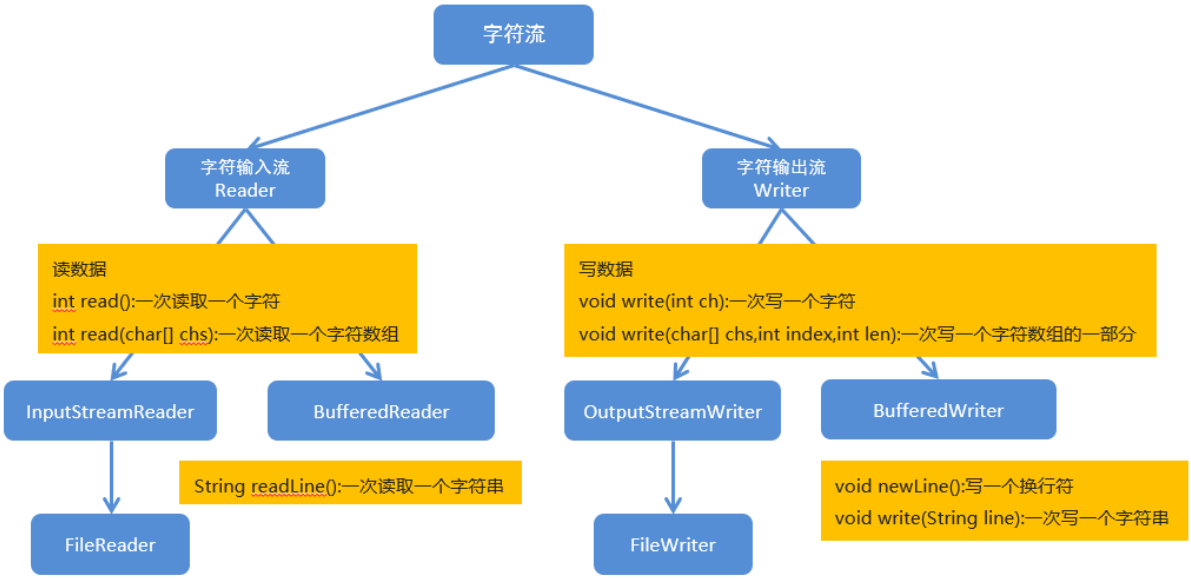
2.13IO流小结【理解】

- 字节流



小结：字节流可以复制任意文件数据，有4种方式一般采用字节缓冲流一次读写一个字节数组的方式

- 字符流



小结：字符流只能复制文本数据，有5种方式，一般采用字符缓冲流的特有功能

3练习案例

3.1集合到文件【应用】

- 案例需求

  把文本文件中的数据读取到集合中，并遍历集合。要求：文件中每一行数据是一个集合元素

- 实现步骤
  - 创建字符缓冲输入流对象
  - 创建ArrayList集合对象
  - 调用字符缓冲输入流对象的方法读数据
  - 把读取到的字符串数据存储到集合中
  - 释放资源
  - 遍历集合
- 代码实现

```
public class TxtToArrayListDemo {
    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("myCharStream\\array.txt"));

        //创建ArrayList集合对象
        ArrayList<String> array = new ArrayList<String>();

        //调用字符缓冲输入流对象的方法读数据
        String line;
        while ((line=br.readLine())!=null) {
            //把读取到的字符串数据存储到集合中
            array.add(line);
        }
```

```
        //释放资源
        br.close();
        //遍历集合
        for(String s : array) {
            System.out.println(s);
        }
    }
}
```

3.2文件到集合【应用】

- 案例需求

  把ArrayList集合中的字符串数据写入到文本文件。要求：每一个字符串元素作为文件中的一行数据

- 实现步骤
  - 创建ArrayList集合
  - 往集合中存储字符串元素
  - 创建字符缓冲输出流对象
  - 遍历集合，得到每一个字符串数据
  - 调用字符缓冲输出流对象的方法写数据
  - 释放资源
- 代码实现

```
public class ArrayListToTxtDemo {
    public static void main(String[] args) throws IOException {
        //创建ArrayList集合
        ArrayList<String> array = new ArrayList<String>();

        //往集合中存储字符串元素
        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("myCharStream\\array.txt"));

        //遍历集合，得到每一个字符串数据
        for(String s : array) {
            //调用字符缓冲输出流对象的方法写数据
            bw.write(s);
            bw.newLine();
            bw.flush();
        }

        //释放资源
        bw.close();
    }
}
```

3.3点名器【应用】

- 案例需求

  我有一个文件里面存储了班级同学的姓名，每一个姓名占一行，要求通过程序实现随点名器

- 实现步骤
  - 创建字符缓冲输入流对象
  - 创建ArrayList集合对象
  - 调用字符缓冲输入流对象的方法读数据
  - 把读取到的字符串数据存储到集合中
  - 释放资源
  - 使用Random产生一个随机数，随机数的范围在：[0,集合的长度)
  - 把第6步产生的随机数作为索引到ArrayList集合中获取值
  - 把第7步得到的数据输出在控制台
- 代码实现

```
public class CallNameDemo {
    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("myCharStream\\names.txt"));

        //创建ArrayList集合对象
        ArrayList<String> array = new ArrayList<String>();

        //调用字符缓冲输入流对象的方法读数据
        String line;
        while ((line=br.readLine())!=null) {
            //把读取到的字符串数据存储到集合中
            array.add(line);
        }

        //释放资源
        br.close();

        //使用Random产生一个随机数，随机数的范围在：[0,集合的长度)
        Random r = new Random();
        int index = r.nextInt(array.size());

        //把第6步产生的随机数作为索引到ArrayList集合中获取值
        String name = array.get(index);

        //把第7步得到的数据输出在控制台
        System.out.println("幸运者是：" + name);
    }
}
```

3.4集合到文件改进版【应用】

- 案例需求

  把ArrayList集合中的学生数据写入到文本文件。要求：每一个学生对象的数据作为文件中的一行数据 格式：学号,姓名,年龄,居住地 举例：itheima001,林青霞,30,西安

- 实现步骤
  - 定义学生类
  - 创建ArrayList集合
  - 创建学生对象
  - 把学生对象添加到集合中
  - 创建字符缓冲输出流对象
  - 遍历集合，得到每一个学生对象
  - 把学生对象的数据拼接成指定格式的字符串
  - 调用字符缓冲输出流对象的方法写数据
  - 释放资源
- 代码实现
  - 学生类

```
public class Student {
    private String sid;
    private String name;
    private int age;
    private String address;

    public Student() {
    }

    public Student(String sid, String name, int age, String address) {
        this.sid = sid;
        this.name = name;
        this.age = age;
        this.address = address;
    }

    public String getSid() {
        return sid;
    }

    public void setSid(String sid) {
        this.sid = sid;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
```

```
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }
}
```

- 测试类

```
public class ArrayListToFileDemo {
    public static void main(String[] args) throws IOException {
        //创建ArrayList集合
        ArrayList<Student> array = new ArrayList<Student>();

        //创建学生对象
        Student s1 = new Student("itheima001", "林青霞", 30, "西安");
        Student s2 = new Student("itheima002", "张曼玉", 35, "武汉");
        Student s3 = new Student("itheima003", "王祖贤", 33, "郑州");

        //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("myCharStream\\students.txt"));

        //遍历集合，得到每一个学生对象
        for (Student s : array) {
            //把学生对象的数据拼接成指定格式的字符串
            StringBuilder sb = new StringBuilder();

 sb.append(s.getSid()).append(",").append(s.getName()).append(",").append(s.ge
tAge()).append(",").append(s.getAddress());

            //调用字符缓冲输出流对象的方法写数据
            bw.write(sb.toString());
            bw.newLine();
            bw.flush();
        }

        //释放资源
        bw.close();
    }
}
```

3.5文件到集合改进版【应用】

- 案例需求

  把文本文件中的数据读取到集合中，并遍历集合。要求：文件中每一行数据是一个学生对象的成员变量值 举例：itheima001,林青霞,30,西安

- 实现步骤
  - 定义学生类
  - 创建字符缓冲输入流对象
  - 创建ArrayList集合对象
  - 调用字符缓冲输入流对象的方法读数据
  - 把读取到的字符串数据用split()进行分割，得到一个字符串数组
  - 创建学生对象
  - 把字符串数组中的每一个元素取出来对应的赋值给学生对象的成员变量值
  - 把学生对象添加到集合
  - 释放资源
  - 遍历集合
- 代码实现
  - 学生类

    同上

  - 测试类

```
public class FileToArrayListDemo {
    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("myCharStream\\students.txt"));

        //创建ArrayList集合对象
        ArrayList<Student> array = new ArrayList<Student>();

        //调用字符缓冲输入流对象的方法读数据
        String line;
        while ((line = br.readLine()) != null) {
            //把读取到的字符串数据用split()进行分割，得到一个字符串数组
            String[] strArray = line.split(",");

            //创建学生对象
            Student s = new Student();
            //把字符串数组中的每一个元素取出来对应的赋值给学生对象的成员变量值
            //itheima001,林青霞,30,西安
            s.setSid(strArray[0]);
            s.setName(strArray[1]);
            s.setAge(Integer.parseInt(strArray[2]));
            s.setAddress(strArray[3]);

            //把学生对象添加到集合
            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.getAddress());
        }
    }
}
```